

# Beaches of islands of tractability: Hardness results, exact algorithms and approximation algorithms for parsimony and minimum perfect phylogeny haplotyping problems

---

Leo van Iersel<sup>1</sup>, Judith Keijsper<sup>1</sup>, Steven Kelk<sup>2</sup>, Leen Stougie<sup>1,2</sup>

(1) Technische Universiteit Eindhoven (TU/e)

(2) Centrum voor Wiskunde en Informatica (CWI), Amsterdam

Email: [S.M.Kelk@cwi.nl](mailto:S.M.Kelk@cwi.nl)

Web: <http://homepages.cwi.nl/~kelk>



## *SNPs and Haplotyping (1)*

---

- **Variation in the human population**

- It is widely known that the genomes of any two humans are 99% identical i.e. they have the same nucleotide at more than 99% of the sites along the genome.
- The sites at which significant variation (5% or more) does occur are called **Single Nucleotide Polymorphisms** (SNPs). Dominant form of human genetic variation.
- Example: Suppose you observe that 15% of human chromosomes have an A nucleotide at site X, while the other 85% have a C nucleotide at that site. Then site X is an SNP.
- Although 3-way variation is possible in an SNP (e.g. A, G, and C are each observed at that site in more than 5% of the population), most SNPs are **binary** (e.g. only A and C are observed at that site in more than 5% of the population.)



## SNPs and Haplotyping (2)

---

- **Humans are diploid organisms**

- For each region of the genome we actually have two chromosome copies.



agtActggagcttGgctcac  
agtActggagcttTgctcac



agtCctggagcttTgctcac  
agtCctggagcttGgctcac



agtActggagcttGgctcac  
agtCctggagcttGgctcac



agtCctggagcttTgctcac  
agtActggagcttGgctcac



agtActggagcttTgctcac  
agtActggagcttTgctcac

## SNPs and Haplotyping (2)

- **Humans are diploid organisms**

- For each region of the genome we actually have two chromosome copies.



agt0ctggagctt0gctcac  
agt0ctggagctt1gctcac



agt1ctggagctt1gctcac  
agt1ctggagctt0gctcac



agt0ctggagctt0gctcac  
agt1ctggagctt0gctcac



agt1ctggagctt1gctcac  
agt0ctggagctt0gctcac



agt0ctggagctt1gctcac  
agt0ctggagctt1gctcac

- Each chromosome copy thus induces a **haplotype** (a string over the {0,1} alphabet). Two chromosome copies per person, so two haplotypes per person. (E.g. orange woman: 00 and 10.)

## SNPs and Haplotyping (3)

- Finding haplotypes is not cheap...

- Ideally we would like to identify the two haplotypes of individuals in a population. But this is still expensive to do in the lab.
- It's much cheaper, however, to obtain **genotype** data. A genotype is a conflation of the two corresponding haplotypes and is written as a string over the {0,1,2} alphabet. Problem: information is lost, cannot distinguish between 0/1 and 1/0.



agt<sup>0</sup>ctggagctt<sup>0</sup>gctcac  
agt<sup>0</sup>ctggagctt<sup>1</sup>gctcac

Genotype: 0

2



agt<sup>1</sup>ctggagctt<sup>1</sup>gctcac  
agt<sup>0</sup>ctggagctt<sup>0</sup>gctcac

Genotype: 2

2



agt<sup>1</sup>ctggagctt<sup>1</sup>gctcac  
agt<sup>1</sup>ctggagctt<sup>0</sup>gctcac

Genotype: 1

2



## *SNPs and Haplotyping (4)*

---

- **Inferring plausible haplotype data from genotype data**

- The goal is to infer the haplotypes (i.e.  $\{0,1\}$  strings) of individuals within a population, given only the genotypes (i.e.  $\{0,1,2\}$  strings) of the individuals in the population.
- Genotypes lose information so in general many solutions are possible. What are “good” (i.e. biologically realistic) solutions?
- A critical observation is that, as a consequence of sexual reproduction (i.e. the inheritance of chromosomes from parents) individuals in a population may have common haplotypes.
- The concept of **parsimony** (for which there is some biological evidence) argues that we should choose the **smallest possible set** of haplotypes that can explain the observed variety in the genotypes.
- Leads to a beautiful combinatorial optimisation problem...



## *Parsimony Haplotyping (PH)*

---

- A genotype is modelled as a string over the alphabet  $\{0,1,2\}$ .
- A haplotype is modelled as a string over the alphabet  $\{0,1\}$ .
- Two haplotypes  $h_1, h_2$  **resolve** a genotype  $g$  iff:-
  - At each site where  $g$  has a 0,  $h_1$  and  $h_2$  both have a 0;
  - At each site where  $g$  has a 1,  $h_1$  and  $h_2$  both have a 1;
  - At each site where  $g$  has a 2,  $h_1$  and  $h_2$  are different i.e. 0/1 or 1/0.
- If  $g$  is resolved by  $h_1$  and  $h_2$  we write  $g = h_1 + h_2$ . (In general a genotype may be resolved by many different pairs of haplotypes.)

### **Parsimony Haplotyping (PH)**

Input: A set of  $n$  genotypes  $G$ , each of length  $m$ ;

Output: The smallest possible set of length- $m$  haplotypes  $H$  such that each genotype is resolved by some pair of the haplotypes (in which case we say that  $H$  resolves  $G$ .)



---

## Parsimony Haplotyping (PH)

Input: A set of  $n$  genotypes  $G$ , each of length  $m$ ;

Output: The smallest possible set of length- $m$  haplotypes  $H$  such that each genotype is resolved by some pair of the haplotypes (in which case we say that  $H$  resolves  $G$ .)

---

- Toy example: suppose the input  $G$  is: 122, 201, 022.
- A smallest possible set of haplotypes that resolves  $G$  has size 4, e.g. 101, 001, 110, 010.

$$\begin{array}{r} 101 \\ 110+ \\ \hline 122 \end{array} \quad \begin{array}{r} 001 \\ 101+ \\ \hline 201 \end{array} \quad \begin{array}{r} 010 \\ 001+ \\ \hline 022 \end{array}$$





## *Adding a phylogenetic restriction to the PH model*

---

### Minimum Perfect Phylogeny Haplotyping (MPPH)

Input: A set of  $n$  genotypes  $G$ , each of length  $m$

Output: The smallest possible set of length- $m$  haplotypes  $H$  such that each genotype is resolved by some pair of the haplotypes **and such that the haplotypes  $H$  permit a perfect phylogeny**. Or 'null' if no solutions exist.

---

- A set of haplotypes permits an (undirected) **perfect phylogeny** iff the haplotypes can be placed at the leaves of an unrooted evolutionary tree, where each site mutates at most once.
- Well-known fact: assuming the haplotypes  $H$  are arranged as the rows of a matrix,  $H$  permits a perfect phylogeny iff the following **forbidden submatrix**  $F$  does not appear:

00  
01  
10  
11



## *Bounded instances of PH and MPPH*

---

- Much work has been done on PH, but very little on MPPH. Both problems are, in general, NP-hard. (The problem of determining whether ANY perfect phylogeny solution exists, PPH, is linear-time solvable.) Heavy reliance on IP (Integer Programming) heuristics.
- Inspired by (amongst others) the paper “Islands of tractability for parsimony haplotyping” (Sharan, Halldórsson, Istrail - 2005) we wanted to explore the interface between ‘hard’ and ‘easy’ instances of these problems.
- $PH(j,k)$  is the PH problem where each genotype has at most  $j$  2s per row, and (assuming the input genotypes are given as the rows of a matrix) at most  $k$  2s per column.
- A ‘\*’ denotes no restriction e.g.  $PH(3,*)$  is the problem with no restriction on the number of 2s per column, but at most three 2s per genotype.
- Same definition for MPPH.



