Algorithms for estimating and reconstructing recombination in populations

Dan Gusfield

UC Davis

Different parts of this work are joint with Satish Eddhu, Charles Langley, Dean Hickerson, Yun Song, Yufeng Wu, Z. Ding

IWBRA 2006, May 29, Reading
England

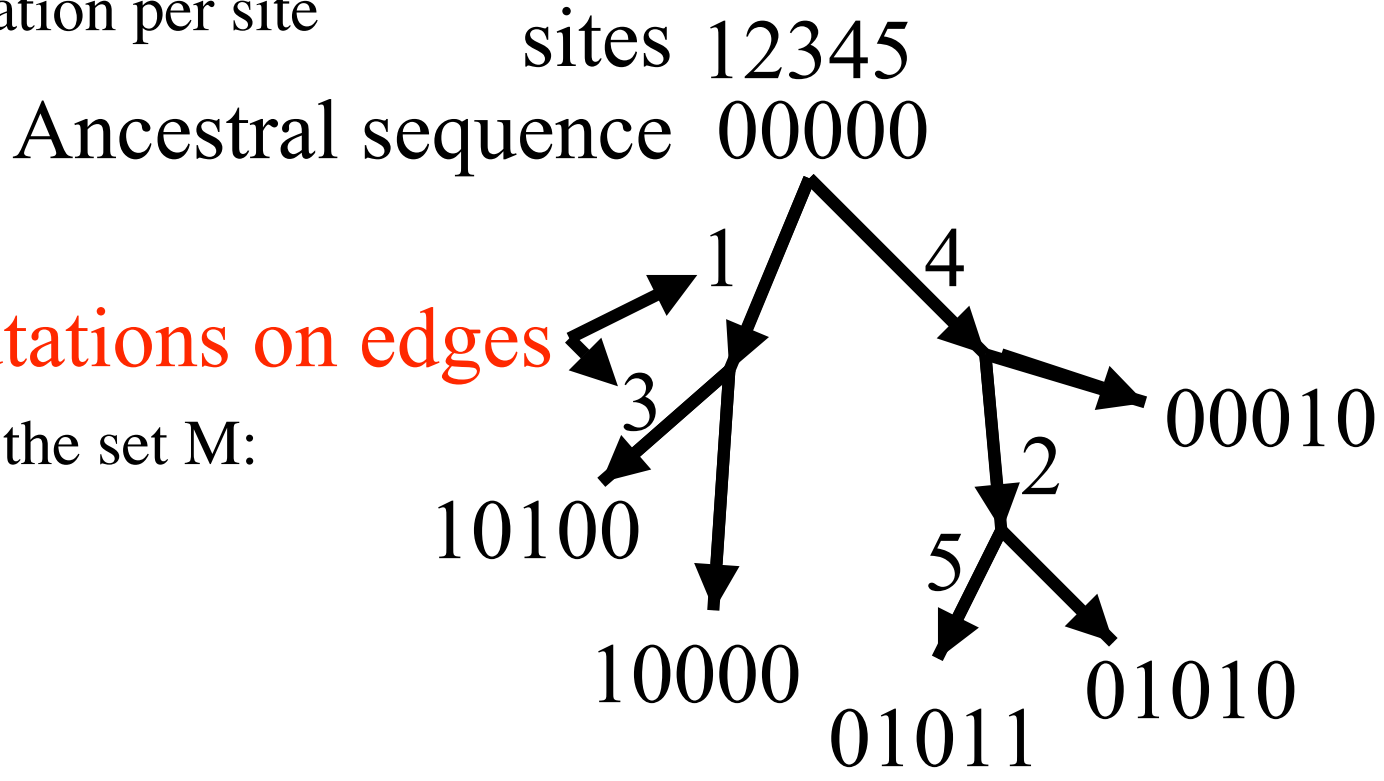
Reconstructing the Evolution of SNP (or SFP) Sequences

- 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
- Part III: Applications
- Part IV: Extension to Gene Conversion

The Perfect Phylogeny Model for SNP 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

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

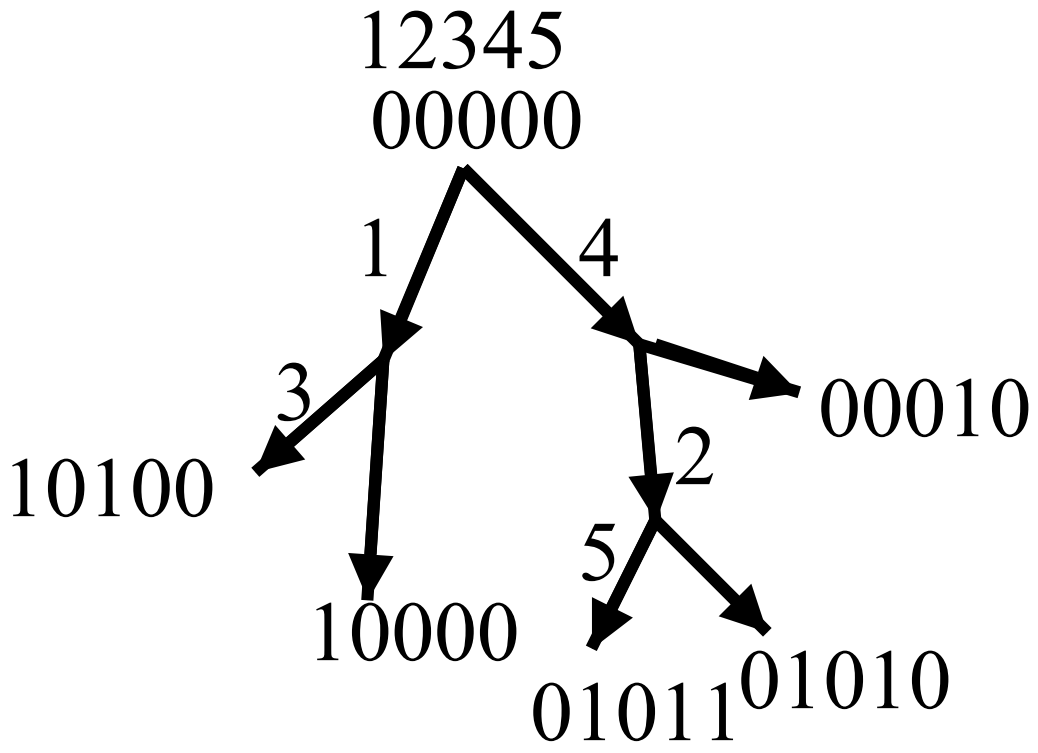
Classic NASC: Arrange the sequences in a matrix. Then (with no duplicate columns), the sequences can be generated on a unique perfect phylogeny if and only if no two columns (sites) contain all four pairs:

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

This is the 4-Gamete Test

A richer model

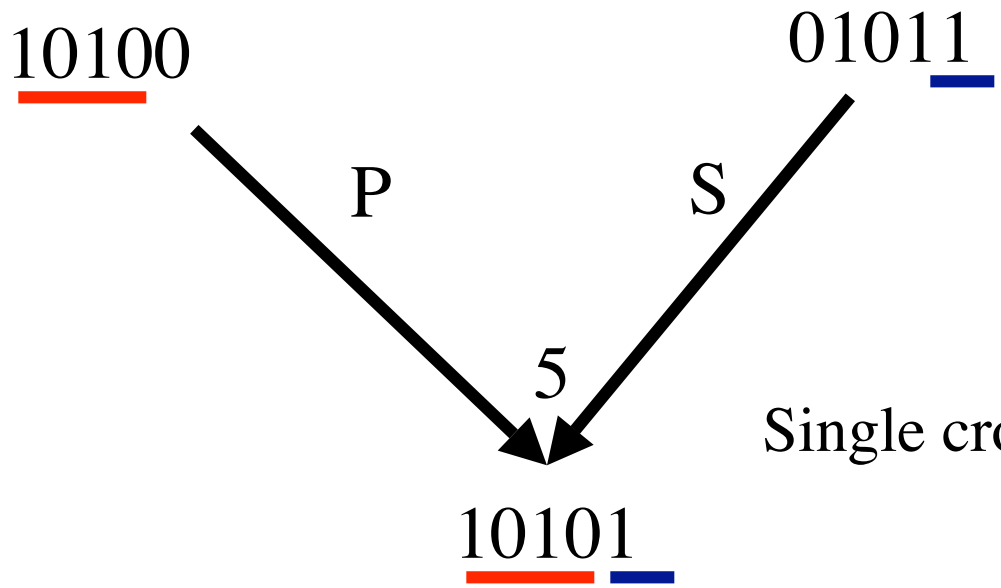
	10100
	10000
M	01011
	01010
	00010
	10101 added



Pair 4, 5 fails the four gamete-test. The sites 4, 5 ``conflict".

Real sequence histories often involve recombination.

Sequence Recombination



Single crossover recombination

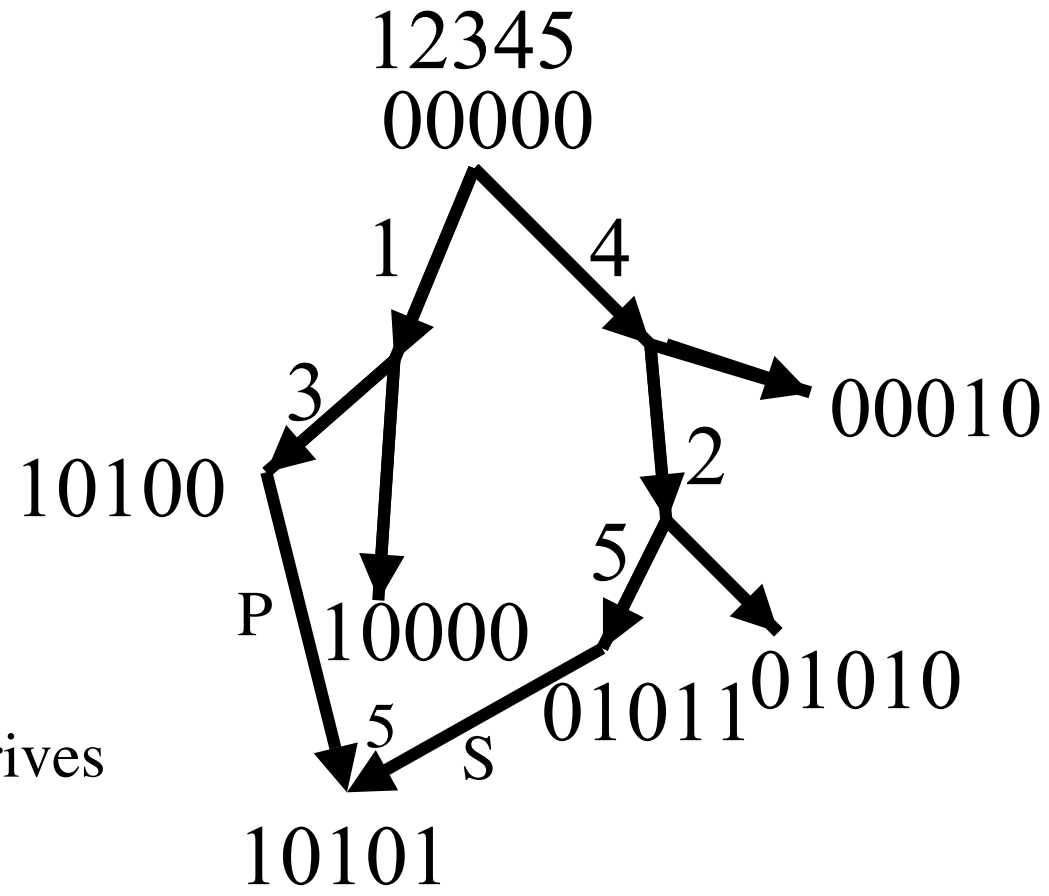
A recombination of P and S at recombination point 5.

