Deterministic Optimization Methods For the Haplotyping Problem

Xiang-Sun Zhang

Academy of Mathematics & Systems Science,
Chinese Academy of Science

zxs@amt.ac.cn
http://zhangroup.aporc.org

May, 2005

Contents

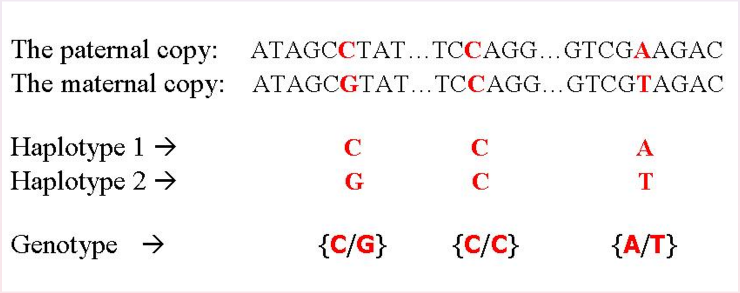
Background

- All humans share about 99.9% identity at the DNA level
- The differences in DNA sequences in a population are called **polymophisms**
- Such regions of variations of DNA sequences are responsible for genetics diseases and phenotype differences
- Therefore, the next important research area is to find the association relationship between DNA variations and genetic disease

Background

- **Single nucleotide polymorphism (SNP)** is a single DNA base where two different nucleotides appear with sufficient frequency in a population
- SNP is the most frequent and important form among various genetic variations of DNA sequences
- SNPs are found approximately every 1000 base pairs in the human genome

Background



The paternal copy:	ATAGC**C**TAT…	TC**C**AGG…	GTCG**A**AGAC
The maternal copy:	ATAGC**G**TAT…	TC**C**AGG…	GTCG**T**AGAC
Haplotype 1 →	**C**	**C**	**A**
Haplotype 2 →	**G**	**C**	**T**
Genotype →	{**C**/**G**}	{**C**/**C**}	{**A**/**T**}

Background

Haplotypes generally have more information content than individual SNPs and genotype in disease association studies, but it is substantially difficult to determine haplotypes through experiments

We generally have two kinds of data resource:

- short haplotype fragments (SNP fragments) from shortgun experiments
- a set of genotype information from a population

Background

Then we have two different problems:

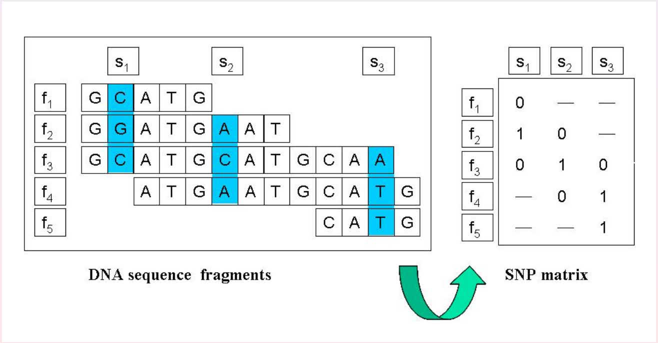
- **Haplotype Assembly** for an individual
  - Assembly a pair of haplotypes from short SNP fragments
- **Haplotype Inference** in a population
  - Infer haplotypes based on the genotype samples in a population

Contents

Modeling

- **Problem:** Given a set of DNA fragments coming from a chromosome by a sequencing method, retrieve a pair of hapltoypes according to the SNP states in DNA fragments
- How to formulate it into a mathematical problem (a combinatorial optimization problem)?

From DNA fragments to SNP matrix



$$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ \hline f_1 & 0 & - & - \\ f_2 & 1 & 0 & - \\ f_3 & 0 & 1 & 0 \\ f_4 & - & 0 & 1 \\ f_5 & - & - & 1 \end{array}$$

Modeling

Conflicts come from two reasons:

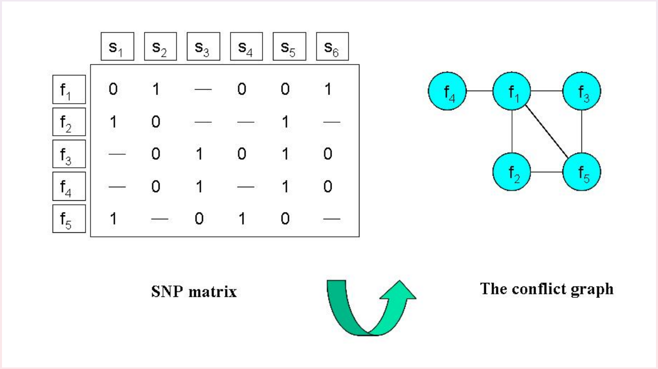
- Conflict between two fragments belong to the two different copies
- Conflict between two fragments from the same copy but with experiment errors

Modeling

Make a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$,

- all fragments consist of the vertex set $\mathbf{V}$
- two conflicting fragments (vertices) are connected by an edge in $\mathbf{E}$.

Conflict graph



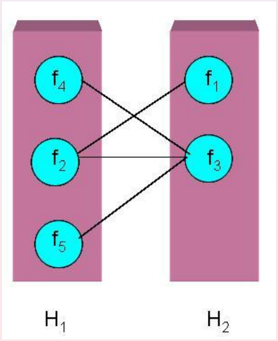
	s$_1$	s$_2$	s$_3$	s$_4$	s$_5$	s$_6$
f$_1$	0	1	—	0	0	1
f$_2$	1	0	—	—	1	—
f$_3$	—	0	1	0	1	0
f$_4$	—	0	1	—	1	0
f$_5$	1	—	0	1	0	—

SNP matrix

The conflict graph

When DNA fragments are error-free

When data has no errors, the conflict graph is a bipartite graph (*a graph which can be decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent*)

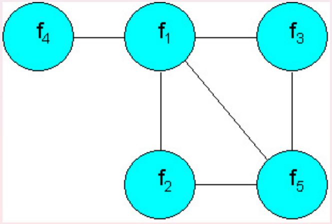


When DNA fragments have errors

- A graph can be tested for bipartiteness using *BipartiteQ* in Mathematica 5.1
- A graph is bipartite if and only if it has no odd cycles (a cycle with odd number of edges) (S.Skiena, 1990)
- How to retrieve the haplotypes from data with errors $\Leftrightarrow$ How to make a graph bipartite?