## *Before and after our paper*

---

### **Parsimony Haplotyping (PH)**

- PH(4,3) is APX-hard (Sharan, Halldórsson, Istrail – 2005)
- PH(3,\*) is APX-hard (Lancia, Pinotti, Rizzi - 2004.)
- PH(2,\*) is in P (Lancia et al, independently Cilibrasi et al - 2005.)
- PH(j,\*) approximable in polynomial time with approx. ratio  $2^{j-1}$
- PH(3,3) is APX-hard (3D-MATCHING-3)
- PH(\*,1) is in P
- Polynomial- time approximation algorithms for PH(\*,k) with approx. ratio linear in k.
- More “islands of tractability”

### **Minimum Perfect Phylogeny Haplotyping (MPPH)**

- NP-hard in general (Bafna, Gusfield, Hannenhalli, Yooseph - 2004.)
- No approximation algorithms (with performance guarantees) were known
- MPPH(3,3) is APX-hard (V.COVER)
- MPPH(2,\*) is in P, by reduction to PH(2,\*)
- MPPH(\*,1) is in P, by reduction to PH(\*,1)
- Polynomial-time approximation algorithms for MPPH(\*,k) with approx. ratio linear in k
- More “islands of tractability”



## *The importance of cliques...*

---

- For both PH and MPPH, the main open problem left is  $(^*,2)$ .
- Sharan et al showed that  $PH(^*,2)$  is in P for 'clique' instances.
- We found the analogous result for  $MPPH(^*,2)$ . Surprisingly complicated!
- STOP PRESS! We've also found a polynomial-time algorithm for  $PH(^*,2)$  'clique' instances where certain given haplotypes must be in the solution. But we're still checking details...might lead eventually to  $PH(^*,2)$  algorithm
- **Today I will sketch the following:**
  - Polynomial-time algorithm for  $PH(^*,1)$
  - Polynomial-time algorithm for  $MPPH(^*,1)$
  - Polynomial-time algorithm for  $MPPH(^*,2)$  on 'clique' instances.
  - Polynomial-time approximation algorithms for  $PH(^*,k)$  and  $MPPH(^*,k)$



## Compatibility and consistency

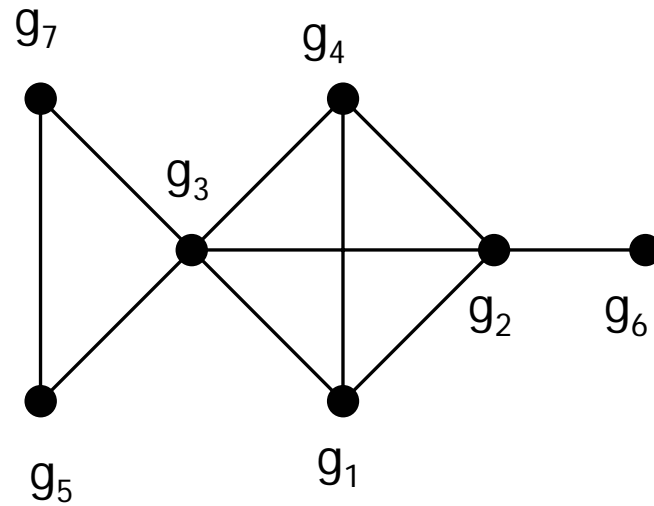
---

- Two genotypes  $g_1$  and  $g_2$  are said to be **compatible** iff at each site where they are non-equal, one of the two genotypes has a 2. For example **020** and **210** are compatible but **020** and **120** are not. (Incompatible genotypes can never share haplotypes.)
- The **Compatibility Graph**  $\text{Comp}(G)$  of a set of genotypes  $G$  has:
  - a vertex for every genotype  $g$  in  $G$ ;
  - an edge between two vertices  $g_1, g_2$  iff  $g_1$  and  $g_2$  are compatible.
- A haplotype  $h$  is said to be **consistent** with  $g$  iff at each site where  $g$  and  $h$  are non-equal,  $g$  has a 2. For example **010** is consistent with **022** but **110** is not. If  $g_1$  and  $g_2$  are compatible and  $h$  is consistent with both, we write  $g_1 \sim_h g_2$ .



$g_1$	0	0	1	0	2	0	1
$g_2$	2	0	2	0	0	0	1
$g_3$	0	0	1	2	0	0	1
$g_4$	0	0	1	0	0	0	2
$g_5$	0	0	1	1	0	2	1
$g_6$	1	2	0	0	0	0	1
$g_7$	0	0	1	1	0	0	1

Example of an input genotype matrix  $G$



Compatibility graph  $\text{Comp}(G)$



## *The $PH(*, 1)$ compatibility graph has a special structure*

---

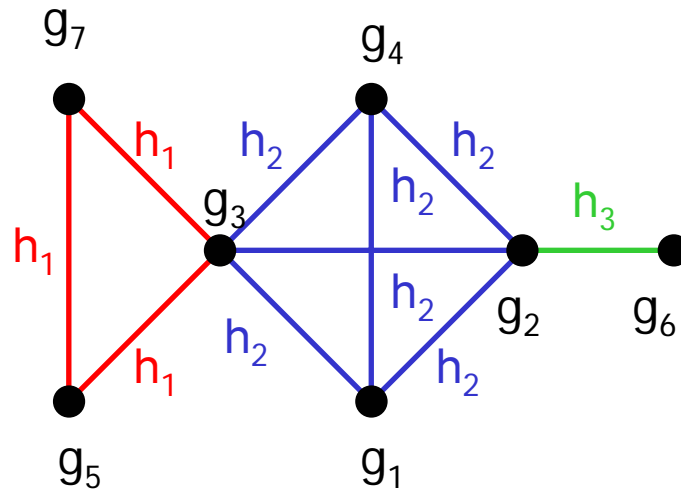
In  $PH(*, 1)$ , the following facts hold:

- (1) If two genotypes  $g_1$  and  $g_2$  are compatible, then there is **precisely one** haplotype  $h$  that is consistent with both of them. (At each column, read off the non-2 element.) So each edge in  $\text{Comp}(G)$  corresponds to a unique haplotype.
- (2) The compatibility graph  $\text{Comp}(G)$  is a 1-sum of cliques, and is thus **chordal**.

Related to (1), the following is also true:

- (3) Given any mutually compatible set of genotypes (which thus appear as a clique  $c$  in the compatibility graph), there is precisely **one** haplotype that is consistent with all of them. (At each column, read off the non-2 element.) We call this the **clique haplotype**  $h_c$  for that clique  $c$ .

$g_1$	0	0	1	0	2	0	1
$g_2$	2	0	2	0	0	0	1
$g_3$	0	0	1	2	0	0	1
$g_4$	0	0	1	0	0	0	2
$g_5$	0	0	1	1	0	2	1
$g_6$	1	2	0	0	0	0	1
$g_7$	0	0	1	1	0	0	1



Clique haplotype for red clique (i.e.  $h_1$ ) is: 0011001  
 Clique haplotype for blue clique (i.e.  $h_2$ ) is: 0010001  
 Clique haplotype for green clique (i.e.  $h_3$ ) is: 1000001






Algorithm idea:

- The graph  $\text{Comp}(G)$  is chordal, and thus has a simplicial vertex. (A vertex whose closed neighbourhood is a clique.) Removing a simplicial vertex (and its incident edges) still leaves a chordal graph.
- We build the solution  $H$  by repeatedly 'peeling' vertices away from  $\text{Comp}(G)$ , each time adding haplotypes to a haplotype set  $H'$  (that is initially empty.) Specifically:
  - At every iteration, locate a simplicial vertex (genotype)  $g$ . Depending on which haplotypes are already in  $H'$ , add (at most) two haplotypes  $h_1$  and  $h_2$  (such that  $g = h_1 + h_2$ ) to  $H'$ , and then remove  $g$  and its incident edges from  $\text{Comp}(G)$ .

Repeat until  $\text{Comp}(G)$  is empty. Return  $H'$  as the final solution  $H$ .

- But, for each  $g$ , how do we decide which  $h_1$  and  $h_2$  to add to  $H'$ ? Tempting to always use the clique haplotypes, but that's not always optimal...



Try each of the following steps in order and stop once a step has been executed. ( $h_c$  is the clique haplotype corresponding to  $g$ , defined at the start of the algorithm.)