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

Network with Recombination

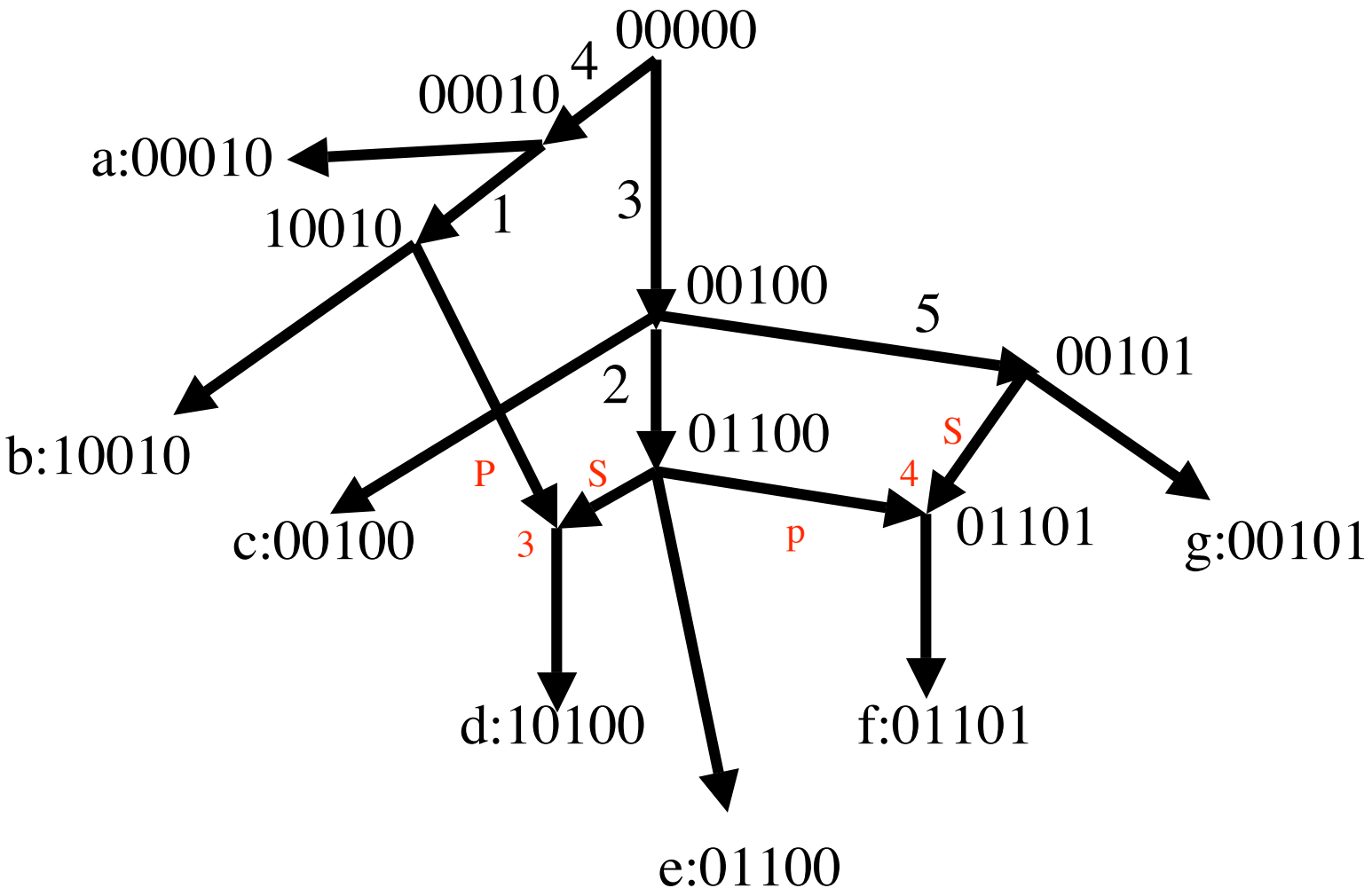
M

10100
10000
01011
01010
00010
10101 new



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

A Phylogenetic Network or ARG

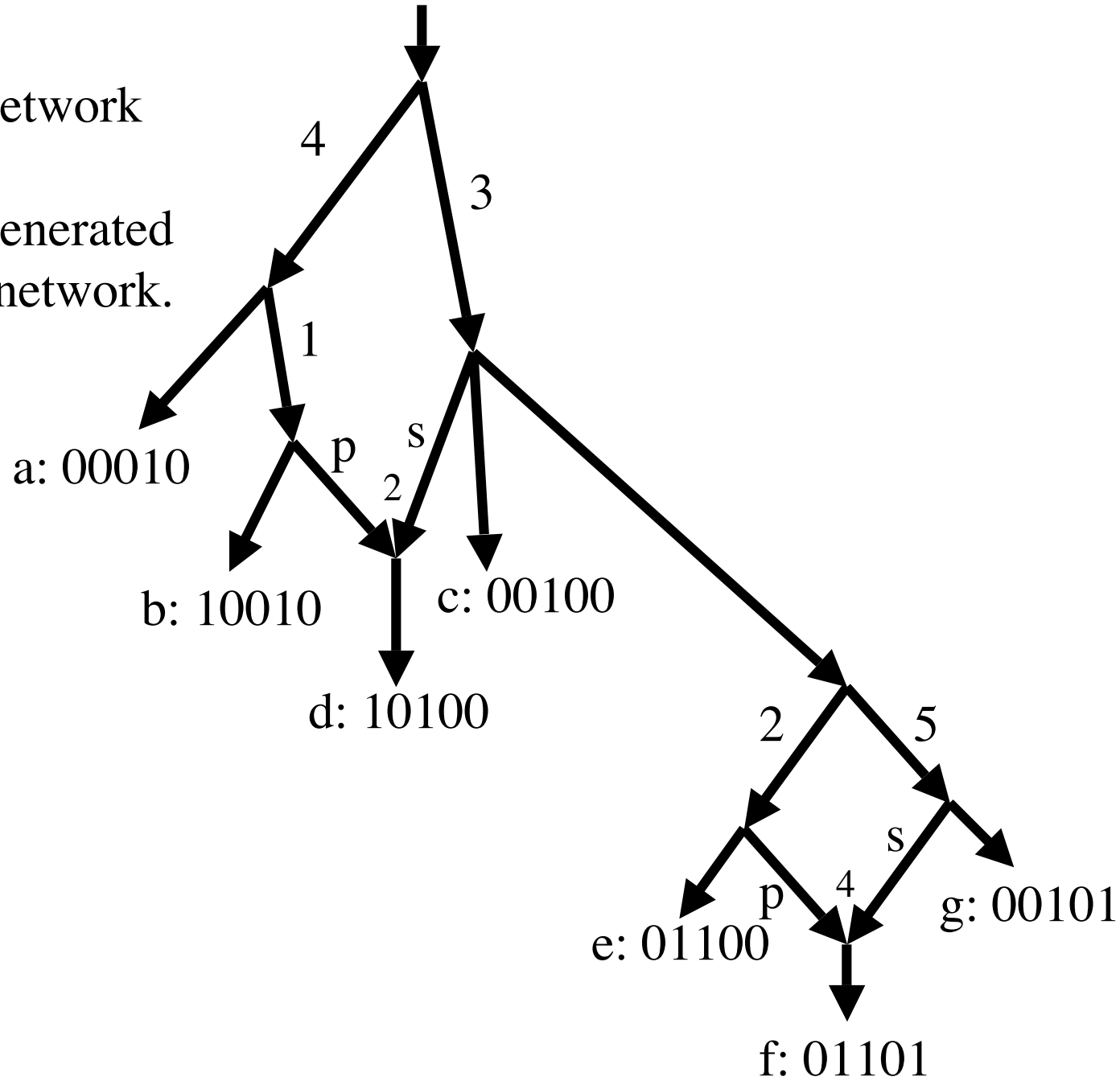


If not a tree, is something very tree like possible?

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 2003-3005.