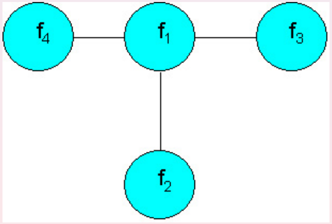
When DNA fragments have errors

Omit some vertices to obtain a bipartite graph, that means delete some contaminated fragments



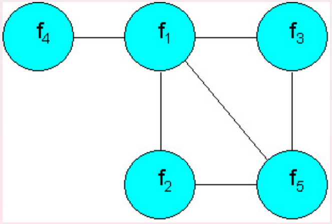
When DNA fragments have errors

Omit vertices to obtain a bipartite graph



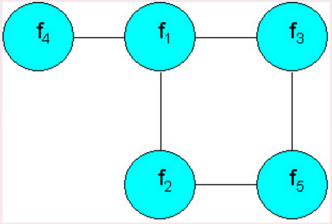
When DNA fragments have errors

Omit edges to obtain a bipartite graph, that means remove some SNP sites or flip some SNP values



When DNA fragments are have errors

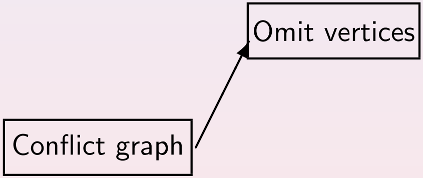
Omit edges to obtain a bipartite graph



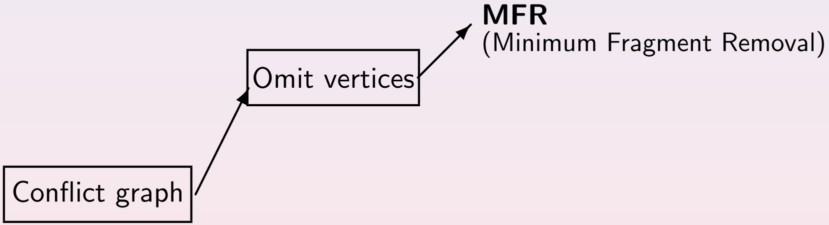
Now we can review the modeling system by above concept

$\boxed{\text{Conflict graph}}$

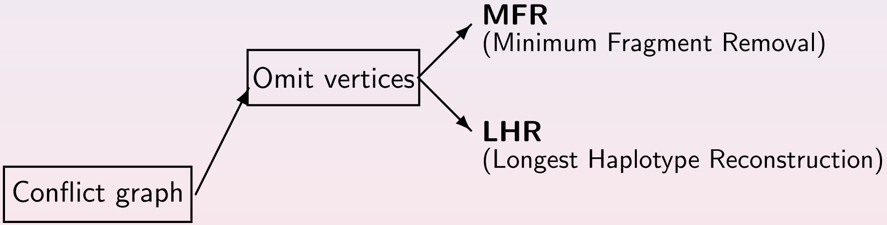
Now we can review the modeling system by above concept



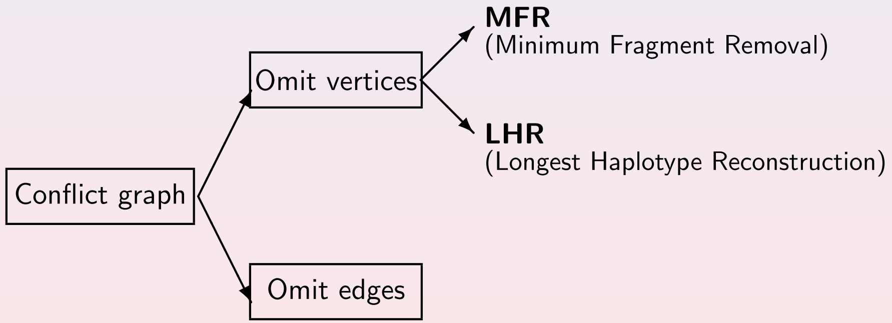
Now we can review the modeling system by above concept



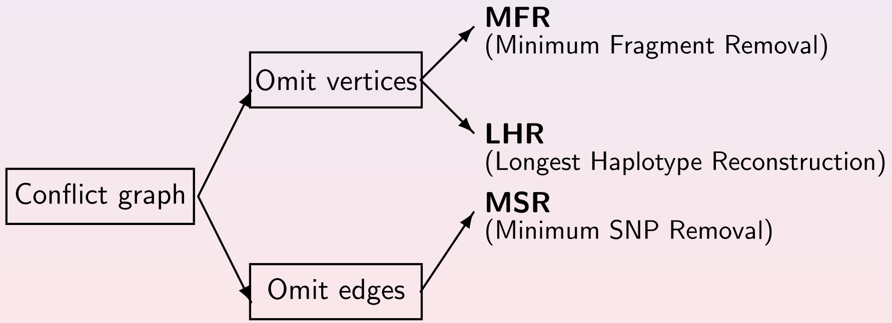
Now we can review the modeling system by above concept


Conflict graph
Omit vertices
MFR
(Minimum Fragment Removal)
LHR
(Longest Haplotype Reconstruction)

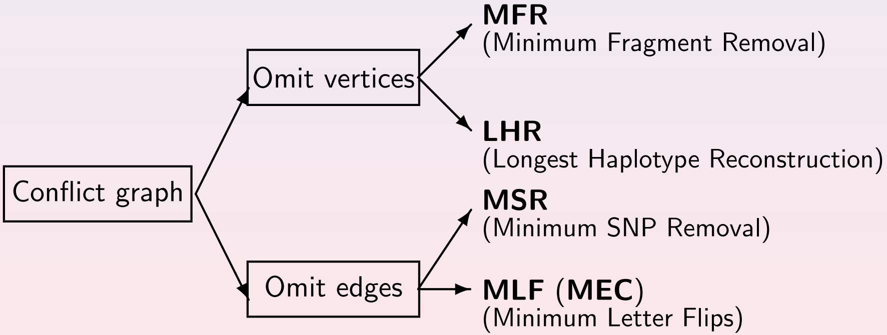
Now we can review the modeling system by above concept


