# Combinatorial Optimization in Computational Biology: three topics that use Perfect Phylogeny

Dan Gusfield

OSB 2008, November 1, 2008

# Outline

- Haplotyping by Perfect Phylogeny, using Graph Realization.
- Multi-state Perfect Phylogeny Problems, using Integer Programming.
- Phylogenetic Networks, using graph theory.

# SNP Data

- A SNP is a Single Nucleotide Polymorphism - a site in the genome where two different nucleotides appear with sufficient frequency in the population (say each with 5% frequency or more).
- Human SNP maps have been compiled with a density of about 1 site per 1000. HapMap.
- SNP data is what is mostly collected in populations - it is much cheaper to collect than full sequence data, and focuses on variation in the population, which is what is of interest.

# Topic I: Perfect Phylogeny Haplotyping via Graph Realization

# Genotypes and Haplotypes

Each individual has two "copies" of each chromosome.

At each site, each chromosome has one of two alleles (states) denoted by 0 and 1 (motivated by SNPs)

0 1 1 1 0 0 1 1 0

1 1 0 1 0 0 1 0 0

Two haplotypes per individual

Merge the haplotypes

2 1 2 1 0 0 1 2 0 Genotype for the individual

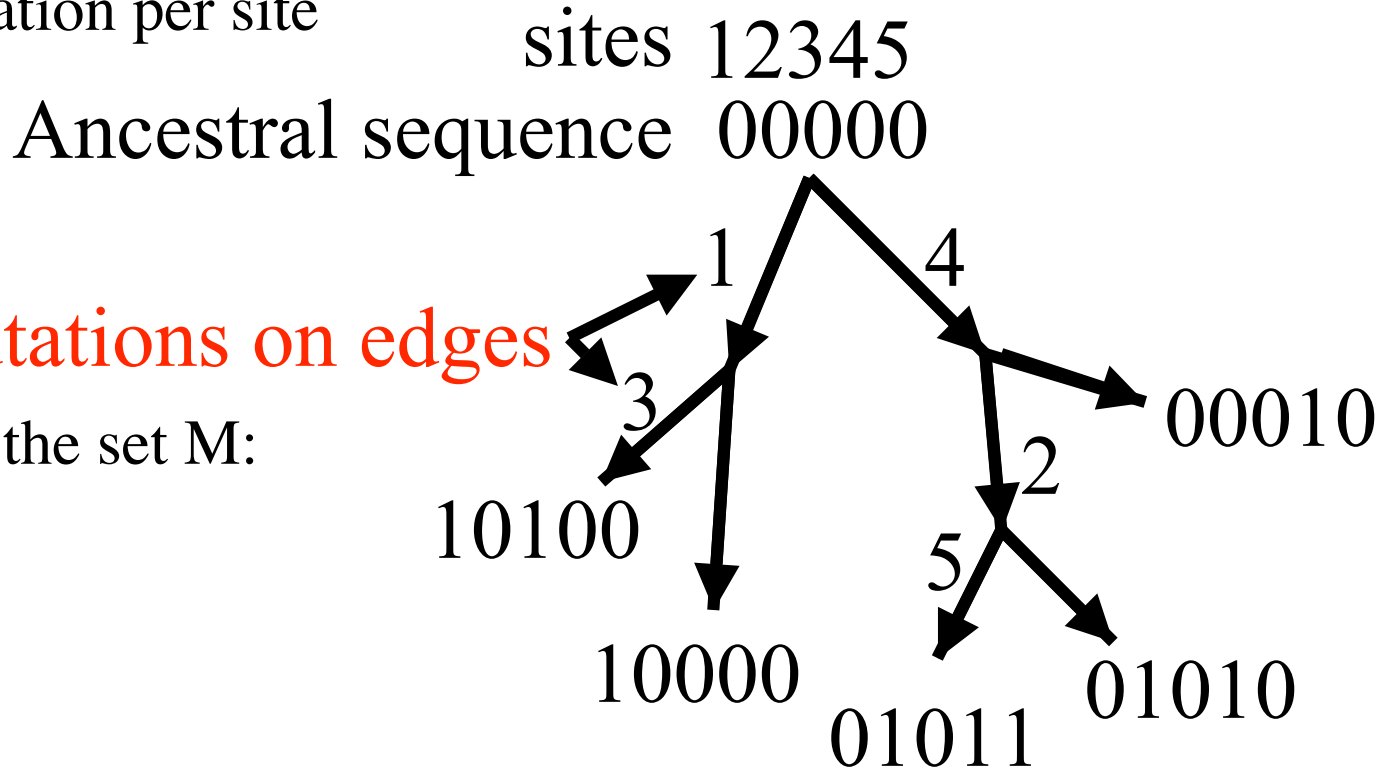
# Haplotyping Problem

- Biological Problem: For disease association studies, haplotype data is more valuable than genotype data, but haplotype data is hard to collect. Genotype data is easy to collect.

- Computational Problem: Given a set of n genotypes, determine the original set of n haplotype pairs that generated the n genotypes. This is hopeless without a genetic model.

# Perfect Phylogeny Model for SNPs: A genetic model

Only one mutation per site allowed.




Site mutations on edges

The tree derives the set M:

10100
10000
01011
01010
00010

Extant sequences at the leaves

# When can a set of sequences be derived on a perfect phylogeny?

Classic NASC: Arrange the sequences in a matrix. Then (with no duplicate columns), the sequences can be generated on a unique perfect phylogeny if and only if no two columns (sites) contain all four pairs:

0,0 and 0,1 and 1,0 and 1,1

This is the 4-Gamete Test

So, in the case of binary characters, if each pair of columns allows a tree, then the entire set of columns allows a tree.

For M of dimension n by m, the existence of a perfect phylogeny for M can be tested in O(nm) time and a
tree built in that time, if there is one. Gusfield, Networks 91

# The Perfect Phylogeny Model

We assume that the evolution of extant haplotypes evolved along a perfect phylogeny with all-0 root.

Justification: Haplotype Blocks, rare recombination, base problem whose solution to be modified to incorporate more biological complexity.

# Perfect Phylogeny Haplotype (PPH)

Given a set of genotypes S, find an explaining set of haplotypes that fits a perfect phylogeny.

sites

S

| | 1 | 2 |
|---|---|---|
| a | 2 | 2 |
| b | 0 | 2 |
| c | 1 | 0 |

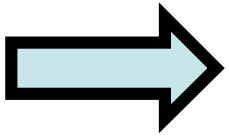
Genotype matrix

A haplotype pair explains a genotype if the merge of the haplotypes creates the genotype. Example: The merge of 0 1 and 1 0 explains 2 2.

# The PPH Problem

Given a set of genotypes, find an explaining set of haplotypes that fits a perfect phylogeny

| | 1 | 2 |
|---|---|---|
| a | 2 | 2 |
| b | 0 | 2 |
| c | 1 | 0 |

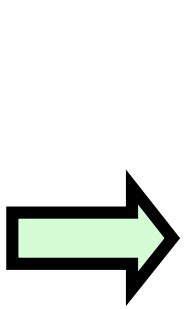


| | 1 | 2 |
|---|---|---|
| a | 1 | 0 |
| a | 0 | 1 |
| b | 0 | 0 |
| b | 0 | 1 |
| c | 1 | 0 |
| c | 1 | 0 |

# The Haplotype Phylogeny Problem

Given a set of genotypes, find an explaining set of haplotypes that fits a perfect phylogeny

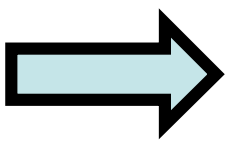
| | 1 | 2 |
|---|---|---|
| a | 2 | 2 |
| b | 0 | 2 |
| c | 1 | 0 |



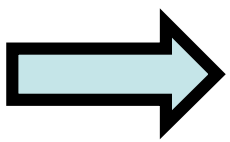
| | 1 | 2 |
|---|---|---|
| a | 1 | 0 |
| a | 0 | 1 |
| b | 0 | 0 |
| b | 0 | 1 |
| c | 1 | 0 |
| c | 1 | 0 |

# The Alternative Explanation

| | 1 | 2 |
|---|---|---|
| a | 2 | 2 |
| b | 0 | 2 |
| c | 1 | 0 |



| | 1 | 2 |
|---|---|---|
| a | 1 | 1 |
| a | 0 | 0 |
| b | 0 | 0 |
| b | 0 | 1 |
| c | 1 | 0 |
| c | 1 | 0 |



No tree possible for this explanation

# Efficient Solutions to the PPH problem - n genotypes, m sites

- Reduction to a graph realization problem (GPPH) - build on Bixby-Wagner or Fushishige solution to graph realization O(nm alpha(nm)) time. Gusfield, Recomb 02
- Reduction to graph realization - build on Tutte's graph realization method O(nm^2) time. Chung, Gusfield 03
- Direct, from scratch combinatorial approach -O(nm^2) Bafna, Gusfield et al JCB 03
- Berkeley (EHK) approach - specialize the Tutte solution to the PPH problem - O(nm^2) time.
- Linear-time solutions - Recomb 2005, Ding, Filkov, Gusfield and a different linear time solution.