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

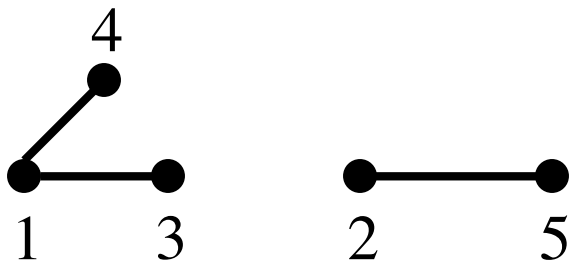
A pair of sites (columns) of M that fail the 4-gametes test are said to be incompatible.

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)

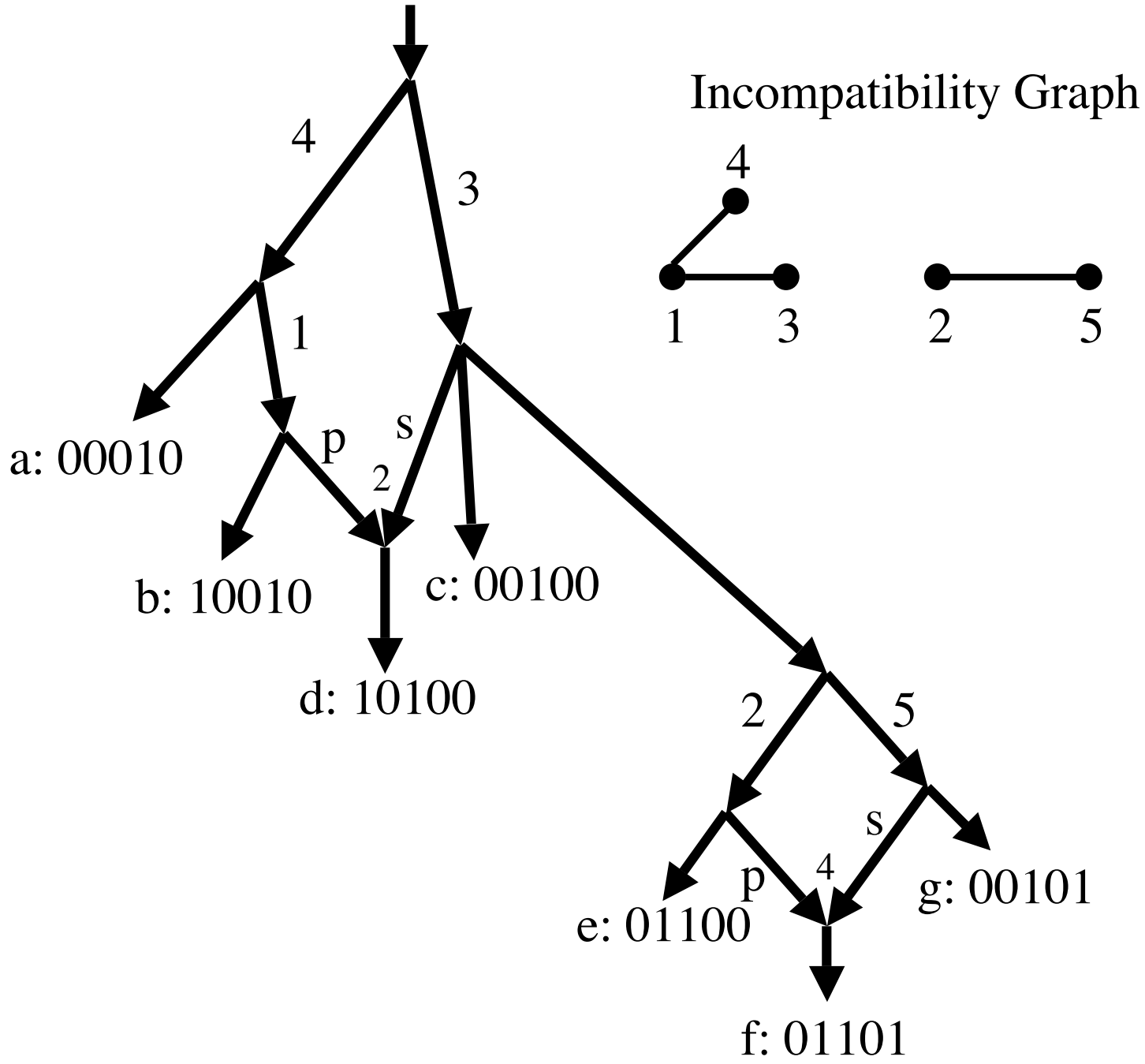


Two nodes are connected iff the pair of sites are incompatible, i.e, fail the 4-gamete test.