Conflict graph
Omit vertices
Omit edges
MFR
(Minimum Fragment Removal)
LHR
(Longest Haplotype Reconstruction)

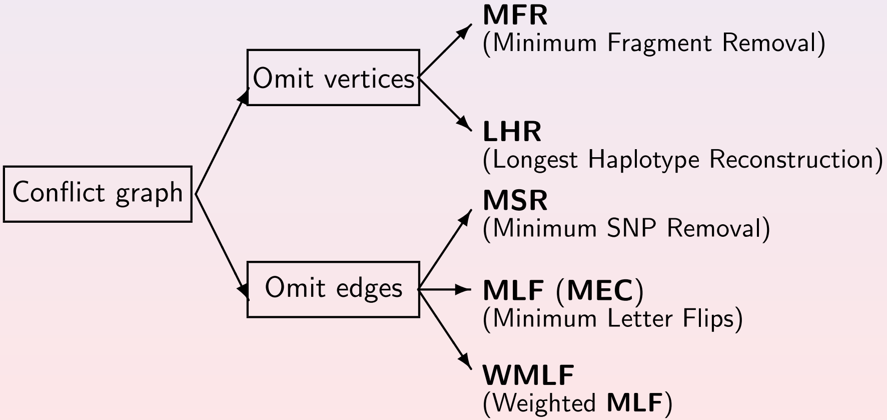
Now we can review the modeling system by above concept



Now we can review the modeling system by above concept



Now we can review the modeling system by above concept



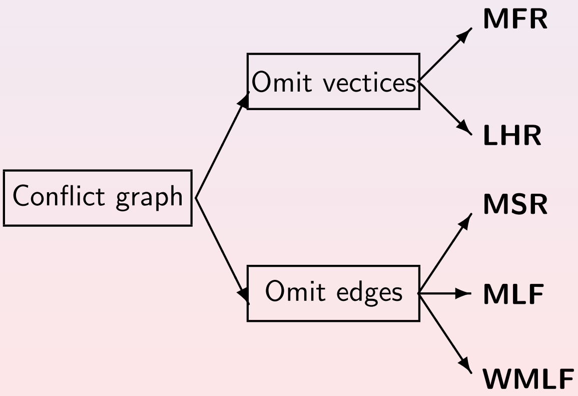
Several combinatorial optimization models

- Cut some vertices on the odd cycles to make the remained graph bipartite:
    - remove a minimum number of fragments (rows) so that the graph is bipartite (the resulted matrix is feasible)— **MFR**: *Minimum Fragment Removal*;
    - remove a set of fragments so that the resulted matrix is feasible and the sum of the lengths of the derived haplotypes is maximized—**LHR**: *Longest Haplotype Reconstruction*.

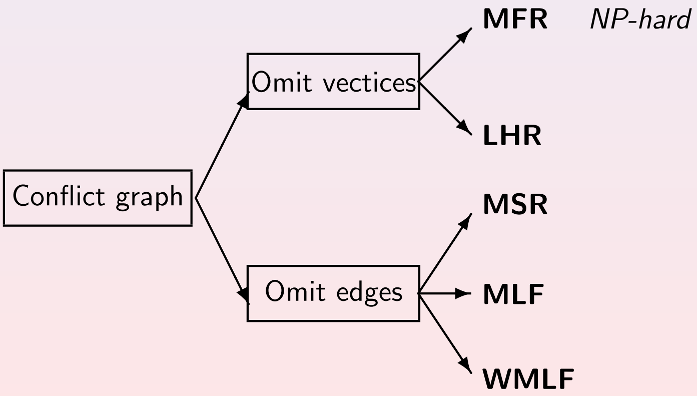
Several combinatorial optimization models (continue)

- Cut some arcs on the odd cycles to make the remained graph bipartite (the matrix feasible):
    - remove a minimum number of SNPs (columns) so that the matrix is feasible—**MSR**: *Minimum SNP Removal*;
    - flip a minimum number of site values so that the matrix is feasible— **MLF**: *Minimum Letter Flips*. Or in some papers, **MEC**: *Minimum Error Correction*.
    - Weighted **MLF** (**WMLF**): flip some letters so that the weighted sum of the flips is minimum and the resulted SNP matrix is feasible.