# The Reduction Approach

This is the original polynomial time method. Conceptually simplest at a high level (but not at the implementation level) and most extendable to other problems; nearly linear-time but not linear-time.

# The case of the 1's

1) For any row i in S, the set of 1 entries in row i specify the exact set of mutations on the path from the root to the least common ancestor of the two leaves labeled i, in every perfect phylogeny for S.
2) The order of those 1 entries on the path is also the same in every perfect phylogeny for S, and is easy to determine by "leaf counting".

# Leaf Counting

In any column c, count two for each 1, and count one for each 2. The total is the number of leaves below mutation c, in every perfect phylogeny for S. So if we know the set of mutations on a path from the root, we know their order as well.

S

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| a | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| b | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| c | 1 | 2 | 0 | 0 | 2 | 0 | 2 |
| d | 2 | 2 | 0 | 0 | 0 | 2 | 0 |
| Count | 5 | 4 | 2 | 2 | 1 | 1 | 1 |

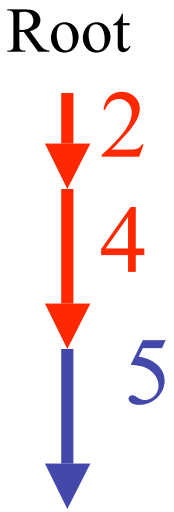
# Simple Conclusions

sites

1 2 3 4 5 6 7

i:0 1 0 1 2 2 2

Subtree for row i data




The order is known for the red mutations together with the leftmost blue(?) mutation.

But what to do with the remaining blue entries (2's) in a row?

# More Simple Tools

3) For any row i in S, and any column c, if S(i,c) is 2, then in every perfect phylogeny for S, the path between the two leaves labeled i, must contain the edge with mutation c.

Further, every mutation c on the path between the two i leaves must be from such a column c.

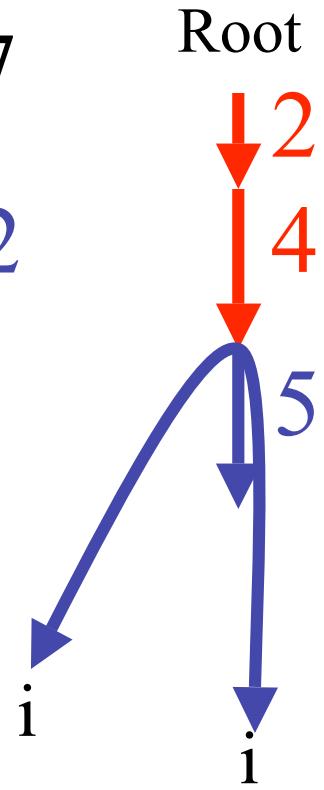
# From Row Data to Tree Constraints

sites

1 2 3 4 5 6 7

i:0 1 0 1 2 2 2

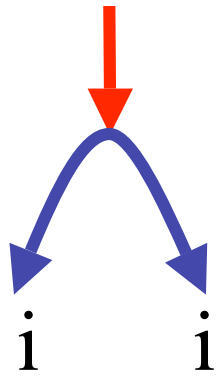
Subtree for row i data




Edges 5, 6 and 7 must be on the blue path, and 5 is already known to follow 4, but we don't where to put 6 and 7.

# The Graph Theoretic Problem

Given a genotype matrix S with n sites, and a red-blue subgraph for each row i,




create a directed tree T where each integer from 1 to n labels exactly one edge, so that each subgraph is contained in T.

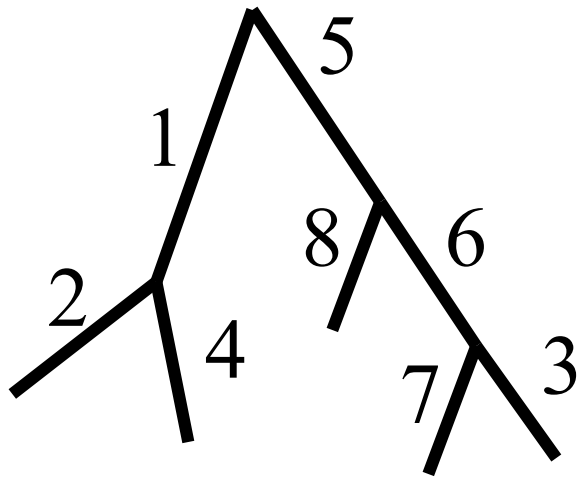
# Powerful Tool: Tree and Graph Realization

- Let Rn be the integers 1 to n, and let P be an unordered subset of Rn. P is called a path set.
- A tree T with n edges, where each is labeled with a unique integer of Rn, realizes P if there is a contiguous path in T labeled with the integers of P and no others.
- Given a family P1, P2, P3…Pk of path sets, tree T realizes the family if it realizes each Pi.
- The graph realization problem generalizes the consecutive ones problem, where T is a path.

- More generally, each set specifies a fundamental cycle in the unknown graph.

# Tree Realization Example

P1: 1, 5, 8
P2: 2, 4
P3: 1, 2, 5, 6
P4: 3, 6, 8
P5: 1, 5, 6, 7




Realizing Tree T

More generally, think of each path set as specifying a fundamental cycle containing the edges in the specified path.

# Graph Realization

Polynomial time (almost linear-time) algorithms exist for the graph realization problem, given the family of fundamental cycles the unknown graph should contain – Whitney, Tutte, Cunningham, Edmonds, Bixby, Wagner, Gavril, Tamari, Fushishige, Lofgren 1930's - 1980's

Most of the literature on this problem is in the context of determining if a binary matroid is graphic.

The algorithms are not simple.

# Reducing PPH to graph realization

We solve any instance of the PPH problem by creating appropriate path sets, so that a solution to the resulting graph realization problem leads to a solution to the PPH problem instance.

The key issue: How to encode the needed subgraph

for each row, and glue them together at the root.