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

Generalizing beyond Galled-Trees

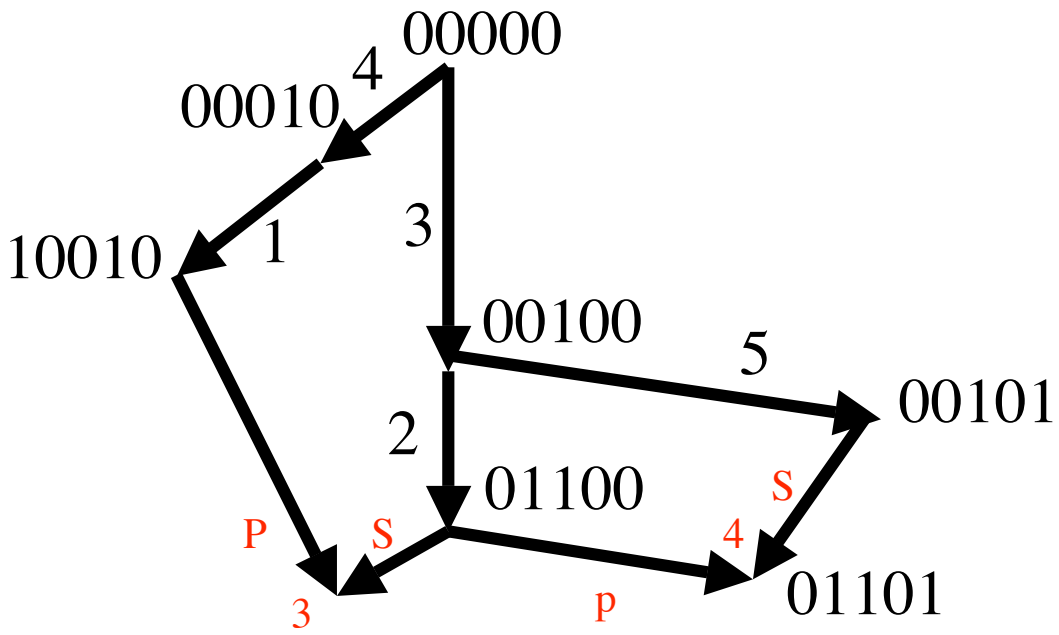
When M cannot be generated on a true tree or a galled-tree, what then?

What role for the connected components of G(M) in general?

What is the most tree-like network for M?

Can we minimize the number of recombinations needed to generate M?

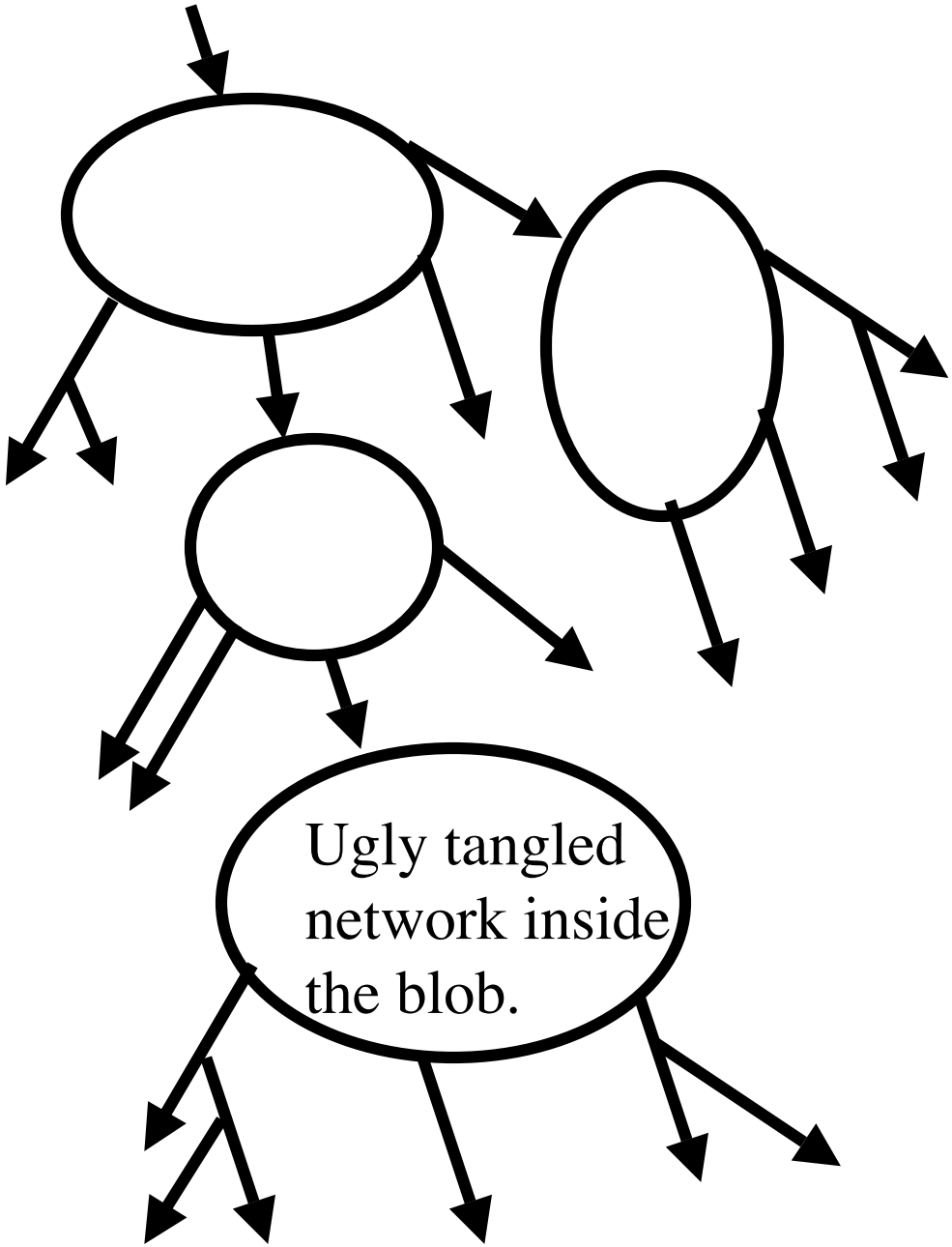
A maximal set of intersecting cycles forms a Blob