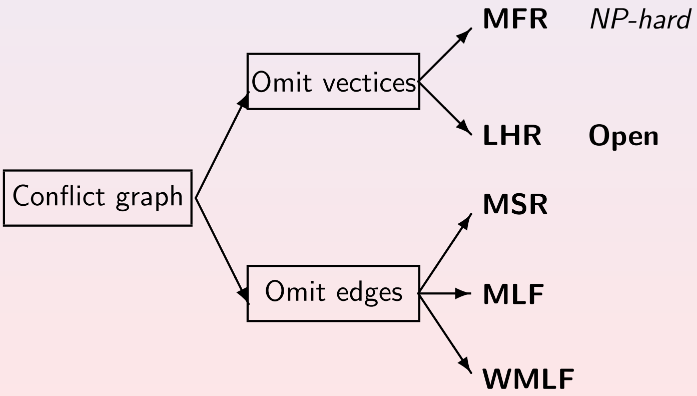
The complexity of these problems



The complexity of these problems



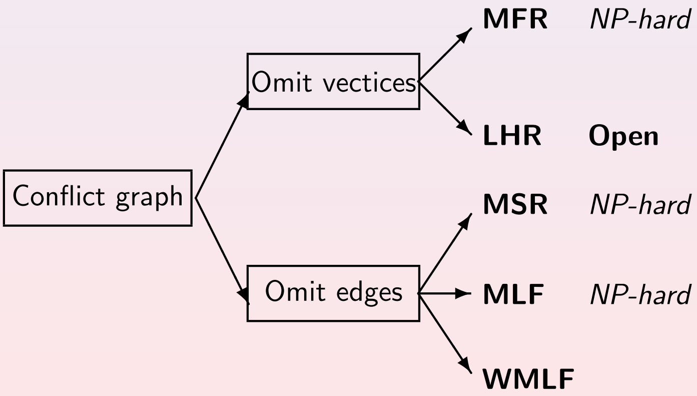
The complexity of these problems



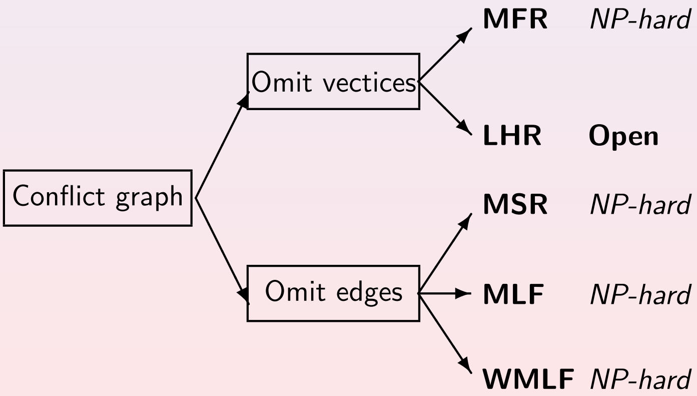
The complexity of these problems

$$\begin{tikzcd}
 & & \textbf{MFR} & \textit{NP-hard} \\
 & \boxed{\text{Omit vectices}} \arrow[ur] \arrow[dr] & & \\
 & & \textbf{LHR} & \textbf{Open} \\
\boxed{\text{Conflict graph}} \arrow[uur] \arrow[ddr] & & & \\
 & & \textbf{MSR} & \textit{NP-hard} \\
 & \boxed{\text{Omit edges}} \arrow[ur] \arrow[r] \arrow[dr] & \textbf{MLF} & \\
 & & \textbf{WMLF} &
\end{tikzcd}$$

The complexity of these problems



The complexity of these problems



The complexity of these problems

- **MFR** is NP-hard even for SNP matrices in which each fragment has at most one gap
- **LHR** has polynomial-time algorithm when fragments are gapless, but the complexity of the general case is open
- **MSR** is NP-hard for SNP matrices with at most two gaps per fragment
- The general **MLF** (**MEC**) is NP-hard
- **WMLF** is NP-hard even if its fragments are gapless

Contents

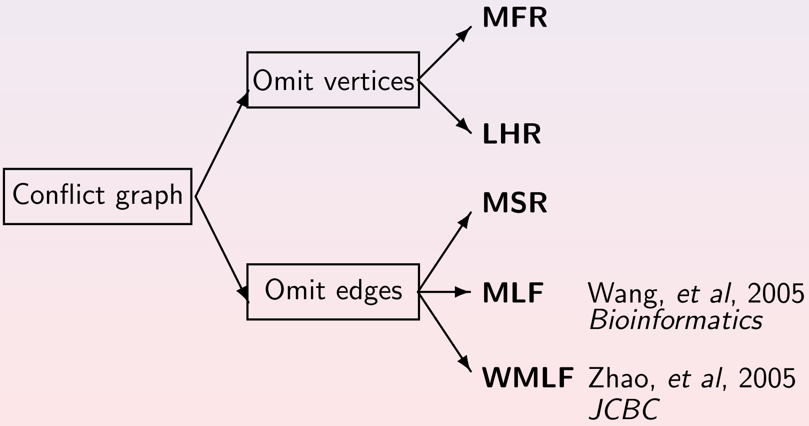
Algorithms

- Algorithms for **MFR** (*Minimum Fragment Removal*)
  - A dynamic programming algorithm with complexity $O(2^{2k}m^2n + 2^{3k}n^3)$ is given by Rizzi R., *et al*, 2002, where k is the maximum number of gaps in the fragments.
- Algorithms for **MSR** (*Minimum SNP Removal*)
  - A dynamic programming algorithm with complexity $O(mn^{2k+2})$ is given by Rizzi R., *et al*, 2002.

Algorithms (continue)

- Algorithms for **MLF (MEC)** (*Minimum Letter Flips, Minimum Error Correction*).
  - An exact algorithm based on branch-and-bound method and a heuristic method based on genetic algorithm (GA) are proposed to solve **MEC** in Wang R.-S., *et al*, 2005.
- Algorithms for **WMLF** (*Weighted MLF*)
  - A heuristic algorithm based on dynamic clustering method is presented in Zhao Y.-Y *et al*, 2005 for **WMLF**.

Our group's work:


Conflict graph
Omit vertices
Omit edges
MFR
LHR
MSR
MLF
Wang, *et al*, 2005 *Bioinformatics*
WMLF
Zhao, *et al*, 2005 *JCBC*

Contents

Modeling

- A haplotype **h** is a vector $(h_1, \cdots, h_n)$ over $\{0,1\}^n$.
- A genotype **g** is a vector $g_1, \cdots, g_n$ over $\{0,1,2\}^n$.
- A pair of haplotypes $(\mathbf{h}^1, \mathbf{h}^2)$ is called **compatible** with a **g** if

$$h_i^1 = h_i^2 = g_i = \begin{cases} 0, & h_i^1, h_i^2 \text{ are wild} \\ 1, & h_i^1, h_i^2 \text{ are mutant}, \end{cases}$$

$$h_i^1 \neq h_i^2 \Leftrightarrow g_i = 2, \quad \text{the } i\text{th SNP site is heterozygous.}$$

Modeling

- The **haplotype inference problem** is:

  Given a set of genotypes G, find a set of haplotypes H, such that for every genotype $\mathbf{g} \in G$, there exists at least one pair of haplotypes in H which are compatible this genotype.

Several versions for haplotype inference

There two problem formulations:

- Find the most likely haplotype (**MLH**) configuration for each genotype $\mathbf{g} \in G$.
- Find a set of haplotypes by some parsimony rule

Several versions for haplotype inference (continue)

- **Parsimony haplotype inference problem**:
  - **MRG** problem — Based on Clark's *inference rule* (Clark, 1990), Gusfield D., 2003 employed a graph-theoretic view to express and analyze the inference problem.
  - Inference by pure parsimony (**HIPP**): Find a cardinality-smallest set H such that for each $g \in G$, there is a haplotype configuration made by two sequences in H.