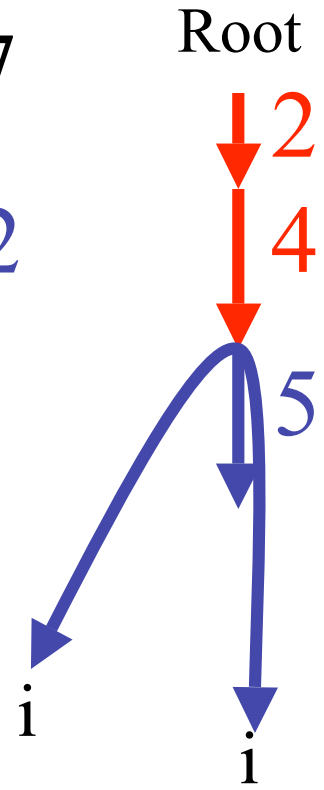
# From Row Data to Tree Constraints

sites

1 2 3 4 5 6 7

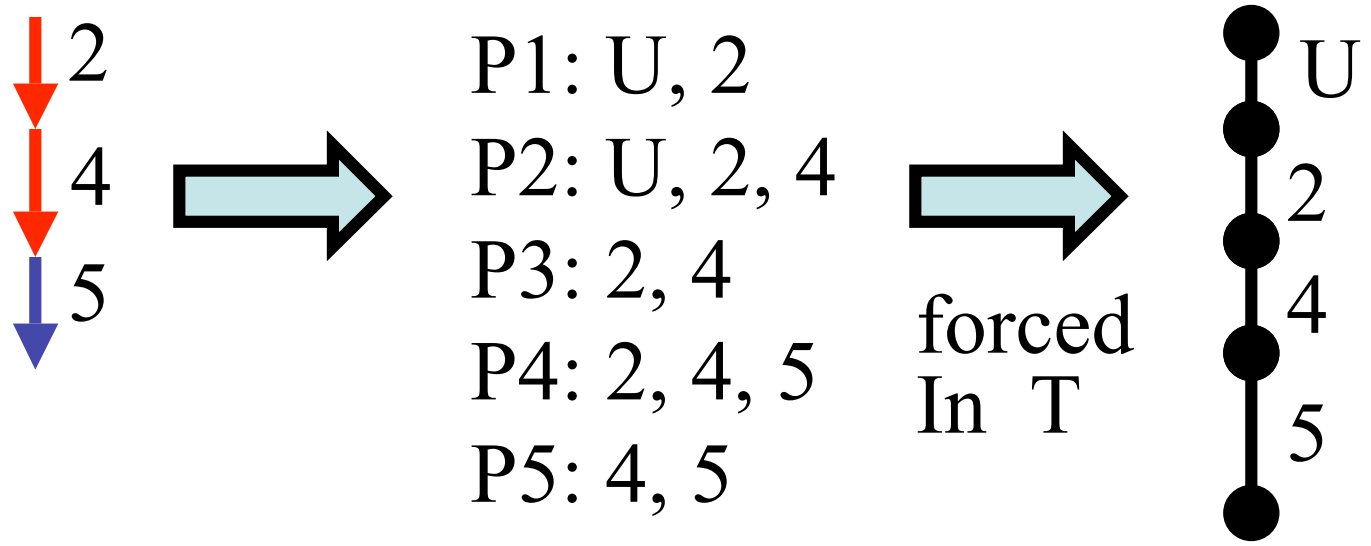
i:0 1 0 1 2 2 2

Subtree for row i data




Edges 5, 6 and 7 must be on the blue path, and 5 is already known to follow 4.

# Encoding a Red-Blue directed path



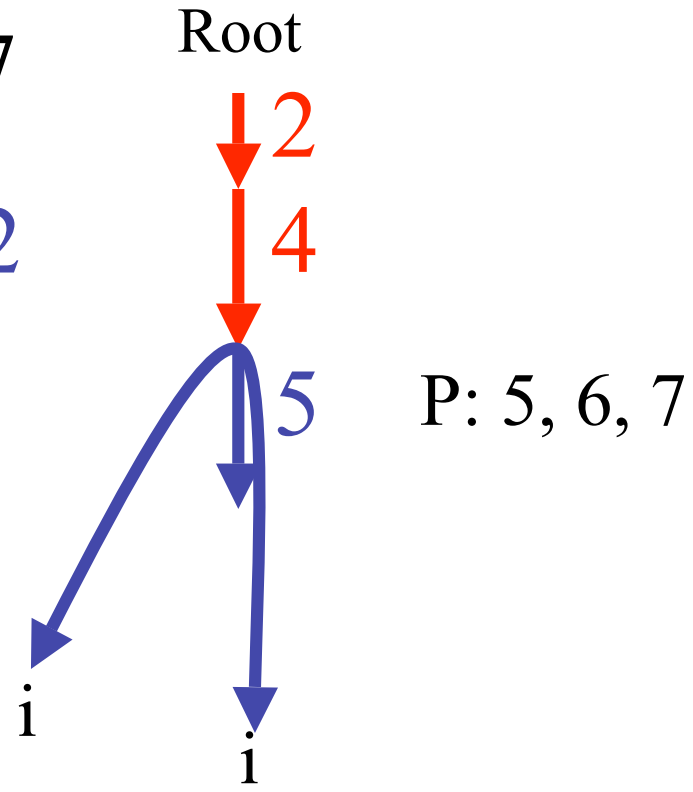

U is a glue edge used to glue together the directed paths from the different rows.

# Now add a path set for the blues in row i.

sites

1 2 3 4 5 6 7

i:0 1 0 1 2 2 2




P: 5, 6, 7

# That's the Reduction

The resulting path-sets encode everything that is known about row i in the input.

The family of path-sets are input to the graph-realization problem, and every solution to the that graph-realization problem specifies a solution to the PPH problem, and conversely.

Whitney (1933?) characterized the set of all solutions to graph realization (based on the three-connected components of a graph) and Tarjan et al showed how to find these in linear time.

# Topic II: Integer Programming for NP-hard Phylogenetic (and Population- Genetic) Problems

# Phylogeny problems often have data with

- Missing entries
- Homoplasy
- Genotype (conflated) sequences, rather than simpler haplotype sequences

Most of these problems are NP-hard, although some elegant poly-time solutions exist (and are well-known) for special cases.

# Question

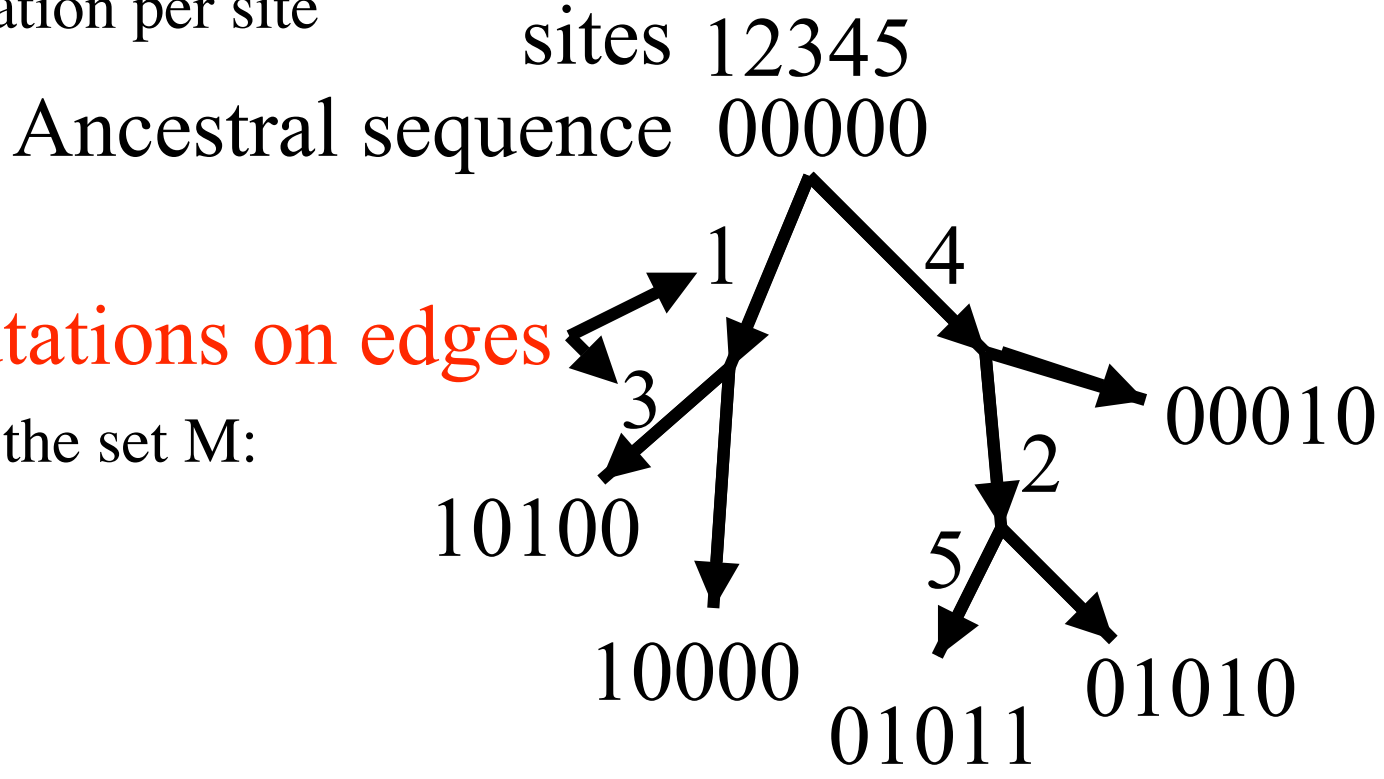
Can Integer Programming efficiently solve these problems in practice on ranges of data of current interest in biology?