Blobs generalize Galls

- In any phylogenetic network a maximal set of intersecting cycles is called a blob. A blob with only one cycle is a gall.
- Contracting each blob results in a directed, rooted tree, otherwise one of the "blobs" was not maximal. Simple, but key insight.
- So every phylogenetic network can be viewed as a directed tree of blobs - a blobbed-tree.

The blobs are the non-tree-like parts of the network.



Every network is a tree of blobs.

A network where every blob is a single cycle is a Galled-Tree.

The Decomposition Theorem (Recomb, April 2005)

Theorem: For any set of sequences M, there is a phylogenetic network that derives M, where each blob contains all and only the sites in one non-trivial connected component of G(M). The compatible sites can always be put on edges outside of any blob. This is the finest network decomposition possible and the most ``tree-like'' network for M.

However, while such networks always exist, they are not guaranteed to minimize the number of recombinations (Y. Song, Aug. 2005)

Minimizing recombinations in unconstrained networks

- When a galled-tree exists it minimizes the number of recombinations used over all possible phylogenetic networks for M. But a galled-tree is not always possible.

- Problem: given a set of sequences M, find a phylogenetic network generating M, minimizing the number of recombinations used to generate M.

Minimization is an NP-hard Problem

There is no known efficient
solution to this problem and there likely will never be one.

What we do:
Solve small data-sets optimally with
algorithms that are not provably efficient but work well in practice;

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

Part II: Constructing optimal phylogenetic networks in general

Computing close lower and upper bounds on the minimum number of recombinations needed to derive M. (ISMB 2005)

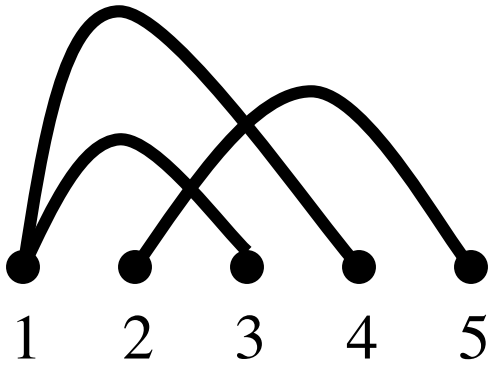
The grandfather of all lower bounds - HK 1985

- Arrange the nodes of the incompatibility graph on the line in order that the sites appear in the sequence. This bound requires a linear order.
- The HK bound is the minimum number of vertical lines needed to cut every edge in the incompatibility graph. Weak bound, but widely used - not only to bound the number of recombinations, but also to suggest their locations.

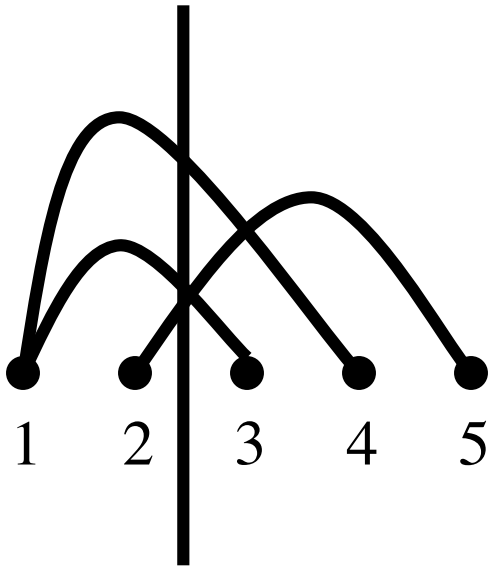
Justification for HK

If two sites are incompatible, there must have been some recombination where the crossover point is between the two sites.

HK Lower Bound



HK Lower Bound = 1



More general view of HK

Given a set of intervals on the line, and for each interval I, a number N(I), define the composite problem: Find the minimum number of vertical lines so that every interval I intersects at least N(I) of the vertical lines.

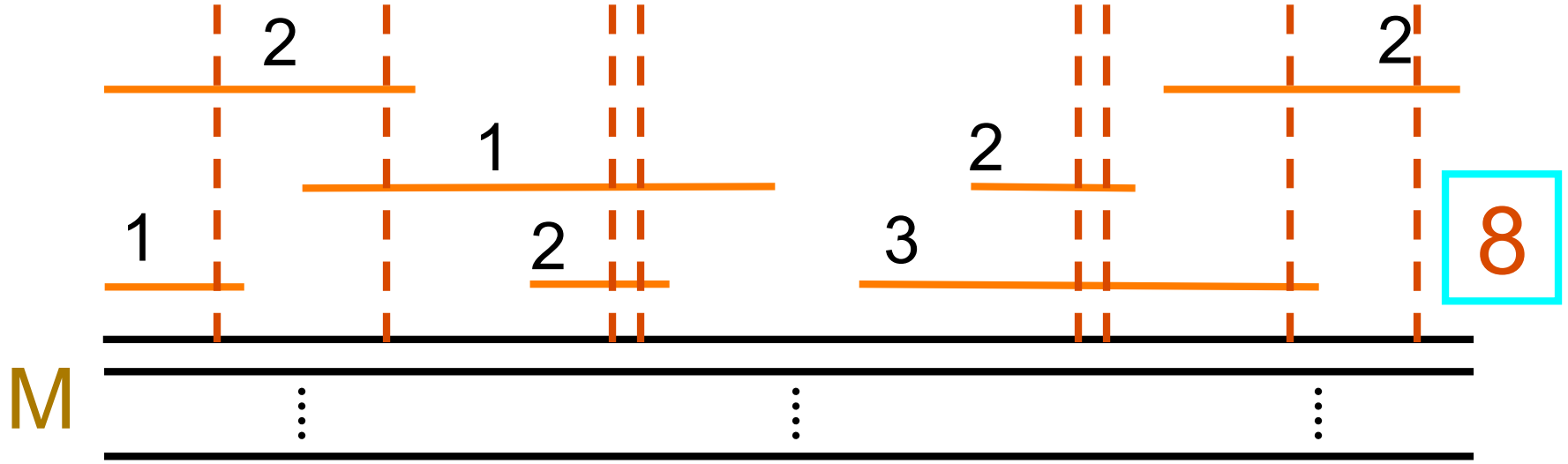
In HK, each incompatibility defines an interval I where N(I) = 1.