The complexity of these problems

- The **MLH** (*most likely haplotype*) model is solved by stochastic methods, such as Markov chain model, maximum likelihood estimation.
- The **MRG** (model is proved NP-hard.
- The **HIPP** model is also an NP-hard problem.

Contents

Algorithms

- Algorithms for **MLH** (*most likely haplotype*)
  - A partition-ligation algorithm (an exhaustive search approach) is used to find the most probable haplotypes.
  - A dynamic programming algorithm based on the Markov chain framework is developed in Zhang J.-H *et al.*, 2005.

Algorithms (continue)

- Algorithms for the deterministic parsimony rule:
  - The **MRG** model by Gusfield, 2003 can be exactly formulated as an integer linear programming.
  - Algorithms for the **HIPP** problem are still in development
    - A branch-and-bound method by Wang and Xu, 2003.
    - **A tree-grow method** with complexity of $O(m^2n)$ by Li, Zhang and Chen, 2005.

Basic ideas of **TGM**

- Resolve columns, one by one, of the genotype matrix $\mathbf{G}$ by haplotype fragments; Let

$$\mathbf{G} = (\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \cdots, \hat{\mathbf{g}}_n)$$

  Then **TGM** solves $(\hat{\mathbf{g}}_1), (\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2), \cdots, \mathbf{G}$ successively.

- Extend the haplotype fragments in growing length by keeping all corresponding genotype fragments resolved;

Basic ideas of **TGM** (continued

- Use a growing tree to represent the haplotype fragments developing. Making a haplotype fragment one site longer means to add a branch to the existing tree and for resolving the corresponding longer genotype fragment.
- Carefully add a new branch to reach the parsimony effect, that is for each $(\hat{\mathbf{g}}_1, \cdots, \hat{\mathbf{g}}_k)$, $k = 1, 2, \cdots, n$, the tree solves it is the **smallest** one.

Algorithm of **TGM**

Initialization:

Input an $m \times n$ **G**. Set a root node v_{01}, $v_{01} = \{1, \cdots, m\}$. Set $f(i) = \mathit{false}$, for every $i = 1, \cdots, m$. Let $j = 0$, and go to step 1.

Step 1 Resolve submatrix $\mathbf{G}[1, j+1]$.
Suppose that there are p nodes $v_{j1}, \cdots, v_{jk}, \cdots, v_{jp}$ in the j-th layer of the growing-tree representing p distinct haplotype fragments resolving $\mathbf{G}[1, j]$. $v_{j1}, \cdots, v_{jp}$ also represent corresponding index sets. Do Substeps 1.1 and 1.2 depicted below.

Algorithm of **TGM**

Substep 1.1 For each $1 \leq k \leq p$, and each i, $(1 \leq i \leq m)$, if $i \in v_{jk}$, resolve the i-th genotype fragment in $\mathbf{G}[1, j]$ when i satisfies either of the following two conditions:
Condition 1: $g_{i,j+1} \neq 2$;
Condition 2: $g_{i,j+1} = 2$, and $f(i) = \mathit{false}$.
Otherwise, record the i in a set $I(j)$; and record v_{jk} in a node set T_{ij}, where T_{ij} is a set of the j-th layer nodes that include node i.

Algorithm of **TGM**

- if $g_{i,j+1} = 0$, then add a branch 0 to v_{jk} when there is no branch 0 growing from v_{jk}; add i to $v_{(j+1)\cdot}$, which is connected to the node v_{jk} by the existing or just added branch 0.
- if $g_{i,j+1} = 1$, then add a branch 1 to v_{jk} when there is no branch 1 growing from v_{jk}; add i to $v_{(j+1)\cdot}$, which is connected to v_{jk} by the existing or just added branch 1.
- if $g_{i,j+1} = 2$ and $f(i) = \textit{false}$, then add a branch 0 or 1, or both branches 0 and 1 or nothing to v_{jk} according to the following cases: only one type exists, no branch exists, or two types of branches exist. Add i into both index sets of the $(j+1)$-th layer nodes connected to node v_{jk}, set $f(i) = \textit{true}$.

Algorithm of **TGM**

Substep 1.2 For $i \in I(j)$, suppose $T_{ij} = \{v_{jk_1}, v_{jk_1}\}$, i belongs to v_{jk_1} and v_{jk_2}. Check whether there are two different branch types growing separately from v_{jk_1} and v_{jk_2}.

1. If there are no such two different types of branches, then add a proper type of branch to v_{jk_1} or v_{jk_2}, or add two different types , one to v_{jk_1} while the other to v_{jk_2}.
2. Choose a pair of different types, one growing from v_{jk_1}, the other from v_{jk_2}. Add i into both index sets of the $(j+1)$-th layer which are connected to v_{jk_1} or v_{jk_2} by one of the chosen branches.

Algorithm of **TGM**

Step 2 If $j + 1 < n$, set $j := j + 1$, and return to Step 1. Otherwise assemble haplotypes as follows. Trace each path from v_{01} to every node in the n-th layer. The sequence of branch type indices (0 or 1) of the path gives a haplotype, which can be used to resolve the genotypes whose indices belong to the corresponding node in the n-th layer. All the haplotypes corresponding to the n-th layer nodes consist of $\mathcal{H}(\mathbf{G})$.

An Example

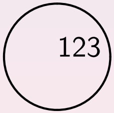
Given a genotype matrix

$$\mathbf{G} = \begin{pmatrix} 2 & 2 & 0 \\ 2 & 0 & 2 \\ 0 & 2 & 2 \end{pmatrix} \quad (1)$$

The columns are

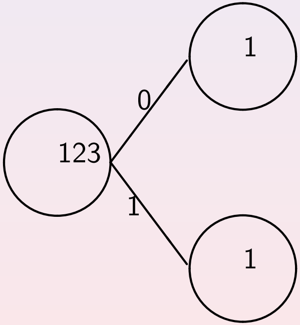
$$\hat{\mathbf{g}}_1 = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}, \hat{\mathbf{g}}_2 = \begin{pmatrix} 2 \\ 0 \\ 2 \end{pmatrix}, \hat{\mathbf{g}}_3 = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix} \quad (2)$$