We have recently developed ILPs for over twenty such problems and intensively studied their performance (speed, size and biological utility). We discuss three such problems

here.

# Starting Model: Compatibility, Perfect Phylogeny, with binary sequences

Only one mutation per site allowed.




Site mutations on edges

The tree derives the set M:

10100
10000
01011
01010
00010

Extant sequences at the leaves

# Everyone here knows

Classic NASC: A set of sequences with no duplicate columns can be generated on a unique perfect phylogeny if and only if no two columns (sites) contain all four binary pairs (gametes):

0,0 and 0,1 and 1,0 and 1,1

This is the 4-Gamete or Compatibility Test

A pair of sites that has all four binary pairs is called incompatible, otherwise is called compatible.

For M of dimension n by m, the existence of a perfect phylogeny, or the test for pairwise compatibility for M, can be tested in O(nm) time and a tree built in that time, if there is one.

# Problem M1: Perfect Phylogeny with Missing Data

Given binary sequences M with some ? entries, change each ? to

0 or 1 in order to minimize the resulting number of incompatible pairs of sites.

Special Case (Existence Problem):

Determine if the ?s can be set to 0, 1so that there are no resulting incompatibilities. NP-hard in general, but

if the root of the required perfect phylogeny is specified, then the problem has an elegant poly-time solution (Pe'er, Sharan, Shamir).

# Simple ILP for Problem M1

If cell (i,p) in M has a ?, create a binary variable Y(i,p) indicating whether the value will be set to 0 or to 1.

For each pair of sites p, q that could be made incompatible, let D(p,q) be the set of missing or deficient gametes in site pair p,q, needed to make sites p,q incompatible.

For each gamete a,b in D(p,q), create the binary variable B(p,q,a,b),

and create inequalities to set B(p,q,a,b) to 1 if the Y variables for cells for sites p,q are set so that gamete a,b is created in some row for sites p,q.

Example

| p | q |
|---|---|
| 0 | 0 |
| ? | 1 |
| 1 | 0 |
| ? | ? |
| ? | 0 |
| 0 | ? |

D(p,q) = {1,1;  0,1}

To set the B variables, the ILP will have inequalities
for each a,b in D(p,q), one for each row where a,b could be created in site pair p,q.

For example, for a,b = 1,1 the ILP has:
Y(2,p) <= B(p,q,1,1)  for row 2
Y(4,p) + Y(4,q) -- B(p,q,1,1) <= 1  for row 4

Example continued

| p | q |
|---|---|
| 0 | 0 |
| ? | 1 |
| 1 | 0 |
| ? | ? |
| ? | 0 |
| 0 | ? |

D(p,q) = {1,1;  0,1}

For a,b = 0,1 the ILP has:

Y(2,p) + B(p,q,0,1) => 1  for row 2
Y(4,q) -- Y(4,p) -- B(p,q,0,1) <= 0  for row 4
Y(6,q) -- B(p,q,0,1) <= 0  for row 6

The ILP also has a variable C(p,q) which is set to 1 if every gamete in D(p,q) is created at site-pair p,q.

In the example:

$$B(p, q, 1, 1) + B(p, q, 0, 1) -- C(p,q) <= 1$$

So, C(p,q) is set to 1 if (but not only if) the Y variables for sites p, q (missing entries in columns p, q) are
set so that sites p and q become incompatible.

If M is an n by m matrix, then we have at most nm Y variables; $2m^2$ B variables; $m^2/2$ C variables; and $O(nm^2)$ inequalities in worst-case.

Finally, we have the objective function:

$$\text{Minimize} \sum_{(p,q) \text{ in } P} C(p, q)$$

Where P is the set of site-pairs that could be made to be incompatible.

Empirically, these ILPs solve very quickly (CPLEX 9) in fractions of seconds, or seconds
even for m = n = 100 and percentage of missing values up to 30%.
Data was generated with recombination and homoplasy by the program MS and modifications of MS.
Details are in COCOON 2007, Gusfield, Frid, Brown

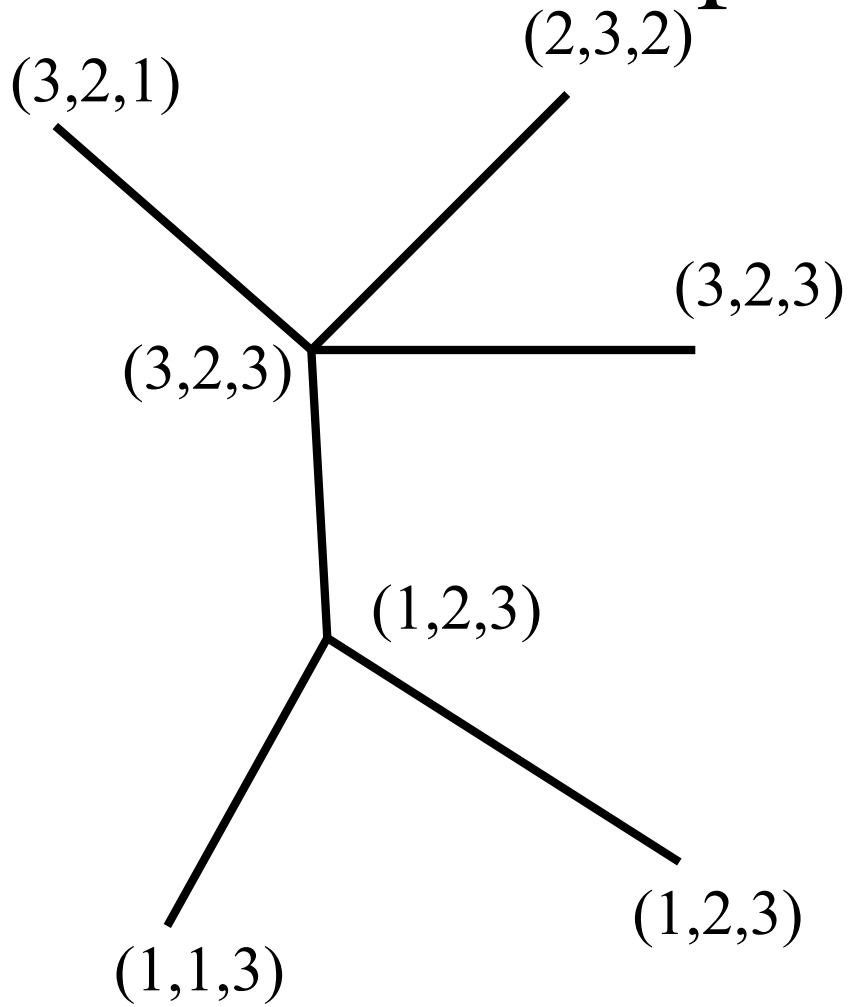
# Extension to non-binary characters

We detail the case of three and four allowed states per character.