The composite problem is easy to solve by a left-to-right myopic placement of vertical lines.

The Composite Method (Myers & Griffiths 2003)

1. Given a set of intervals, and
2. for each interval **I**, a number **N(I)**

Composite Problem: Find the minimum number of vertical lines so that every **I** intersects at least N(I) vertical lines.



If each N(I) is a ``local" lower bound on the number of recombinations needed in interval I, then the solution to the composite problem is a valid lower bound for the full sequences. The resulting bound is called the composite bound given the local bounds.

This general approach is called the Composite Method (Simon Myers 2002).

Haplotype Bound (Simon Myers)

- Rh = Number of distinct sequences (rows) - Number of distinct sites (columns) -1 <= minimum number of recombinations needed (folklore)

- Before computing Rh, remove any site that is compatible with all other sites. A valid lower bound results - generally increases the bound.

- Generally Rh is really bad bound, often negative, when used on large intervals, but Very Good when used as local bounds in the Composite Interval Method, and other methods.

Composite Subset Method (Myers)

- Let S be subset of sites, and Rh(S) be the haplotype bound for subset S. If the leftmost site in S is L and the rightmost site in S is R, then use Rh(S) as a local bound N(I) for interval I = [S,L].
- Compute Rh(S) on many subsets, and then solve the composite problem to find a composite bound.

RecMin (Myers)

- Computes Rh on subsets of sites, but limits the size and the span of the subsets. Default parameters are s = 6, w = 15 (s = size, w = span).
- Generally, impractical to set s and w large, so generally one doesn't know if increasing the parameters would increase the bound.
- Still, RecMin often gives a bound more than three times the HK bound. Example LPL data: HK gives 22, RecMin gives 75.

Optimal RecMin Bound (ORB)

- The Optimal RecMin Bound is the lower bound that RecMin would produce if both parameters were set to their maximum possible values.
- In general, RecMin cannot compute (in practical time) the ORB.
- We have developed a practical program, HAPBOUND, based on integer linear programming that guarantees to compute the ORB, and have incorporated ideas that lead to even higher lower bounds than the ORB.

HapBound vs. RecMin on LPL from Clark et al.

Program	Lower Bound	Time
RecMin (default)	59	3s
RecMin –s 25 –w 25	75	7944s
RecMin –s 48 –w 48	No result	5 days
HapBound ORB	75	31s
HapBound -S	78	1643s

2 Ghz PC

Example where RecMin has difficulity in Finding the ORB on a 25 by 376 Data Matrix

Program	Bound	Time
RecMin default	36	1s
RecMin –s 30 –w 30	42	3m 25s
RecMin –s 35 –w 35	43	24m 2s
RecMin –s 40 –w 40	43	2h 9m 4s
RecMin –s 45 –w 45	43	10h 20m 59s
HapBound	44	2m 59s
HapBound -S	48	39m 30s

Constructing Optimal Phylogenetic Networks in General

Optimal = minimum number of recombinations.
Called Min ARG.

The method is based on the coalescent viewpoint of sequence evolution. We build the network backwards in time.

Definition: A column is non-informative if all entries are the same, or all but one are the same.

The key tool

- Given a set of rows A and a single row r, define w(r | A - r) as the minimum number of recombinations needed to create r from A-r (well defined in our application).
- w(r | A-r) can be computed efficiently by a greedy-type algorithm.

Upper Bound Algorithm

1) Set W = 0
2) Collapse identical rows together, and remove non-informative columns. Repeat until neither is possible.
3) Let A be the data at this point. If A is empty, stop, else remove some row r from A, and set W = W + W(r | A-r). Go to step 2).

Note that the choice of r is arbitrary in Step 3), so the resulting W can vary.

An execution gives an upper bound W and specifies how to construct a network that derives the sequences using exactly W recombinations.

Each step 2 corresponds to a mutation or a coalescent event; each step 3 corresponds to a recombination event.

We can find the lowest possible W with this approach in O(2^n) time by using Dynamic Programming, and build the Min ARG at the same time.

In practice, we can use branch and bound to speed up the computation, and we have also found that branching on the best local choice, or randomizing quickly builds near-optimal ARGs.