1. If  $g$  has no 2s, simply add  $g$  to  $H'$ .
2. If  $g$  is already resolved by some pair of haplotypes in  $H'$ , there's no need to add new haplotypes to  $H'$ .
3. If just adding the clique haplotype of  $g$  to  $H'$  allows  $H'$  to resolve  $g$ , do it.
4. If just adding some non-clique haplotype to  $H'$  allows  $H'$  to resolve  $g$ , do it.
5. If  $g$  is not an isolated vertex, add  $\{h_c, h\}$  to  $H'$  (where  $g = h_c + h$ .)
6. Add any two haplotypes  $h_1, h_2$  to  $H'$  such that  $g = h_1 + h_2$ .

Optimality of algorithm proved by induction. (Not given here.)

*In MPPH(\*, 1) some resolutions are forbidden...*

0	0
0	1
1	0
2	2

Two columns in G

resolve

0	0
0	1
1	0
0	0
1	1

Corresponding columns in H

Eliminate  
duplicates

0	0
0	1
1	0
1	1

= FORBIDDEN  
RESOLUTION

0	0
0	1
1	0
0	1
1	0

Corresponding columns in H

Eliminate  
duplicates

0	0
0	1
1	0

= SAFE  
RESOLUTION

## Reducing $MPPH(*,1)$ to $PH(*,1)$ by discouraging forbidden resolutions

- In  $PH(*,1)$ , there will – for each pair of columns – be at most one row that is 22, and if such a row exists there will be no other 2s in those columns.
- Idea: to reduce  $MPPH(*,1)$  to  $PH(*,1)$ , we have to discourage such rows from resolving the forbidden way.
- We do this by adding, for each pair of columns where a 22 can be seen, a ‘blocking’ column that biases resolutions in favour of the safe way.

0	0	
0	1	
1	0	
2	2	

becomes

0	0	0
0	1	1
1	0	1
2	2	1

Idea is that, within  $PH(*,1)$ , the 22 might still choose the forbidden resolution (e.g. 00/11 in this case) but the haplotypes used to do this cannot be shared by any other genotypes, because of the extra column. So just as good, if not better, to choose the safe resolution. So (assuming feasibility) there exist optimal solutions to  $PH(*,1)$  where all such 22 resolutions are safe.



## *MPPH(\*,2) on clique instances*

---

- Sharan et al showed that  $PH(*,2)$  is in P if the compatibility graph is a clique i.e. if all genotypes are mutually compatible.
- We have proved the same for  $MPPH(*,2)$ .
- Surprisingly complicated! Main complication is that, unlike the  $PH(*,2)$  case, using the all-0 haplotype can sometimes cause forbidden resolutions.
- Here I demonstrate the most important ideas/steps behind the algorithm.



## Foundations for the $MPPH(*,2)$ -clique algorithm

---

- We can assume (by a relabelling argument) that the input matrix is restricted to  $\{0,2\}$ .
- If there are 3 or more genotypes in the input, the only haplotype that is consistent with all genotypes is the all-0 haplotype (because every column must contain a 0.)
- The **restricted compatibility graph**  $\text{ResComp}(G)$  is defined as follows:-
  - A vertex for every genotype  $g$ ;
  - An edge between two genotypes  $g_1$  and  $g_2$  iff **there exists some haplotype  $h$  not equal to the all-0 haplotype, such that  $g_1 \sim_h g_2$ .**  
(Equivalently, iff there exists some column where  $g_1$  and  $g_2$  both have 2s.)
- For an edge  $(g_1, g_2)$  in  $\text{ResComp}(G)$ , define the **edge haplotype** for that edge as the haplotype having 1s in columns where  $g_1$  and  $g_2$  both have a 2, and 0s everywhere else. (These are useful for avoiding the forbidden submatrix!)

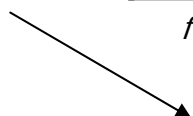


## Critical observations

1. If the input  $G$  does permit solutions, then every vertex in the restricted compatibility graph has degree at most 2 i.e. consists of paths, cycles and isolated vertices.
2. It is never permitted to resolve a degree-2 vertex in the restricted compatibility graph with the all-0 haplotype.
3. If you resolve a genotype  $g$  with haplotypes  $h_1$  and  $h_2$ , and  $h_1$  and  $h_2$  are both consistent with other haplotypes, then  $h_1$  and  $h_2$  are **uniquely defined** (and, for deg-2 vertices, are equal to the adjacent edge haplotypes.)
4. Genotypes that cannot share both their haplotypes, can be thought of as having at least one **private haplotype** that no other genotypes can share.