# What is a Perfect Phylogeny for non-binary characters?

- Input consists of  n sequences M with m sites (characters) each, where each site can take one of k states.
- In a Perfect Phylogeny T for M, each node of T is labeled with an m-length sequence where each site has a value from 1 to k.
- T has n leaves, one for each sequence in M, labeled by that sequence.
- For each character-state pair (C,s), the nodes of T that are labeled with state s for character C, form a connected subtree of T. It follows that the subtrees for any C are node-disjoint

# Example: A perfect phylogeny for input M

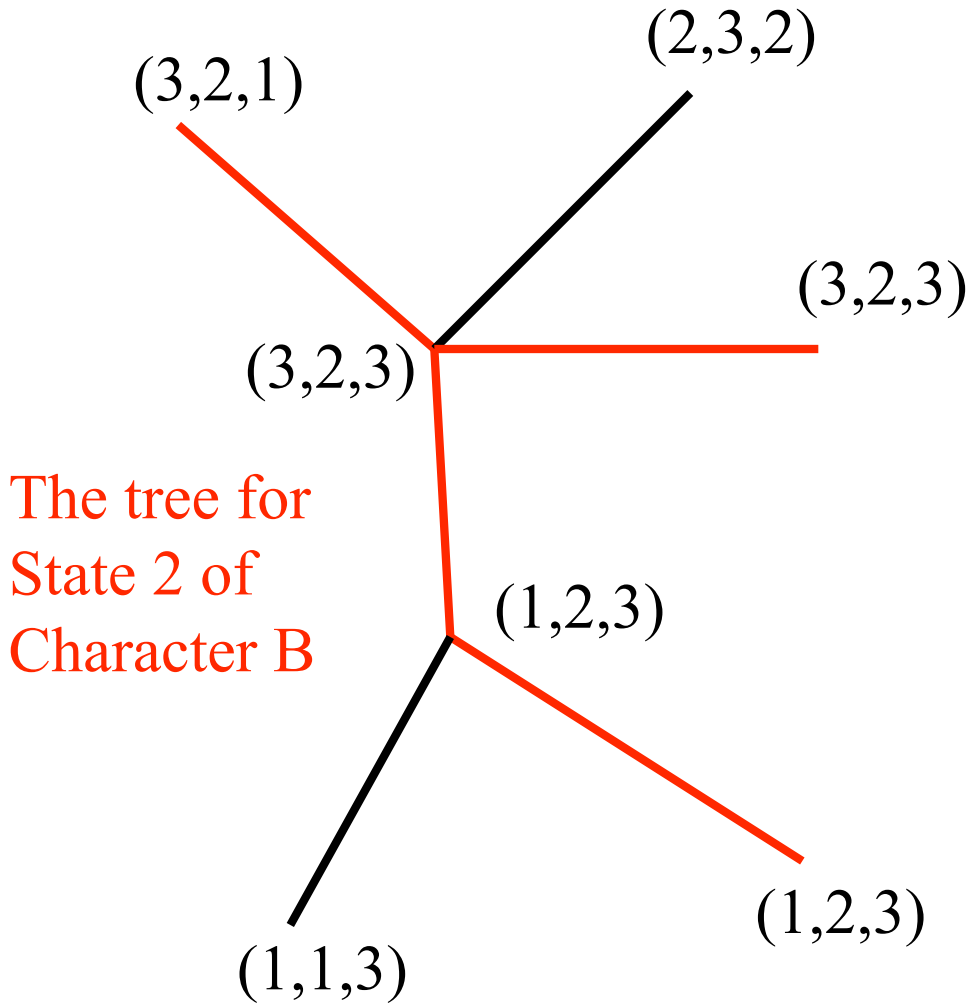



| | A | B | C |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 2 | 3 | 2 |
| 3 | 3 | 2 | 3 |
| 4 | 1 | 1 | 3 |
| 5 | 1 | 2 | 3 |

M

n = 5
m = 3
k = 3

# Example




The tree for State 2 of Character B

| | A | B | C |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 2 | 3 | 2 |
| 3 | 3 | 2 | 3 |
| 4 | 1 | 1 | 3 |
| 5 | 1 | 2 | 3 |

M

n = 5
m = 3
k = 3

# Existence problem for three states

Is there a way to set the ?s so there is a 3-state perfect phylogeny?

# Dress-Steel solution for 3-state Perfect phylogeny given complete data (1991)

- Recode each site M(i) of M as three binary sites M'(i,1), M'(i,2), M'(i,3) each indicating the taxa that have state 1, 2, or 3.
- Theorem (DS) There is a 3-state perfect phylogeny for M, if and only if there is a binary-character perfect phylogeny for some subset of M' consisting of exactly two of the columns

  M'(i,1), M'(i,2), M'(i,3), for each column i of M.

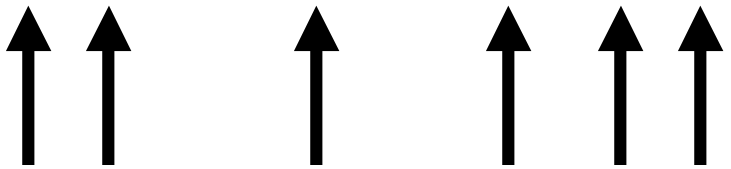
# Example

M

| | A | B | C |
|---|---|---|---|
| 1 | 3 | 2 | 1 |
| 2 | 2 | 3 | 2 |
| 3 | 3 | 2 | 3 |
| 4 | 1 | 1 | 3 |
| 5 | 1 | 2 | 3 |

M’

| | A,1 | A,2 | A,3 | B,1 | B,2 | B,3 | C,1 | C,2 | C,3 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 4 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |



Compatible subset

# ILP for the DS solution

S(i,1), S(i,2), S(i,3) are binary variables indicating which columns of M' associated with column i in M will be selected. Then we use the inequalities

$S(i,1) + S(i,2) + S(i,3) = 2$

$S(i,x) + C(i,x; j,y) + S(j,y) <= 2$ etc. for $x,y = \{1,2,3\}$, and

C(i,x;j,y) is the variable (essentially from the M1 ILP) that is forced to 1 if columns (i,x) and (j,y) in M' are incompatible.

From the DS theorem, the ILP is feasible if and only if there is a 3-state perfect phylogeny for M.

# Back to the problem of missing data

Now we can extend the DS solution to the case of missing values:

When there is a ? in cell (p,q) of M, we use binary variables
Y(p,q,1), Y(p,q,2), Y(p,q,3) to indicate their values in M', and add the equality:

Y(p,q,1) + Y(p,q,2) + Y(p,q,3) = 1
which sets the ? in (p,q) to either 1,2, or 3.

The resulting ILP is feasible if and only if the ?s in M have been set to allow a 3-state perfect phylogeny.
That solves the Existence Problem for three states per character.

# Empirical Results: The 3-state Existence Problem

| | 0% | 5% | 10% | 20% | 35% | missing values |
|---|---|---|---|---|---|---|
| 50 by 25, 3PP exists | 0.0098 | 0.3 | 0.6 | 1.16 | 56.0 | seconds |
| 100 by 50 3PP exists | 0.03 | 4.0 | 6.9 | 13.9 | 2492.0 | seconds |

Times for data where no 3-state Perfect Phylogeny exists were similar, but smaller!