Solving $\hat{\mathbf{g}}_1 = (2, 2, 0)^T$



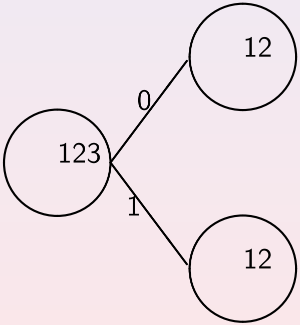
Set $f(1) = \mathit{False}, f(2) = \mathit{False}, f(3) = \mathit{False}$

Solving $\hat{\mathbf{g}}_1 = (2, 2, 0)^T$



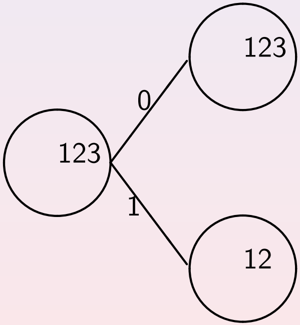
Set $f(1) = \textit{Ture}, f(2) = \textit{False}, f(3) = \textit{False}$

Solving $\hat{\mathbf{g}}_1 = (2, 2, 0)^T$



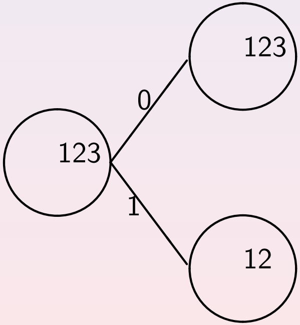
Set $f(1) = \mathit{Ture}, f(2) = \mathit{Ture}, f(3) = \mathit{False}$

Solving $\hat{\mathbf{g}}_1 = (2, 2, 0)^T$



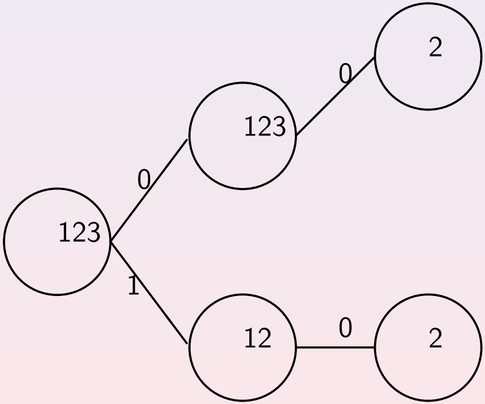
Set $f(1) = \mathit{Ture}, f(2) = \mathit{Ture}, f(3) = \mathit{False}$

Solving $\hat{\mathbf{g}}_2 = (2, 0, 2)^T$



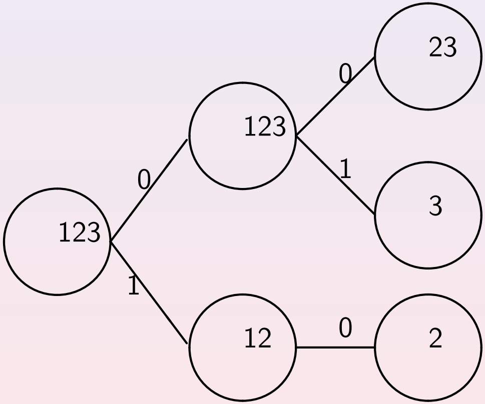
Set $f(1) = \mathit{Ture}, f(2) = \mathit{Ture}, f(3) = \mathit{False}$

Solving $\hat{\mathbf{g}}_2 = (2, 0, 2)^T$



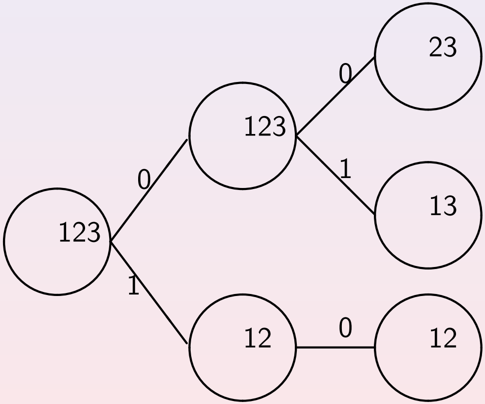
Set $f(1) = Ture, f(2) = Ture, f(3) = False$

Solving $\hat{\mathbf{g}}_2 = (2, 0, 2)^T$



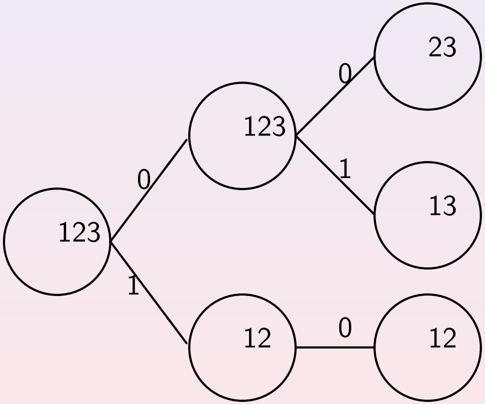
Set $f(1) = Ture, f(2) = Ture, f(3) = Ture$

Solving $\hat{\mathbf{g}}_2 = (2, 0, 2)^T$



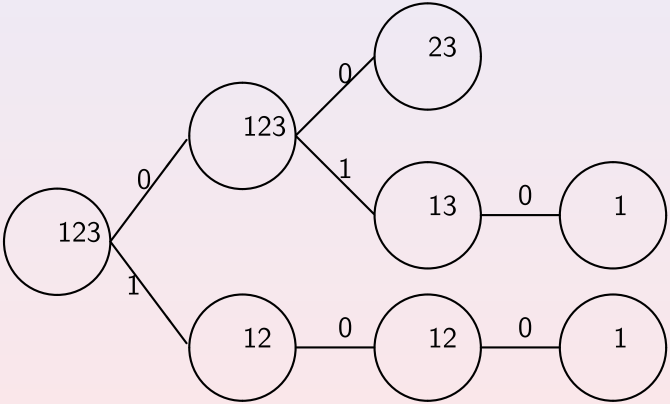
Set $f(1) = Ture, f(2) = Ture, f(3) = Ture$