2	2	2
2	0	0
0	2	0
0	0	2

*forbidden!*



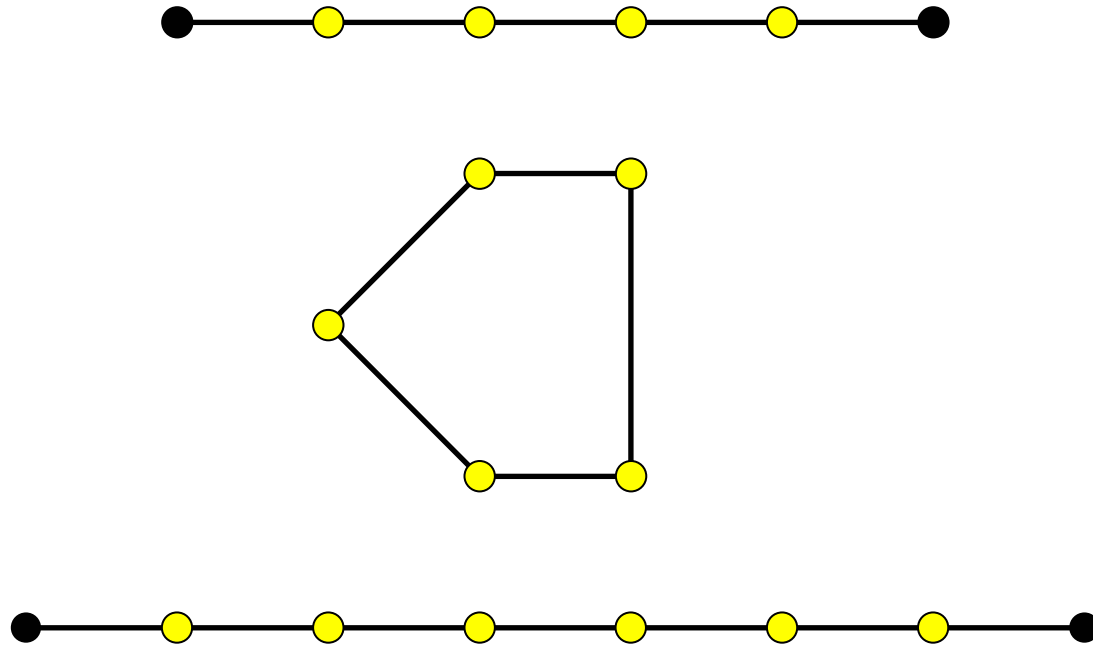
2	2
0	2
2	0

*The 22 here must not be resolved 00/11.*



## *Example of restricted compatibility graph*

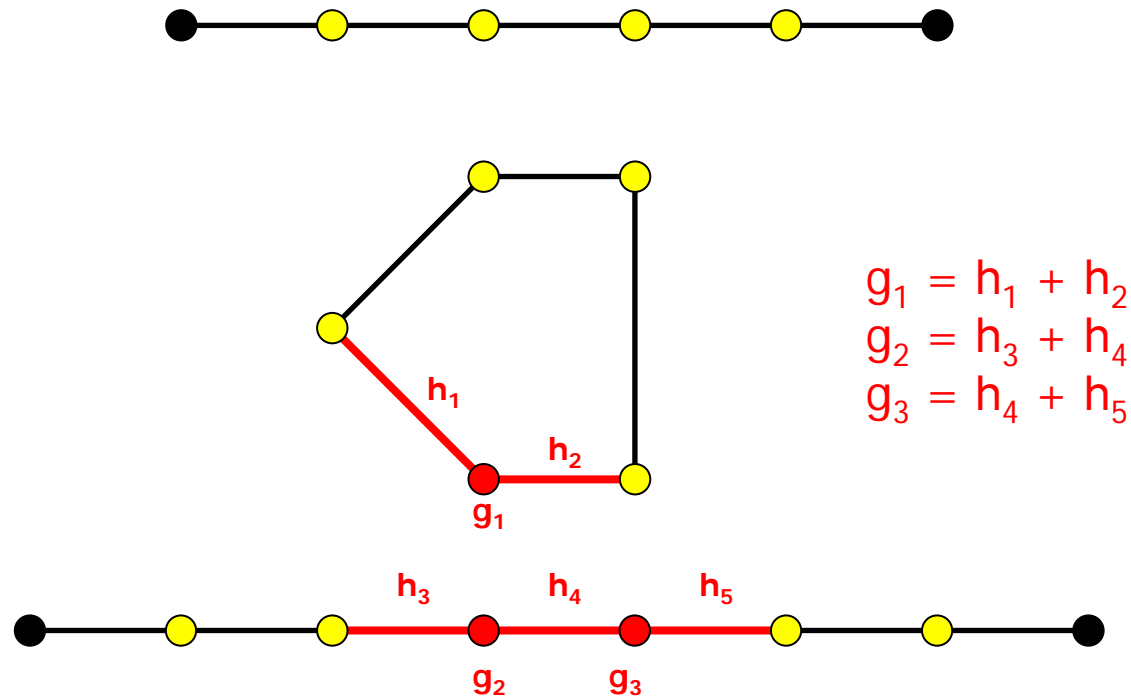
---



Vertices that in the beginning are degree-2 are coloured yellow.  
These can never be resolved using the all-0 haplotype!

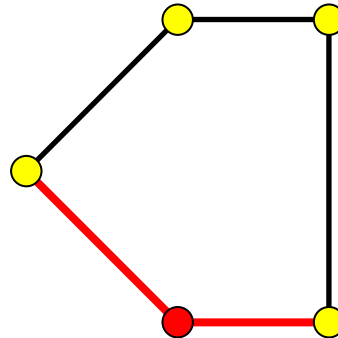


*Step 1: Where degree-2 vertices can be resolved as the sum of their two adjacent edge haplotypes, do that.*

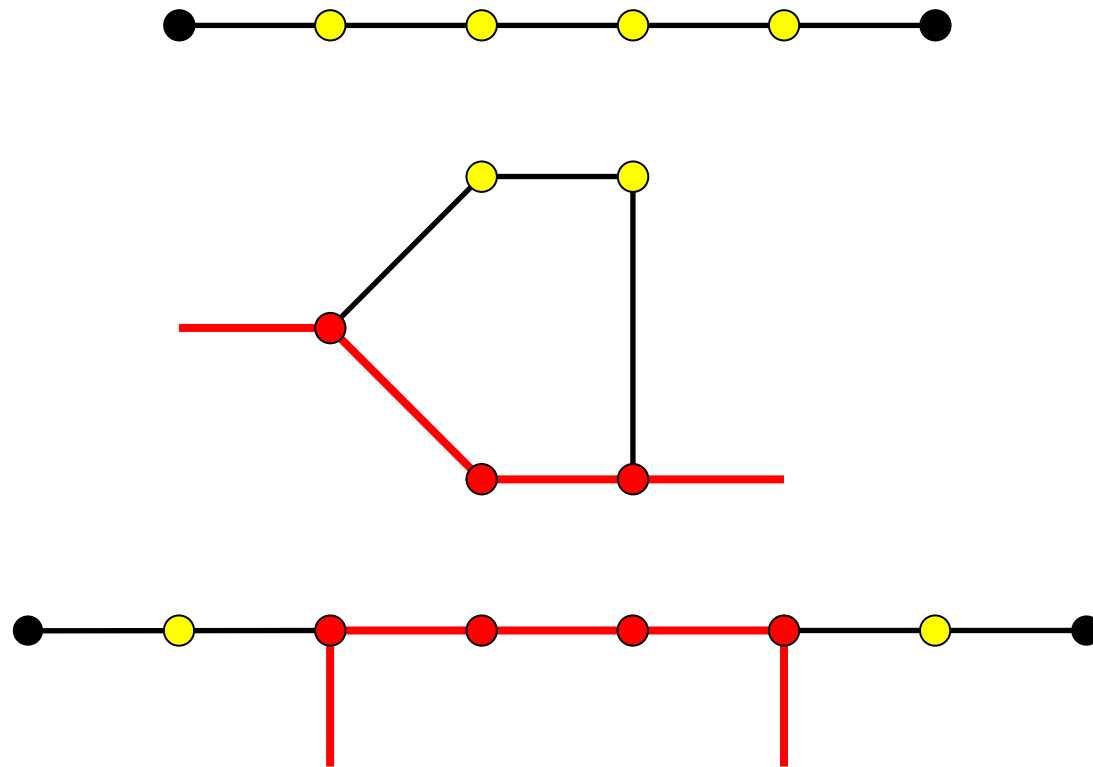


A red edge denotes that the corresponding edge haplotype has been put in the solution. Resolved genotypes are also shown in red.

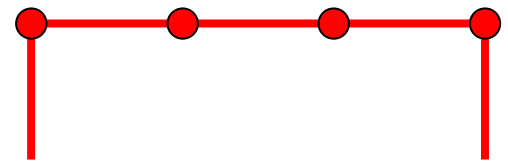
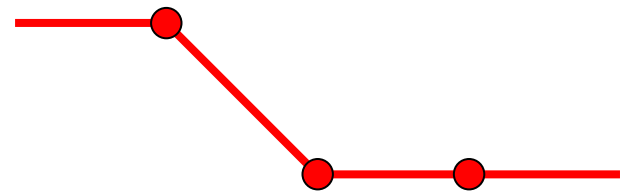
*Step 2: Resolve genotypes that are adjacent to edge haplotypes added in the previous step (i.e. in this case where yellow vertices are next to red edges)*



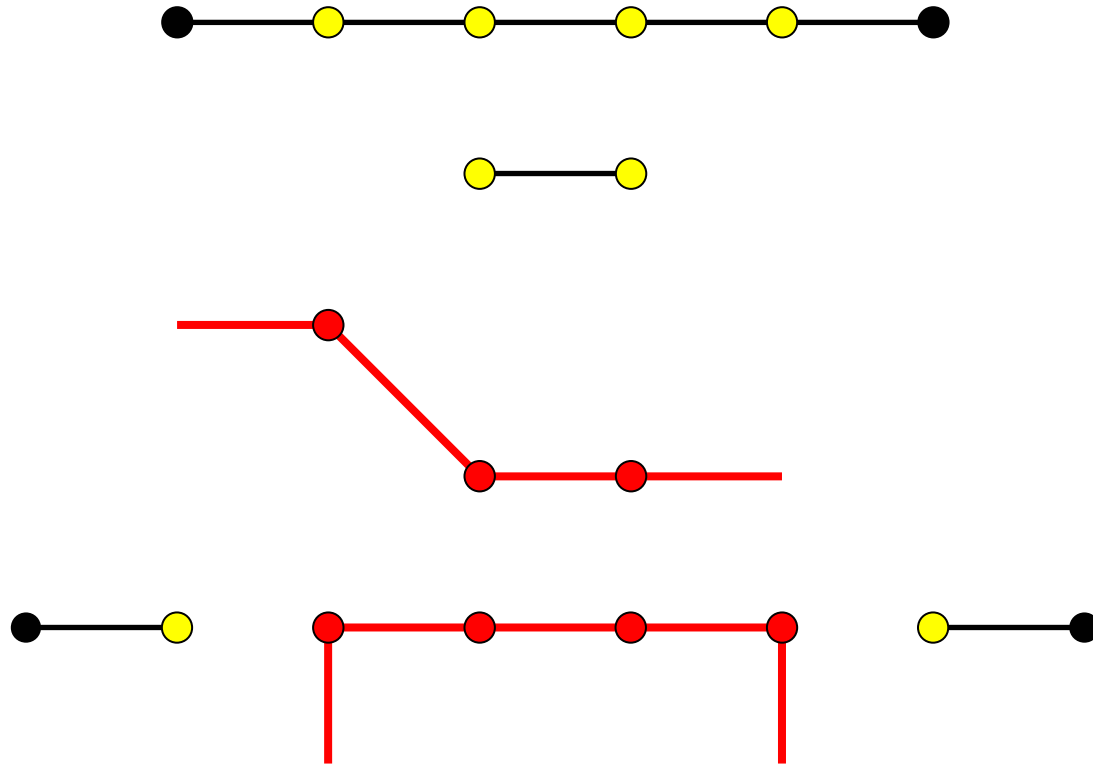
*Step 2: Resolve genotypes that are adjacent to edge haplotypes added in the previous step (i.e. in this case where yellow vertices are next to red edges), done*



The new red edges, representing the newly added haplotypes, will be private haplotypes and thus cannot be shared by any other genotypes (symbolised here by the edge leaving the vertex at an angle.)