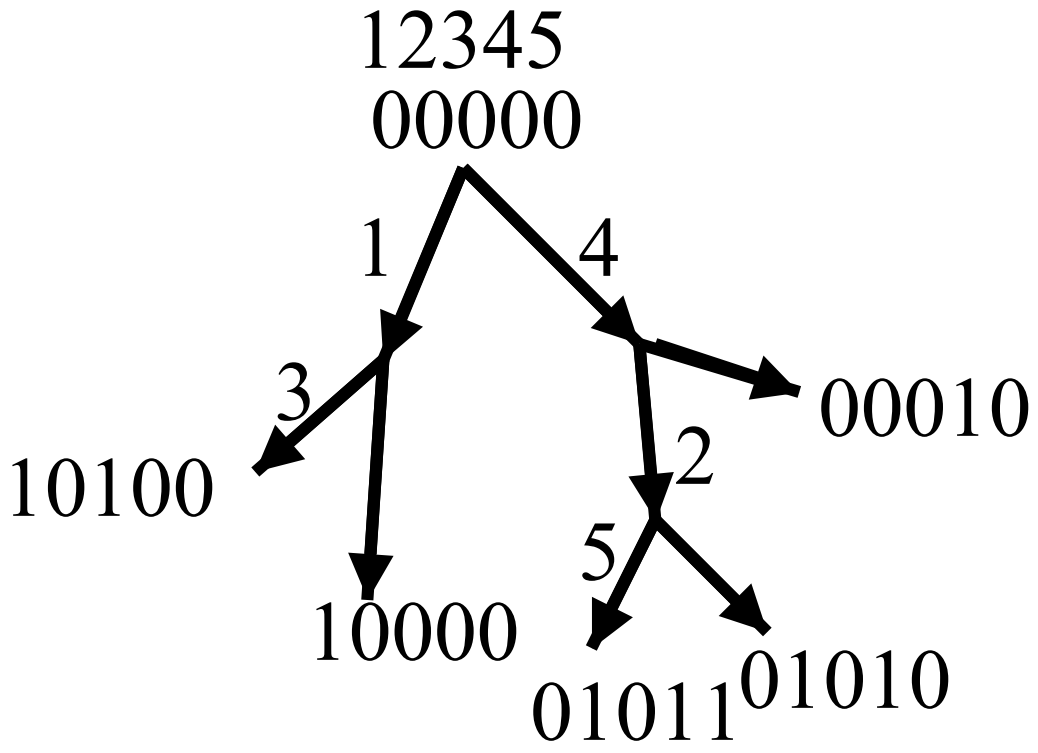
We have also developed efficient ILP solutions for the Perfect Phylogeny Problem with missing data, for the case of 4 allowed states, and a different solution for any arbitrary, but fixed number of states.

# Topic III: Phylogenetic Networks with Recombination

# Recombination: A richer model than Perfect Phylogeny

M

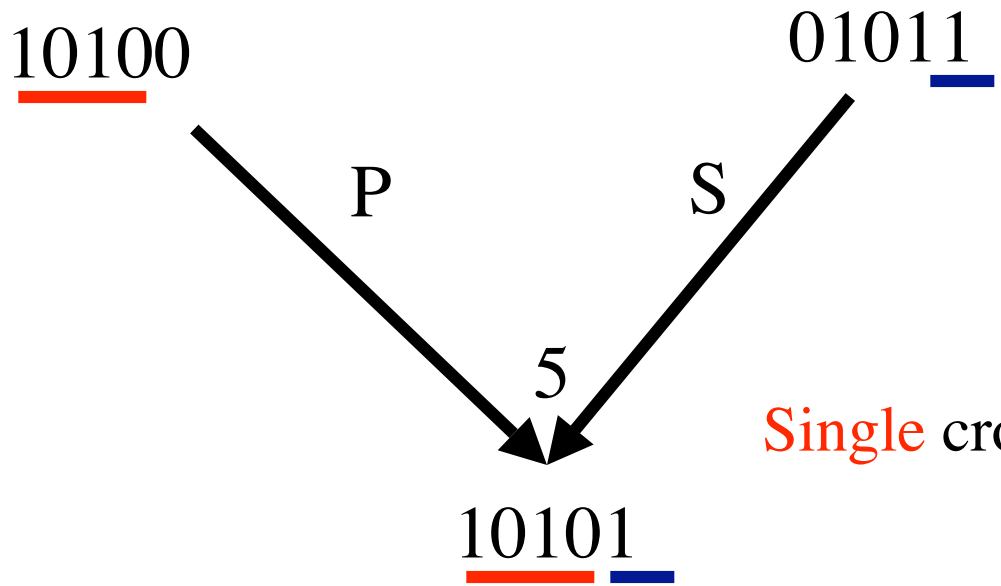
10100
10000
01011
01010
00010
10101 added




Pair 4, 5 fails the four gamete-test. The sites 4, 5 are incompatible.

Real sequence histories often involve recombination.

# Sequence Recombination




Single crossover recombination

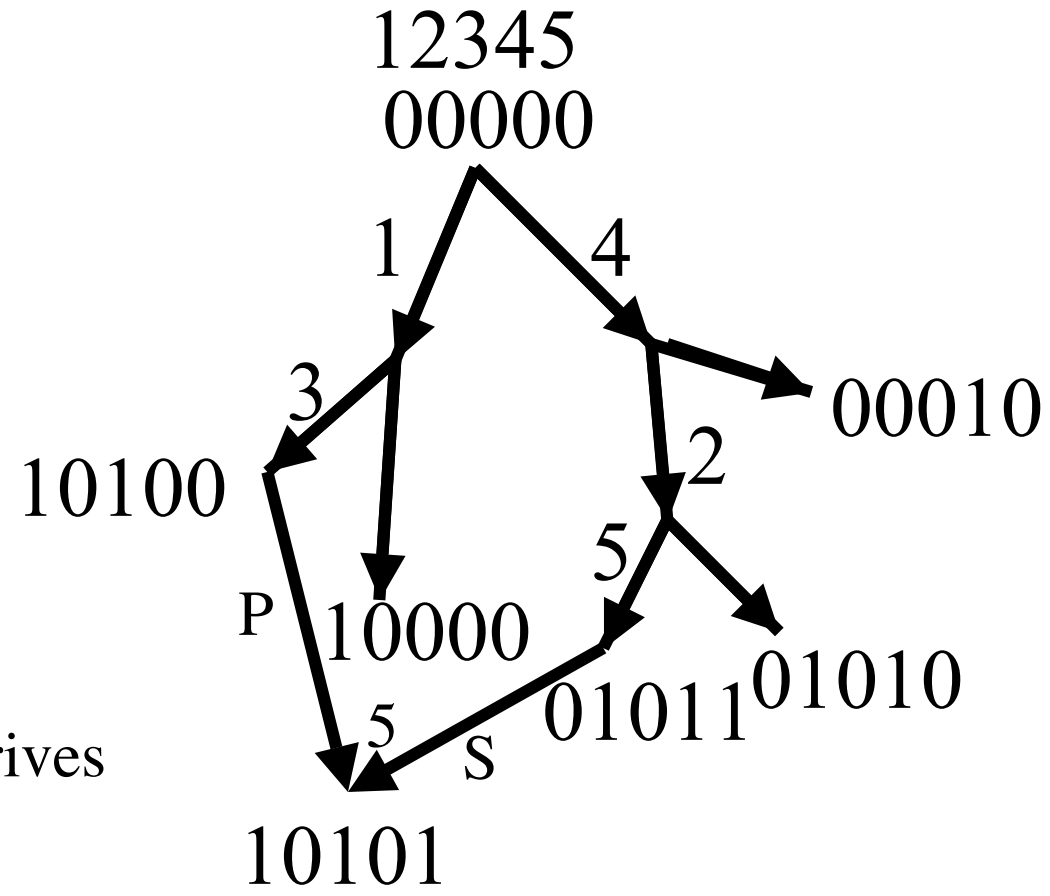
A recombination of P and S at recombination point 5.

The first 4 sites come from P (Prefix) and the sites from 5 onward come from S (Suffix).