Solving $\hat{\mathbf{g}}_3 = (0, 2, 2)^T$



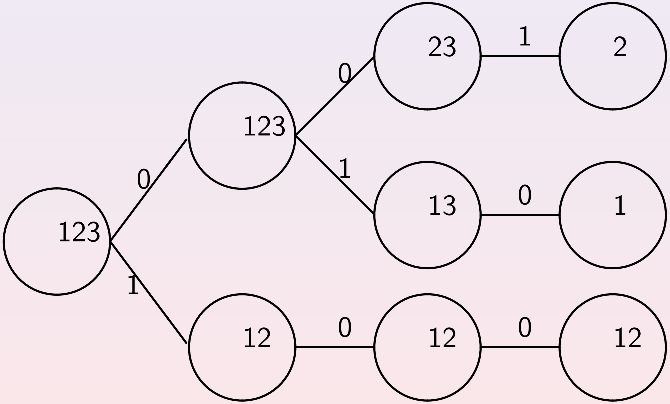
Set $f(1) = Ture, f(2) = Ture, f(3) = Ture$

Solving $\hat{\mathbf{g}}_3 = (0, 2, 2)^T$



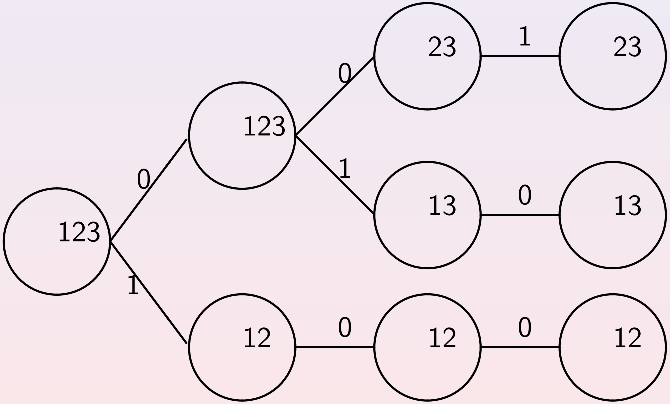
Set $f(1) = \mathit{Ture}, f(2) = \mathit{Ture}, f(3) = \mathit{Ture}$

Solving $\hat{\mathbf{g}}_3 = (0, 2, 2)^T$



Set $f(1) = Ture, f(2) = Ture, f(3) = Ture$

Solving $\hat{\mathbf{g}}_3 = (0, 2, 2)^T$



Set $f(1) = Ture, f(2) = Ture, f(3) = Ture$

Complexity and Convergence Rate

- A convergence analysis and an error bound is given on the base of the microstructure discussion of the genotype matrix **G**.
- **Theorem 1**. Given an $m \times n$ genotype matrix **G**, the computational complexity of **TGM** is $O(m^2 n)$.

Contents

Evaluation criteria

- Reconstruction error rate (**ER**) : to measure the proportion of genotypes which are resolved by a wrong pair of haplotypes.

Data sets

4 experiments

- 18 genotypes coming from $\beta_1 AR$ gene
- 11 genotypes coming from ACE gene
- Simulated genotypes based on Maize data
- Simulated genotypes and haplotypes

Experiment result 1 — on $\beta_1 AR$ data

- The resolution of every genotype obtained by **TGM** is exactly the same as the real ones, that is, with an **ER** 0.
- The total running time is 0.016 second, very efficient in contrast to over a minute for **HAPAR** (*Niu, et.al.*, 2001)and over ten minutes for **PHASE** (*Stephens, et.al.*, 2001).

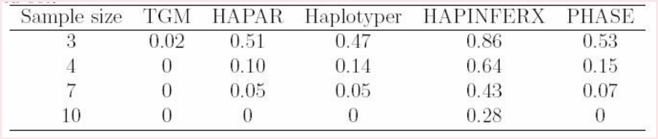
Experiment result 2 — on ACE data

- **TGM** obtained 13 haplotypes with 9 correct haplotypes that resolve 9 out of the 11 genotypes correctly with an **ER** 0.182. It is is better than or at least equal to widely used existing programs,
  - **HAPAR** with **RER** 0.273,
  - **Haplotyper** with **RER** 0.182,
  - **HAPINFERX** with **RER** 0.273,
  - **PHASE** with **RER** 0.273.

Experiment result 3 — on Maize data

Generate a sample of n genotypes each of which is conflated by two randomly picked haplotypes in a set.

- **TGM** correctly resolves all genotypes for sample sizes from 4 to 10, and behaves best among five programs.



Sample size	TGM	HAPAR	Haplotyper	HAPINFERX	PHASE
3	0.02	0.51	0.47	0.86	0.53
4	0	0.10	0.14	0.64	0.15
7	0	0.05	0.05	0.43	0.07
10	0	0	0	0.28	0

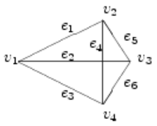
Contents

The hybrid haplotyping problem

- The **MEC/GI** is an **MEC** (*Minimum Error Correction*) with added genotype information:
- Given a SNP matrix $\mathbf{W} = (w_{ij})$ and a genotype $\mathbf{g}$, correct minimum number of elements (0 into 1 or vice versa) so that the resulting matrix is feasible and $\mathbf{g}$-compatible, i.e., the corrected SNP fragments will determine a pair of haplotypes that is compatible with g.

The hybrid haplotyping problem (continue)

- The **MEC/GI** problem can be described as an integer linear programming.
- The **MEC/GI** problem is NP-hard by reduction from **MAX-CUT**:



$\Longrightarrow$

g	0	0	0	0	0	0	2	2	2	2	0	0	0	0	0
e_1	0	0	0	0	0	0	0	1	–	–	–	–	–	–	–
e_2	–	0	0	0	0	0	0	–	1	–	0	–	–	–	–
e_3	–	–	0	0	0	0	0	–	–	1	0	0	–	–	–
e_4	–	–	–	0	0	0	–	0	–	1	0	0	0	–	–
e_5	–	–	–	–	0	0	–	0	1	–	0	0	0	0	–
e_6	–	–	–	–	–	0	–	–	0	1	0	0	0	0	0

The hybrid haplotyping problem (continue)

- A dynamic programming algorithm is given for a special case to illustrate the problem structure.
- A feed-forward neural network is proposed for the general case in **Zhang X.-S** *et al.*, 2005.