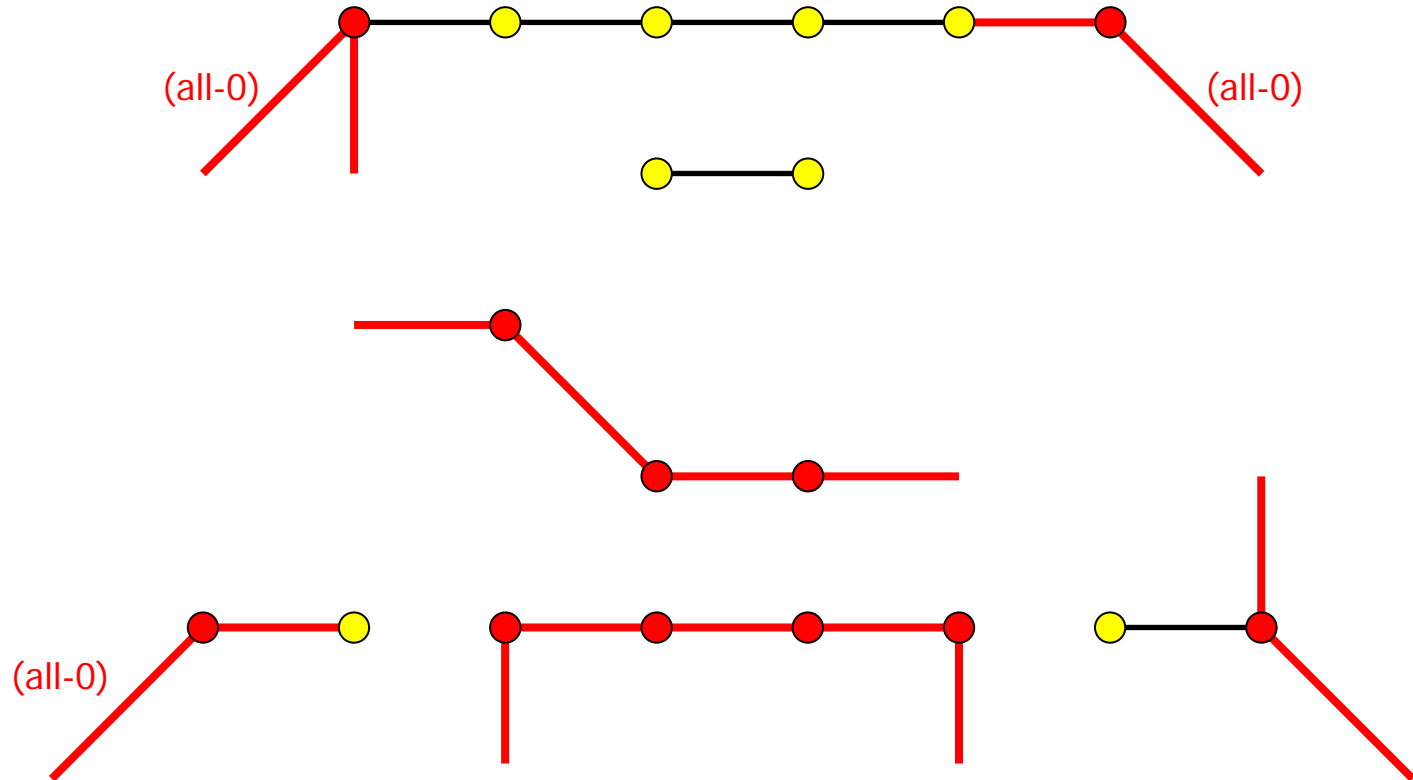


*Step 3: Unless all components are 'bad', or no black genotypes left, add all-0 haplotype and use it to resolve remaining black genotypes*