# Network with Recombination: ARG

M

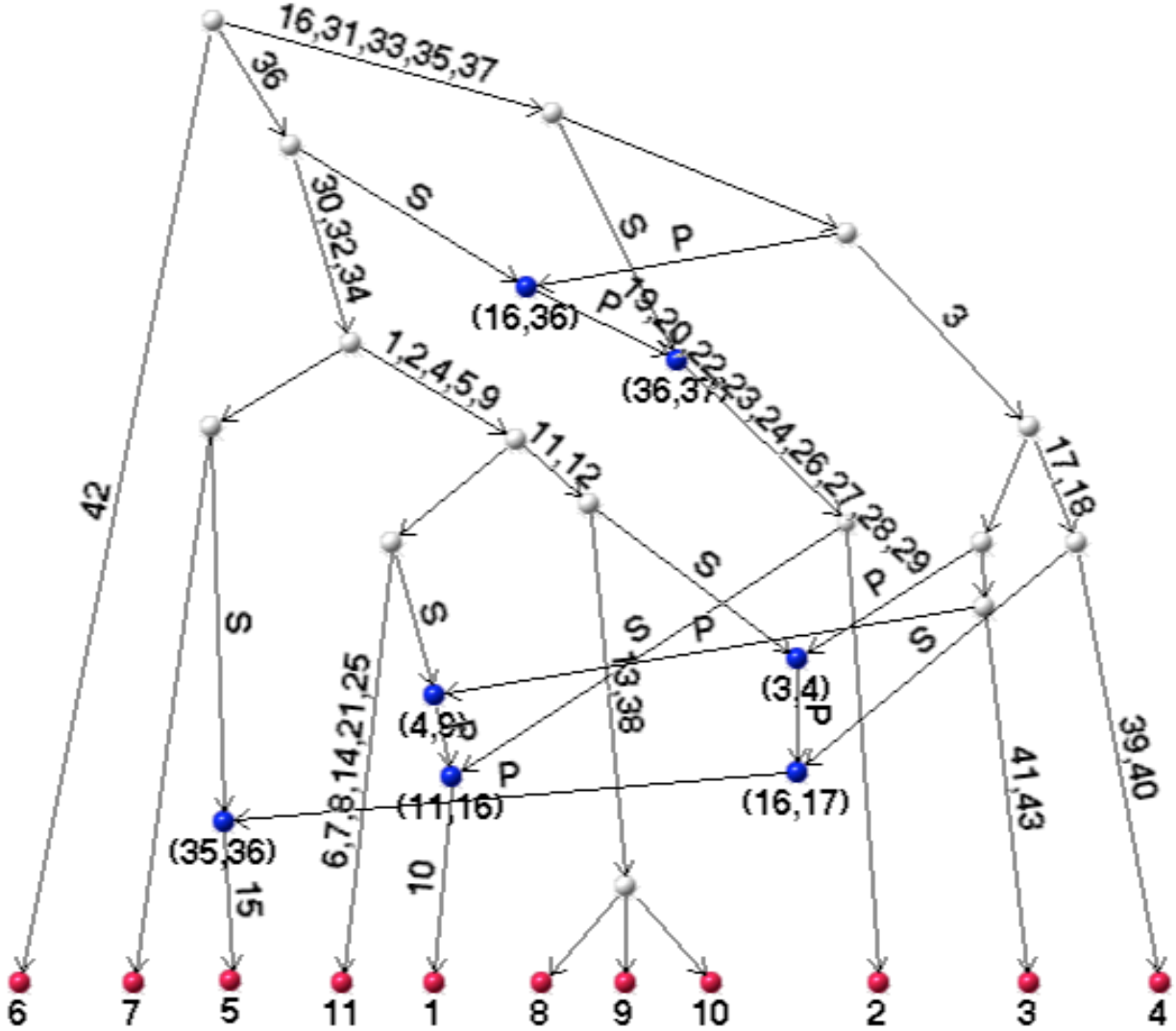
10100
10000
01011
01010
00010
10101 new




The previous tree with one recombination event now derives all the sequences.

# A Min ARG for Kreitman's data

ARG created by SHRUB



# Results on Reconstructing the Evolution of SNP Sequences

- Part I: Clean mathematical and algorithmic results: Galled-Trees, near-uniqueness, graph-theory lower bound, and the Decomposition theorem

- Part II: Practical computation of Lower and Upper bounds on the number of recombinations needed. Construction of (optimal) phylogenetic networks; uniform sampling; haplotyping with ARGs; LD mapping …
- Part III: Varied Biological Applications
- Part IV: Extension to Gene Conversion
- Part V: The Minimum Mosaic Model of Recombination

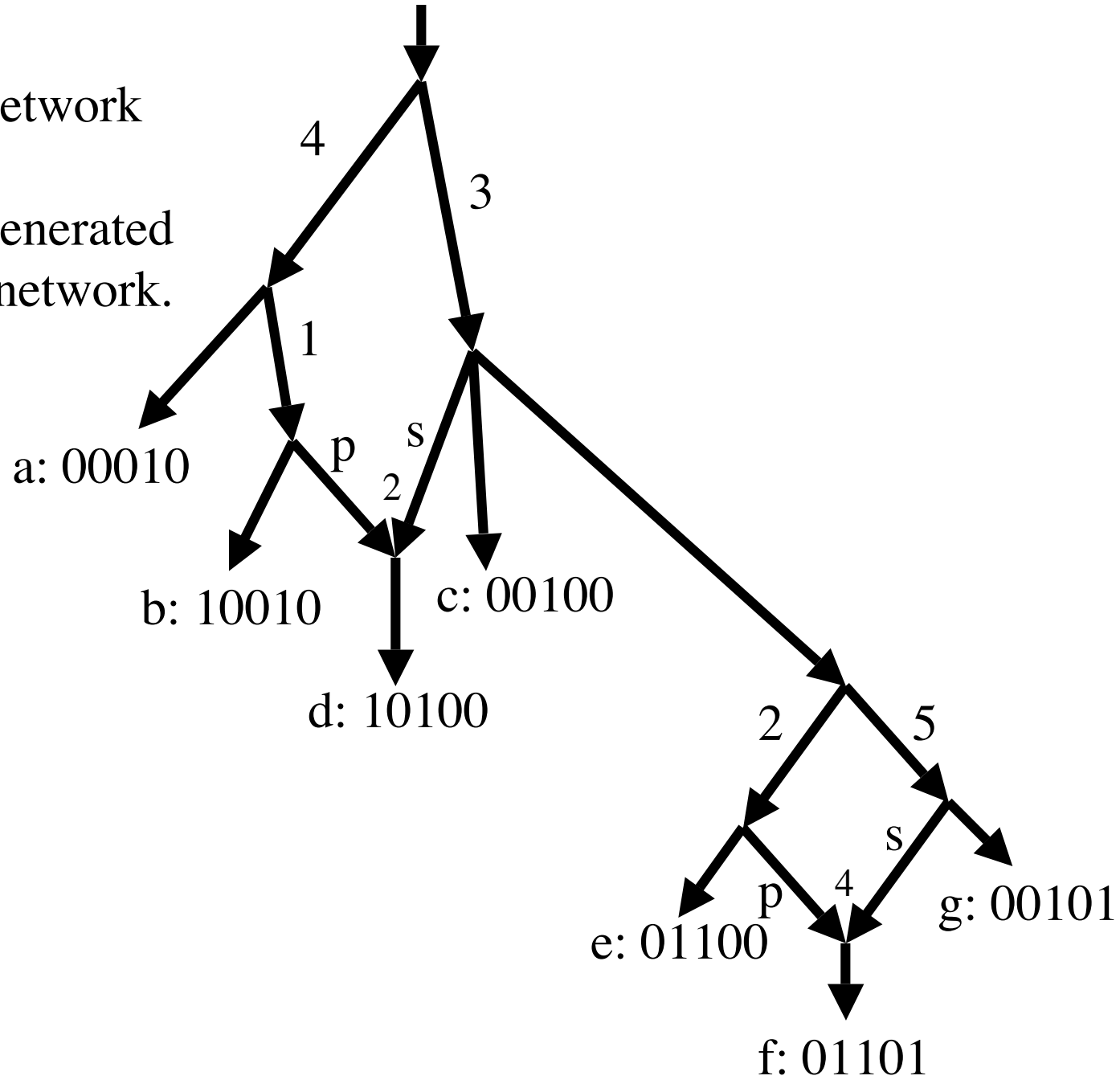
This talk will discuss topics in Parts I

# Problem: If not a tree, then what?

If the set of sequences M cannot be derived on a perfect phylogeny (true tree) how much deviation from a tree is required?

We want a network for M that uses a small number of recombinations, and we want the resulting network to be as ``tree-like'' as possible.

A tree-like network for the same sequences generated by the prior network.