NN Algorithms for **MEC/GI**

Using 2 to denote the wild homogenous allele, -2 to denote the mutant homogenous allele, and 0 to denote the heterozygous allele, then a genotype is a vector on $\{2, -2, 0\}$ while a haplotype is a vector on $\{-1, 1\}$. Let

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{in}), \quad i = 1, 2, \cdots, m$$

be m SNP fragments.

NN Algorithms for **MEC/GI**

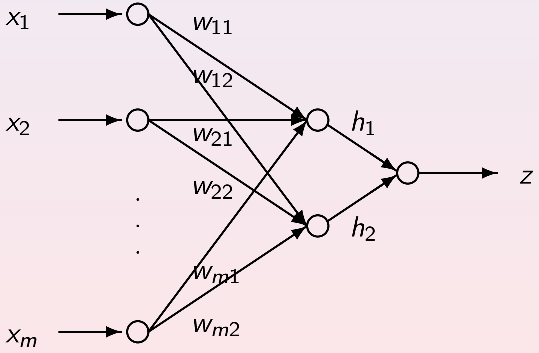
The **MEC/GI** problem is to find a pair of haplotypes $(\mathbf{h}_1, \mathbf{h}_2)$ to minimize

$$\sum_{k=1}^{n} (h_{1k} + h_{2k} - g_k)^2$$

and

$$\sum_{\mathbf{x}_i \in X_1} HD(\mathbf{h}_1, \mathbf{x}_i) + \sum_{\mathbf{x}_i \in X_2} HD(\mathbf{h}_2, \mathbf{x}_i)$$

The structure of a feed-forward neural network



The objectives that neural network learns

- For the neurons corresponding to $\mathbf{h}_1$ ($\mathbf{h}_2$) in the second layer, the network learns to minimize the following error function between $\mathbf{h}_1$ ($\mathbf{h}_2$) and the SNP fragments in X_1 (X_2):

$$f_{21} = \sum_{\mathbf{x}_i \in X_1} \sum_{k=1}^{n} (h_{1k} - x_{ik})^2 |x_{ik}|$$

$$f_{22} = \sum_{\mathbf{x}_i \in X_2} \sum_{k=1}^{n} (h_{2k} - x_{ik})^2 |x_{ik}|$$

The objectives that neural network learns

- The objective that the third layer adjusts to is to minimize the following error function between the output of the third layer and the original genotype:

$$f_1 = \sum_{k=1}^{n} (h_{1k} + h_{2k} - g_k)^2$$

Details of the algorithm

Set parameter values L_1, L_2, ρ, λ and ε. Randomly initiate weight matrix $W(0)$ with $w_{il} \in [0,1],\ i = 1, \cdots, m,\ l = 1, 2.$ $t = 0.$

1. Obtain a pair of two haplotypes (h_1, h_2) according to the current weight matrix;
2. Classify all SNP fragments using (h_1, h_2) and calculate the derivatives $\nabla_{w_{i1}} f_{11},\ \nabla_{w_{i2}} f_{12},\ \nabla_{w_{i1}} f_2,\ \nabla_{w_{i2}} f_2,$ $i = 1, 2, \cdots, m;$

Details of the algorithm

1.
2.
3. Update the current weight matrix $W(t)$ by using the formulae

$$w_1(t+1) = w_1(t) - \rho(L_1 \nabla_{w_1} f_1 + L_2 \nabla_{w_1} f_{21}),$$

$$w_2(t+1) = w_2(t) - \rho(L_1 \nabla_{w_2} f_1 + L_2 \nabla_{w_2} f_{22}),$$

   where ρ is step length and $L_1,\ L_2$ are parameters;
4. Repeat Step 1 to Step 3 until no change occurs for $w_{il}, i = 1, 2, \cdots, m,\ l = 1, 2$, i.e. $||W(t+1) - W(t)|| < \varepsilon$.

Contents

Data sets and evaluation criteria

Data sets:

- 100 pairs of simulated haplotypes,
  $s = 0.5, \;\; s = 0$
- 8 pairs of haplotypes coming from ACE gene
- 129 pairs of haplotypes coming from $5q31$ gene

Data sets and evaluation criteria

Evaluation criteria:

- Haplotype reconstruction rate, Set $r_{ij} = HD(\mathbf{h}_i, \hat{\mathbf{h}}_j),\ i = 1, 2,\ j = 1, 2.$ Define haplotype reconstruction rate **RR**:

$$\mathbf{R}R(\mathbf{h}, \hat{\mathbf{h}}) = 1 - \frac{\min\{r_{11} + r_{22}, r_{12} + r_{21}\}}{2n}$$

- The number of Error Correction

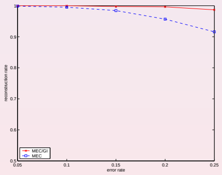
$$E(P) = \sum_{i=1}^{2} \sum_{f \in C_i} HD(f, \mathbf{h}_i).$$

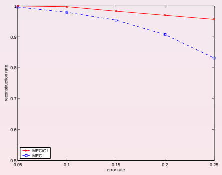
Experiment result 1 — on simulation data set

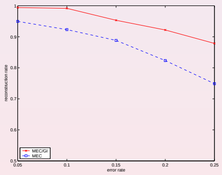
Table: The comparative results of the MEC/GI model and the MEC model

error rate	s=0.5		s=0.0	
	MEC	MEC/GI	MEC	MEC/GI
0.05	0.941	1.000	0.965	0.996
0.1	0.904	0.969	0.950	0.984
0.15	0.863	0.969	0.890	0.946
0.2	0.786	0.908	0.834	0.922
0.25	0.763	0.863	0.766	0.830

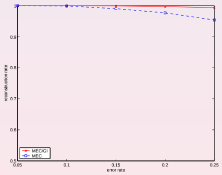
Experiment result 2 — on ACE data set

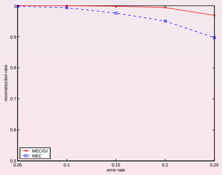


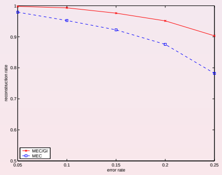




Experiment result 2 — on $5q31$ data set







Thank you