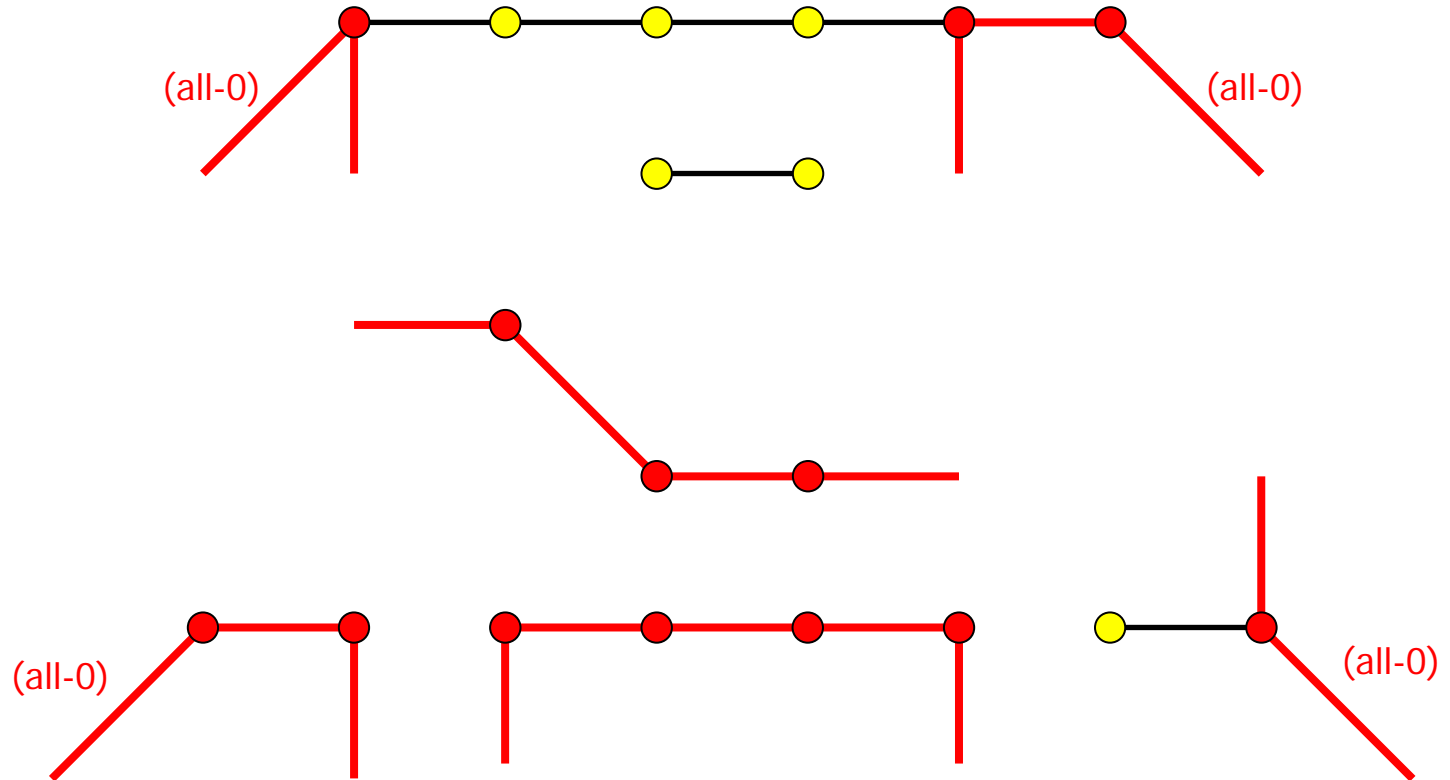




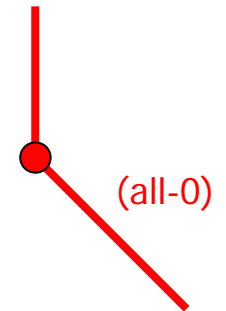
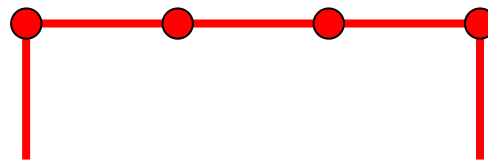
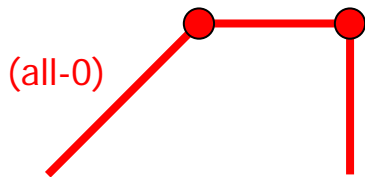
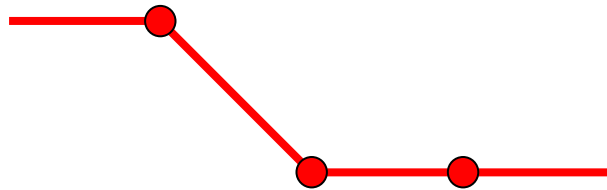
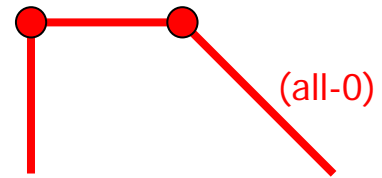
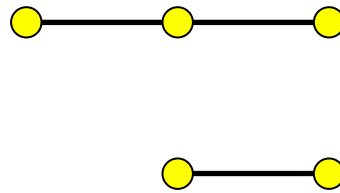
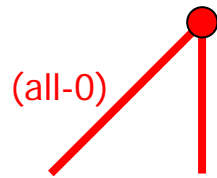
*Step 4: Each genotype that is still unresolved but adjacent to an edge haplotype already in the solution, resolve it using that edge haplotype...*



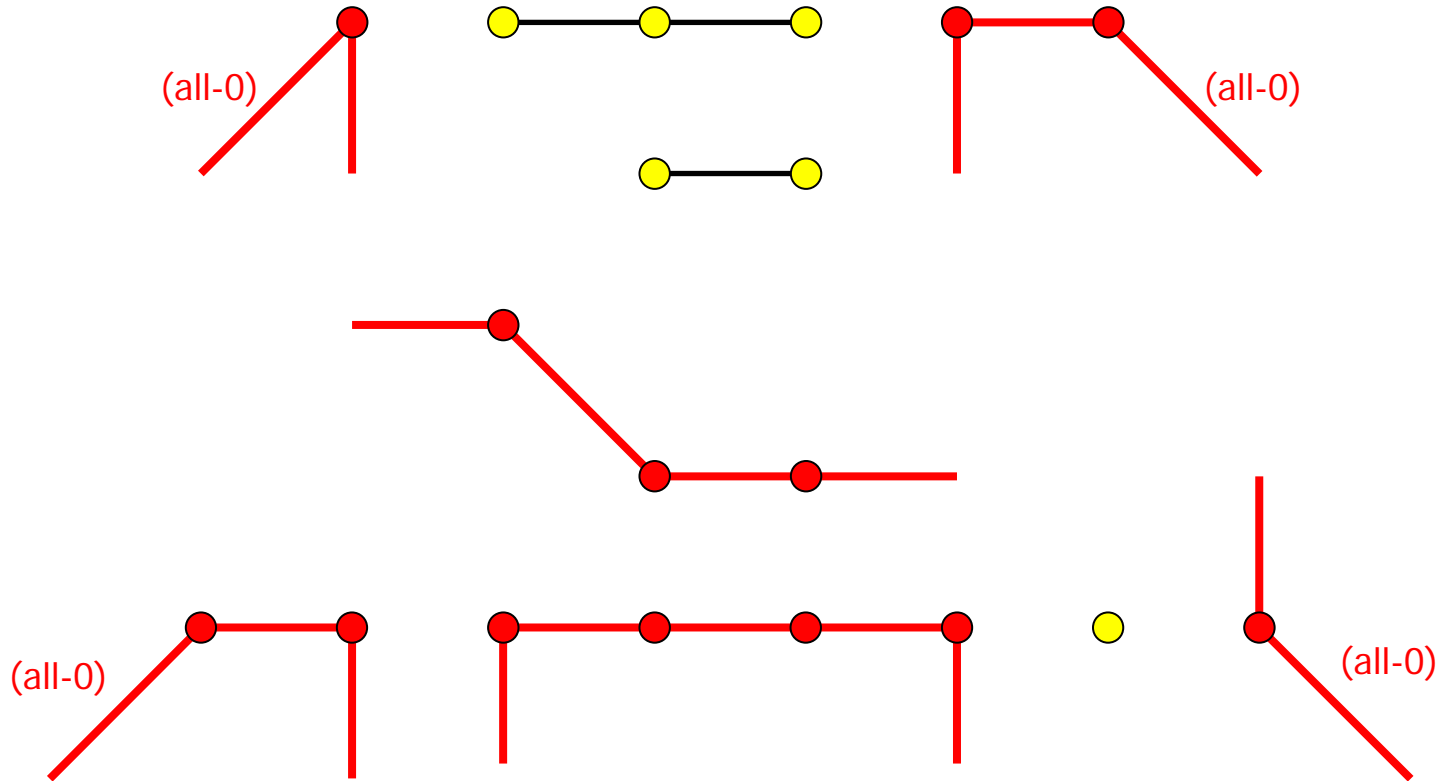
*Step 4: Each genotype that is still unresolved but adjacent to an edge haplotype already in the solution, resolve it using that edge haplotype...done.*