# Recombination Cycles

- In a Phylogenetic Network, with a recombination node x, if we trace two paths backwards from x, then the paths will eventually meet.
- The cycle specified by those two paths is called a ``recombination cycle”.

# Galled-Trees

- A phylogenetic network where no recombination cycles share an edge is called a galled tree.
- A cycle in a galled-tree is called a gall.
- Question: if M cannot be generated on a true tree, can it be generated on a galled-tree?

# Results about galled-trees

- Theorem: Efficient (provably polynomial-time) algorithm to determine whether or not any sequence set M can be derived on a galled-tree.

- Theorem: A galled-tree (if one exists) produced by the algorithm minimizes the number of recombinations used over all possible phylogenetic-networks.

- Theorem: If M can be derived on a galled tree, then the Galled-Tree is ``nearly unique". This is important for biological conclusions derived from the galled-tree.

Papers from 2003-2007.

# Elaboration on Near Uniqueness

Theorem: The number of arrangements (permutations) of the sites on any gall is
at most three, and this happens only if the gall has two sites.

If the gall has more than two sites, then the number of arrangements is at most two.

If the gall has four or more sites, with at least two sites on each side of the recombination point (not the side of the gall) then the arrangement is forced and unique.

Theorem: All other features of the galled-trees for M are invariant.

# A whiff of the ideas behind the results

# Incompatible Sites

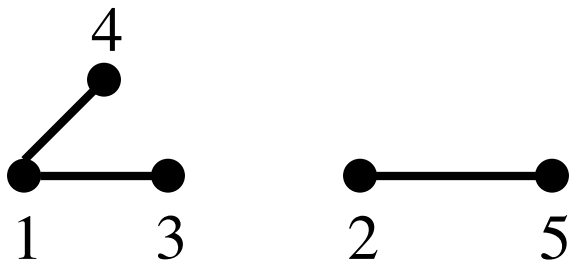
A pair of sites (columns) of M that fail the 4-gametes test are said to be incompatible.

A site that is not in such a pair is compatible.

M

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| a | 0 | 0 | 0 | 1 | 0 |
| b | 1 | 0 | 0 | 1 | 0 |
| c | 0 | 0 | 1 | 0 | 0 |
| d | 1 | 0 | 1 | 0 | 0 |
| e | 0 | 1 | 1 | 0 | 0 |
| f | 0 | 1 | 1 | 0 | 1 |
| g | 0 | 0 | 1 | 0 | 1 |

Incompatibility Graph G(M)




Two nodes are connected iff the pair of sites are incompatible, i.e, fail the 4-gamete test.

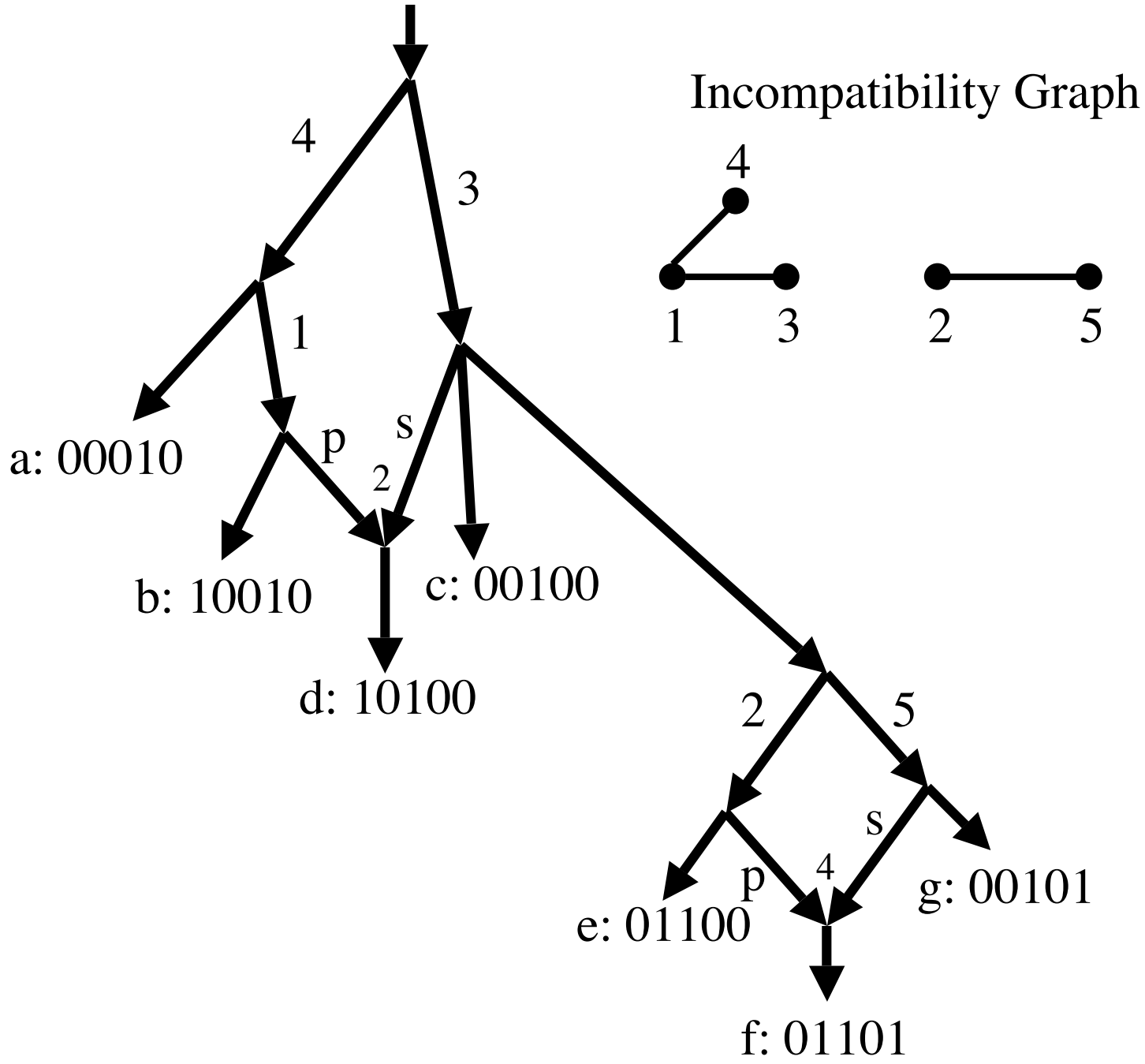
THE MAIN TOOL: We represent the pairwise incompatibilities in a incompatibility graph.

# The connected components of G(M) are very informative

- Theorem: The number of non-trivial connected components is a lower-bound on the number of recombinations needed in any network.

- Theorem: When M can be derived on a galled-tree, all the incompatible sites in a gall must come from a single connected component C, and that gall must contain all the sites from C. Compatible sites need not be inside any blob.

- In a galled-tree the number of recombinations is exactly the number of connected components in G(M), and hence is minimum over all possible phylogenetic networks for M.


Incompatibility Graph
4
1
3
2
5
4
3
1
a: 00010
p
s
2
b: 10010
c: 00100
d: 10100
2
5
s
p
4
e: 01100
g: 00101
f: 01101

# Coming full circle - back to genotypes

When can a set of genotypes be explained by a set of haplotypes that derived on a galled-tree, rather than on a perfect phylogeny?

Recently, we developed an Integer Linear Programming solution to this problem, and are now testing the practical efficiency of it.

(Brown, Gusfield).

# Minimizing Recombinations in unconstrained networks

- Problem: given a set of sequences M, find a phylogenetic network generating M, minimizing the number of recombinations used to generate M, allowing only one mutation per site. This has biological meaning in appropriate contexts.

- We can solve this problem in poly-time for the special case of Galled-Trees.
- The minimization problem is NP-hard in general.

# Minimization is an NP-hard Problem

What we have done:

1. Solve small data-sets optimally with exponential-time methods or with algorithms that work well in practice;

2. Efficiently compute lower and upper bounds on the number of needed recombinations.

3. Apply these methods to address specific biological and bio-tech questions.