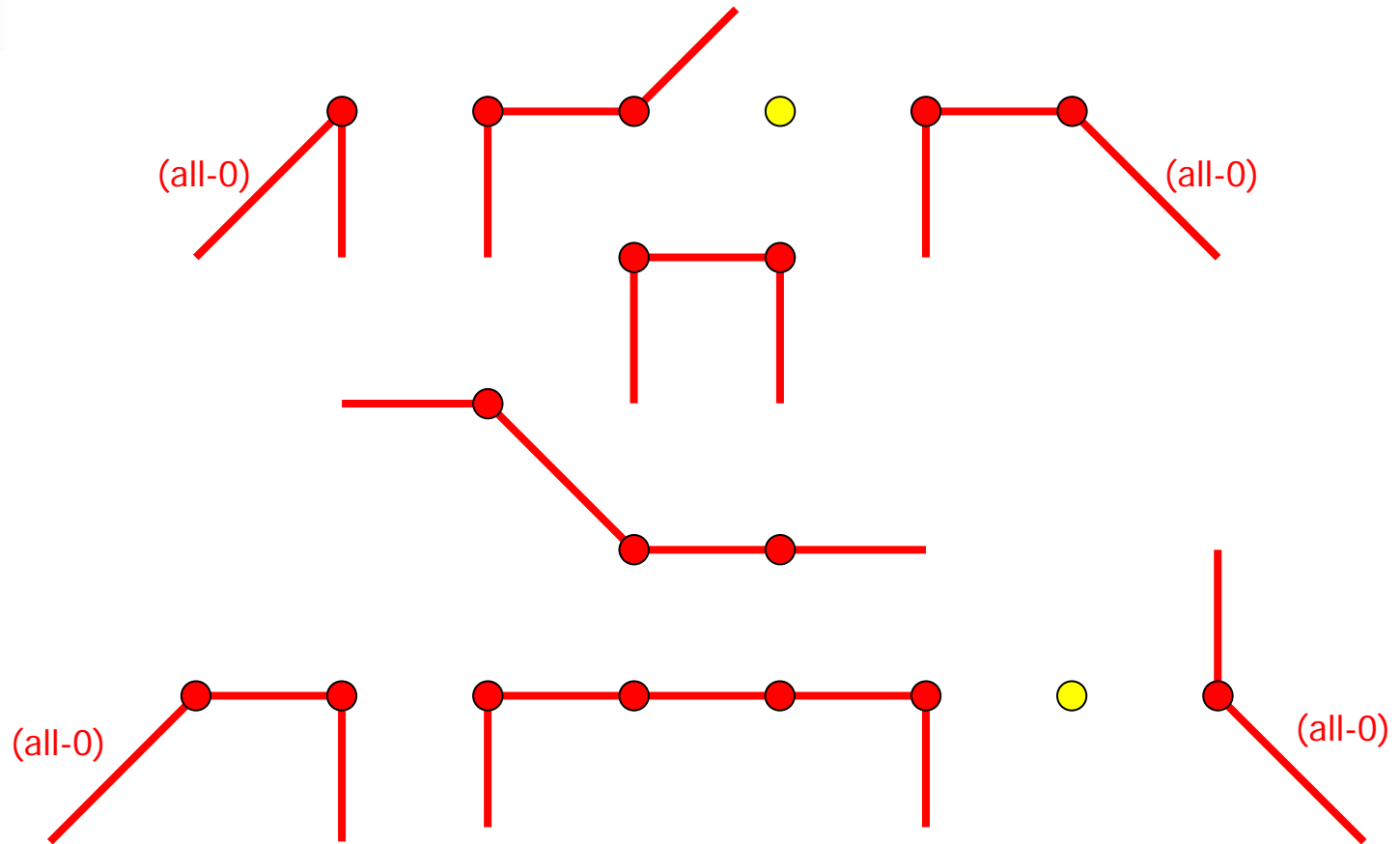




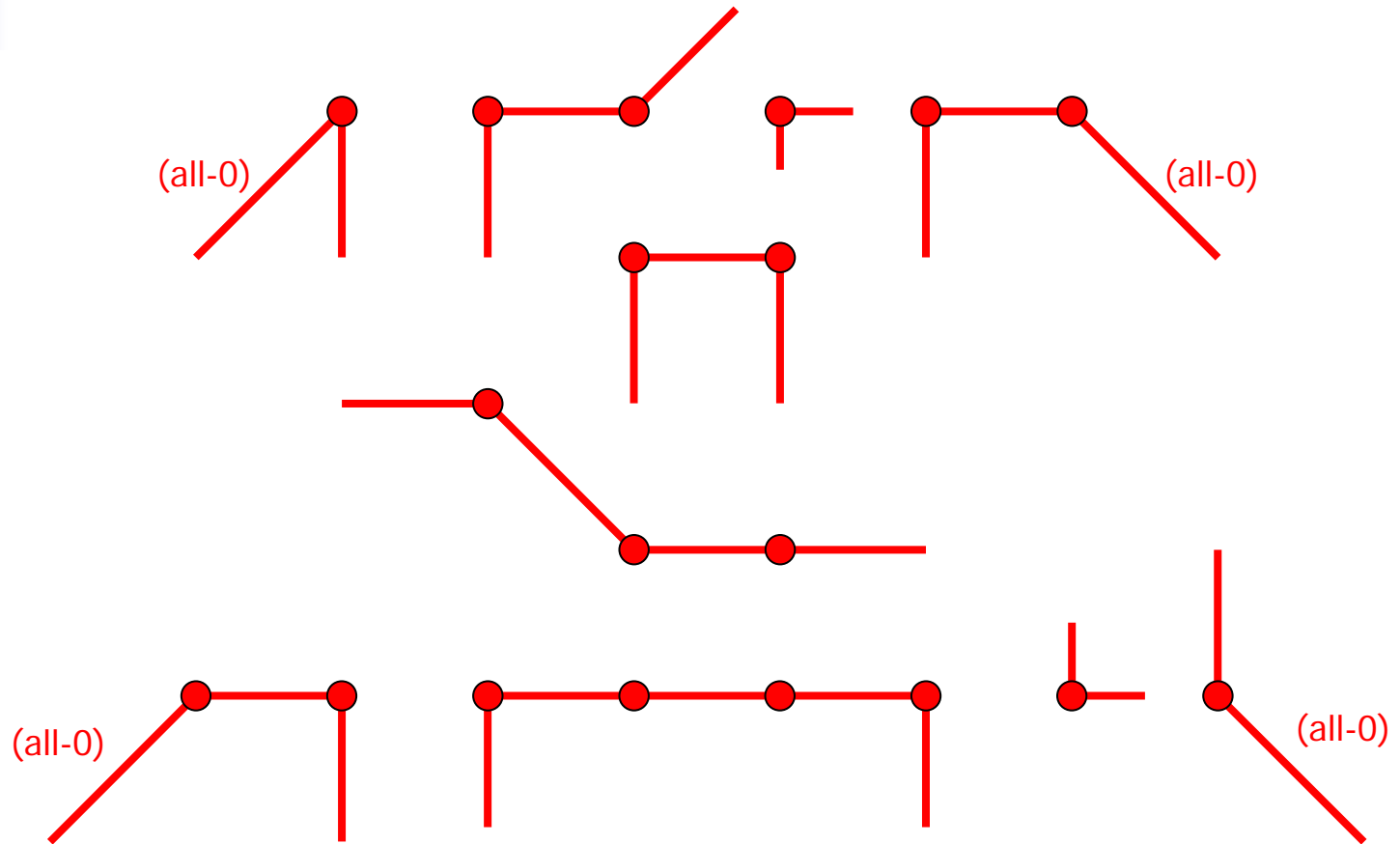
*Step 5: Take a maximum matching on the remaining yellow genotypes and add the corresponding edge haplotypes...*



*Step 5: Take a maximum matching on the remaining yellow genotypes and add the corresponding edge haplotypes...done.*



*Step 6: Finally, resolve each remaining isolated genotype using an arbitrary edge haplotype (plus one other haplotype.)*





# Approximation Algorithms



## *Solving/approximating PH and MPPH*

---

- In practice, Integer Programming (IP) and SDP (Semi-Definite Programming) methods are used to solve PH (and, to a lesser extent, MPPH.) These methods can cope with inputs with up to tens of rows and tens of columns.
- Interestingly enough there are very few methods which give guaranteed approximation ratios.
- Lancia et al have given two separate approximation algorithms (one combinatorial, one based on LP-rounding) which can approximate  $PH(j, *)$  with approximation ratio  $2^{j-1}$ . (LP-rounding method does not actually need  $j$  fixed to guarantee polynomial running time.)
- Yao-Ting Huang, Kun-Mao Chao & Ting Chen (2005) have given an approximation algorithm with approx. ratio  $O(\log n)$  but with running time potentially exponential in  $j$ .
- General problem: approximation ratios and/or running time explode exponentially as a function of the maximum number of 2s per row!



## *Approximation PH(j,k)...eliminating dependency on j*

---

- We have developed bounds (and simple polynomial-time algorithms) which allow us to give approximation ratios for  $PH(*,k)$  that are linear in  $k$ , for both PH and MPPH. Actually, we do not need  $k$  to be a constant.
- These bounds/algorithms are thus an improvement over existing “j”-based ratios for instances where the maximum number of 2s seen in a genotype, is large.
- In our definition of PH and MPPH it is possible that some genotypes in the input do not contain any 2s i.e. are themselves “haplotypes” that have to appear in the solution.
- The versions of PH and MPPH where every genotype contains at least one 2, we call  $PH^h$  and  $MPPH^h$
- We make this distinction because we can approximate  $PH^h$  better than PH (and likewise  $MPPH^h$  better than MPPH.)



## *Our approximation ratios*

<b>Problem</b>	<b>Approx. ratio as function of k</b>	<b>Approx. ratio for k=2, 3, 4...</b>
PH(*,k) <sup>-h</sup>	$\frac{3k}{4} + \frac{7}{4} - \left(\frac{3}{2}\right)\left(\frac{1}{k+1}\right)$	2.75, 3.625, 4.45, ...
PH(*,k)	$\frac{3k}{2} + \frac{1}{2}$	3.5, 5, 6.5, ...
MPPH(*,k) <sup>-h</sup>	$\frac{(k+1)(k+2) - 2}{k+1}$	3.33, 4.5, 5.6, ...
MPPH(*,k)	$2k$	4, 6, 8, ...

Today I'll show how the PH(\*,k)<sup>-h</sup> ratio is achieved.





## *Two existing lower bounds*

---

- Well known crude lower bound: For any PH or MPPH instance with  $n$  genotypes, at least the following number of haplotypes are needed:

$$LB_{sqr}t(n) = \left\lceil \frac{1 + \sqrt{1 + 8n}}{2} \right\rceil \approx \sqrt{2n}$$

- Sharan et al showed that, for any **clique** instance of  $PH(*,k)^h$  on  $n$  genotypes, the following is a tight lower bound:

$$LB_{sha}(n, k) = \left\lceil \frac{2n}{k+1} + 1 \right\rceil$$



## *A weaker, but more general bound for the clique*

- For clarity, let's focus on the case  $k=2$ . The  $LB_{sha}$  clique bound gives in this case

$$LB_{sha}(n,2) = \left\lceil \frac{2n}{3} + 1 \right\rceil \approx 0.66n$$

- By combining  $LB_{sqr}$ ,  $LB_{sha}$  and some case analysis we developed a new bound for clique instances of  $PH(*,k)^h$ . In the case  $k=2$  our bound gives:

$$LB_{our}(n,2) = \left\lceil \frac{3n+6}{5} \right\rceil \approx 0.6n$$

- Our bound is asymptotically weaker. But it has a critical advantage: the **additive term is larger**. This will help us generalise the bound to non-clique instances!



- We can prove by induction on the compatibility graph that our clique bound holds for **all**  $\text{PH}(*,2)^h$  instances, not just cliques. We use cliques as the induction basis. Two critical observations:

Observation 1: If the compatibility graph of an  $\text{PH}(*,2)^h$  instance is not a clique, then it can be **disconnected into at least two non-empty components by removing at most 2 genotypes.**

Proof: If the compatibility graph is not a clique, then some two genotypes are non-compatible i.e. in some column **one genotype has a 0, the other has a 1...**

2
0
2
0
1
0

...there will be at most two 2s in that column. If the **genotypes with 2s** in that column are removed, there is no path from the "1" genotype to any of the "0" genotypes in the resulting compatibility graph: disconnection!



## *Splitting the compatibility graph into smaller pieces*

Observation 2: Suppose I remove two genotypes to disconnect the compatibility graph. Summing lower bounds for the (at least) two resulting components gives a lower bound for the original compatibility graph.

- We use Observation 2 to build the inductive lemma. Suppose we begin with  $n$  genotypes, we remove 2 genotypes and we are left with two components with  $n_1$  and  $n_2$  genotypes respectively i.e.  $n = n_1 + n_2 + 2$ .
- To complete the induction we need to show that we need at least  $(3n+6)/5 = (3(n_1+n_2+2) + 6)/5 = (3(n_1+n_2) + 12)/5$  haplotypes.
- By induction the first component needs  $(3n_1 + 6)/5$  haplotypes, the second component needs  $(3n_2 + 6)/5$ , summing them together gives  $(3(n_1+n_2)+12)/5$  – we are finished!
- **So our bound holds not only for cliques, but for all instances.**



*The lower bound alone gives us a constant-factor approximation ratio*

---


- So, for  $\text{PH}(*,2)^h$  instances we know that at least  $(3n+6)/5$  haplotypes are needed. This bound also holds for  $\text{MPPH}(*,2)^h$  by the same analysis.
- But, given an instance of PH or MPPH with  $n$  genotypes, it is easy to see that even a naïve solution **never needs more than  $2n$  haplotypes**.
- So simply returning **any** feasible solution to  $\text{PH}(*,2)^h$  or  $\text{MPPH}(*,2)^h$  guarantees an approximation ratio of  $2n / ((3n+6)/5) < 3.333\dots$
- Large improvements in approximation ratios are almost certainly possible by developing 'real' approximation algorithms...
- We haven't yet done this for MPPH but we have built a very simple 'real' approximation algorithm for  $\text{PH}(*,k)^h$  and  $\text{PH}(*,k)$ ...

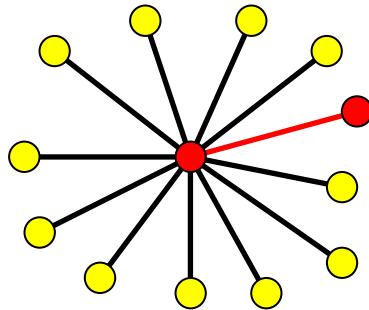


## *Strengthening the approximation ratio with a “matching” algorithm*

---

- We can strengthen the ratio for  $\text{PH}(*, 2)^h$  from 3.33... to 2.75 as follows.
- Observe that, if the compatibility graph has an independent set of size  $N$ , then  $2N$  is also a valid lower bound. (In other words: no haplotype sharing is possible between those  $N$  genotypes, so arbitrarily use 2 haplotypes per genotype.)
- **Algorithm:** Take **any maximum matching  $M$**  in the compatibility graph. Each matching edge involves two genotypes; these two genotypes can be resolved by 3 haplotypes (because they can share one haplotype.) All remaining genotypes can be resolved arbitrarily by 2 haplotypes each. Note that because the matching is maximal, these  $n - 2|M|$  remaining genotypes must form an independent set i.e. give a lower bound of  $2(n - 2|M|)$ .
- The 2.75 approximation algorithm is obtained by comparing our output to the two lower bounds  $(3n+6)/5$  and  $2(n - 2|M|)$ , and taking the better of the two.

- 
- Comparing with the “independent set” lower bound is useful when maximum matchings in the compatibility graph are very small e.g. star-like graphs:



- Here  $n=13$ , so the standard lower bound gives  $(3n+6)/5 = 9$  haplotypes.
- A maximum matching has only one edge, so the algorithm uses in total 3 haplotypes for the 2 covered genotypes, and then 2 each for the remainder i.e.  $3 + 2(13 - 2) = 25$ . Comparing to the  $(3n+6)/5$  bound gives an approx. ratio of  $25/9 = 2.777... > 2.75$ .
- But the yellow genotypes form an independent set of size 11, giving a lower bound of 22, so we get an approximation ratio of  $25/22 < 2.75$ .



## *Some open problems...lots to do!*

---

- Complexity of  $PH(*, 2)$ ?
- Complexity of  $MPPH(*, 2)$ ?
- Complexity of  $PH(3, 2)$ ?
- Complexity of  $MPPH(3, 2)$ ?
- Is  $MPPH(j, k)$  always of the same complexity as  $PH(j, k)$ ?
- In how far is  $PH$  reducible to  $MPPH$ , and vice-versa?
- Approximation algorithms for  $MPPH(j, *)$  (they don't yet exist)
- Adapting the simple "matching" approximation algorithm to  $MPPH$
- Approximation algorithms for  $PH(j, k)$  and  $MPPH(j, k)$  that are a non-trivial function of both  $j$  and  $k$ . Better ratios needed.
- How close are the solutions provided by the various  $PPH$  feasibility algorithms (i.e. "produce ANY solution") to optimal  $MPPH$  solutions?
- How feasible is it to simply search through the (implicit) set of all  $PPH$  solutions?
- Using more biologically interesting restraints (e.g. galled networks, SNP block partitioning models).





## *Some open problems...lots to do!*

---

- Complexity of  $\text{PH}(*, 2)$ ?
- Complexity of  $\text{MPPH}(*, 2)$ ?
- Complexity of  $\text{PH}(3, 2)$ ?
- Complexity of  $\text{MPPH}(3, 2)$ ?
- Is  $\text{MPPH}(j, k)$  always of the same complexity as  $\text{PH}(j, k)$ ?
- In how far is PH reducible to MPPH, and vice-versa?
- Approximation algorithms for  $\text{MPPH}(j, *)$  (they don't yet exist)
- Adapting the simple "matching" approximation algorithm to MPPH
- Approximation algorithms for  $\text{PH}(j, k)$  and  $\text{MPPH}(j, k)$  that are a non-trivial function of both  $j$  and  $k$ . Better ratios needed.
- How close are the solutions provided by the various PPH feasibility algorithms (i.e. "produce ANY solution") to optimal MPPH solutions?
- How feasible is it to simply search through the (implicit) set of all PPH solutions?
- Using more biologically interesting restraints (e.g. galled networks, SNP block partitioning models).

Thanks for listening!