Program: SHRUB

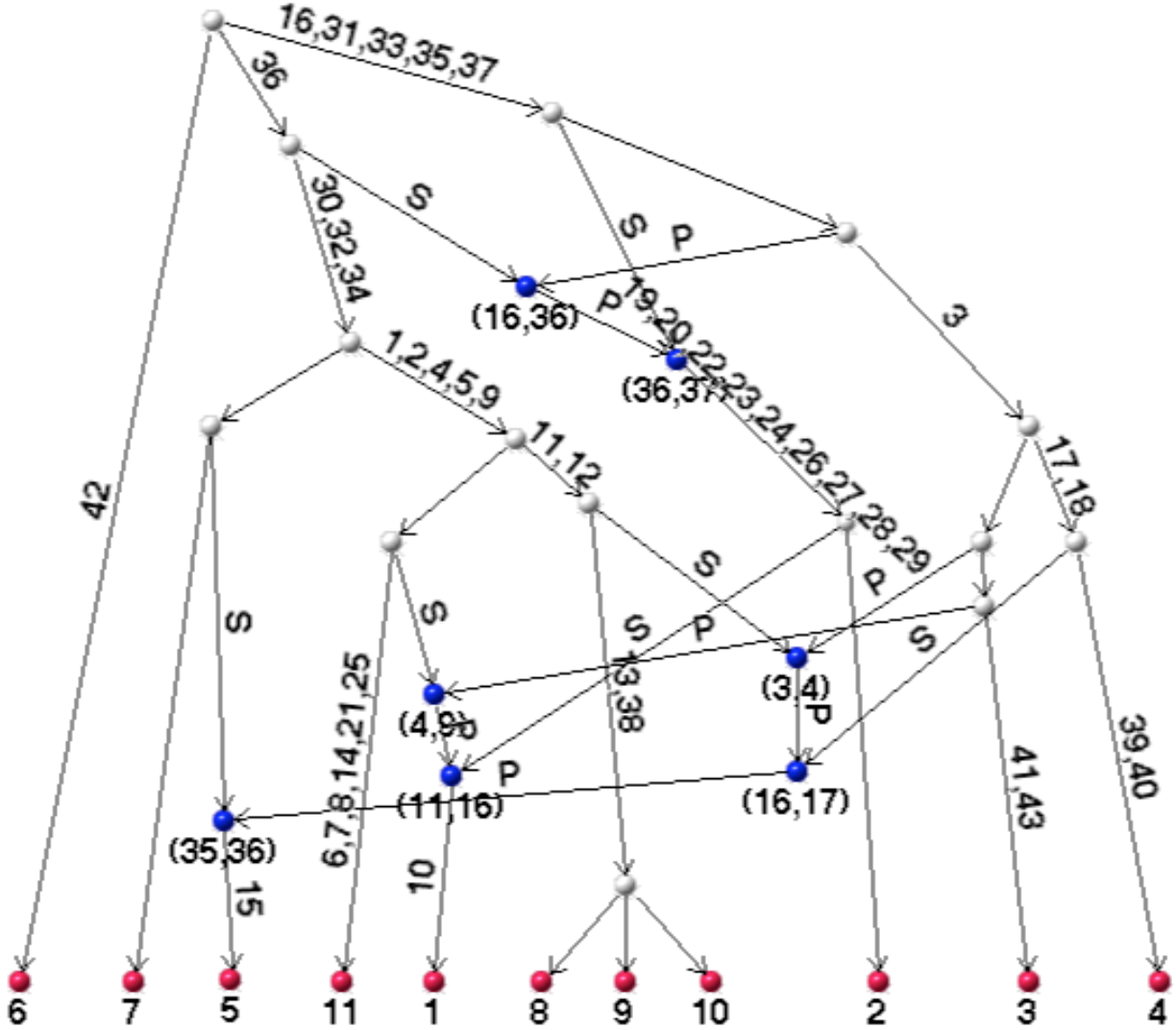
Kreitman's 1983 ADH Data

- 11 sequences, 43 segregating sites
- Both HapBound and SHRUB took only a fraction of a second to analyze this data.
- Both produced 7 for the number of detected recombination events

Therefore, independently of all other methods, our lower and upper bound methods together imply that 7 is the minimum number of recombination events.

A Min ARG for Kreitman's data

ARG created by SHRUB



The Human LPL Data (Nickerson et al. 1998)

(88 Sequences, 88 sites)

Our new lower and upper bounds

	site regions		
Population	reg 1	reg 2	reg 3
Jackson	11 (13)	10 (10)	13 (16)
N. Karelia	2 (2)	15 (17)	8 (10)
Rochester	1 (1)	14 (14)	8 (8)
All	13 (14)	21 (23)	25 (31)

Optimal RecMin Bounds

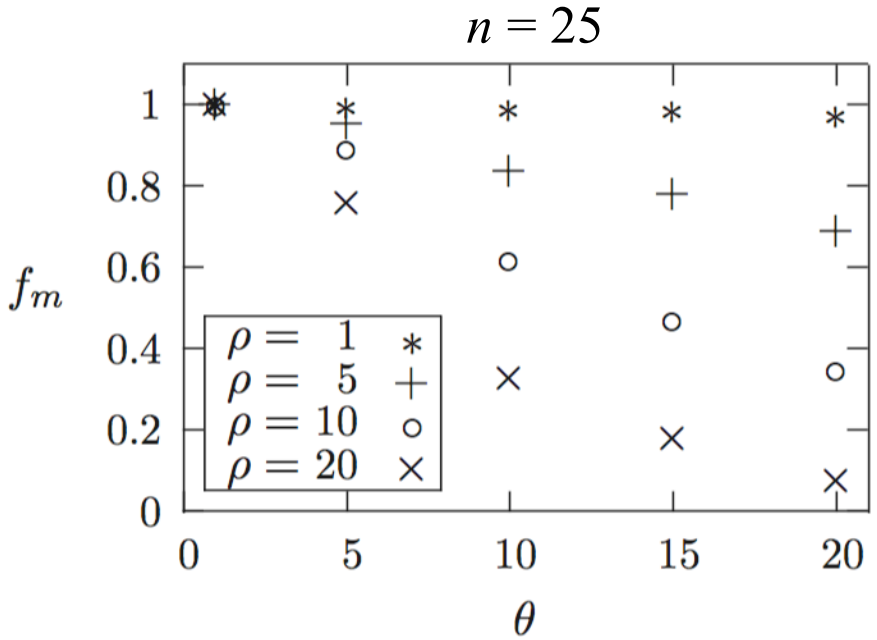
	site regions		
Population	reg 1	reg 2	reg 3
Jackson	10	9	12
N. Karelia	2	13	7
Rochester	1	12	7
All	12	21	22

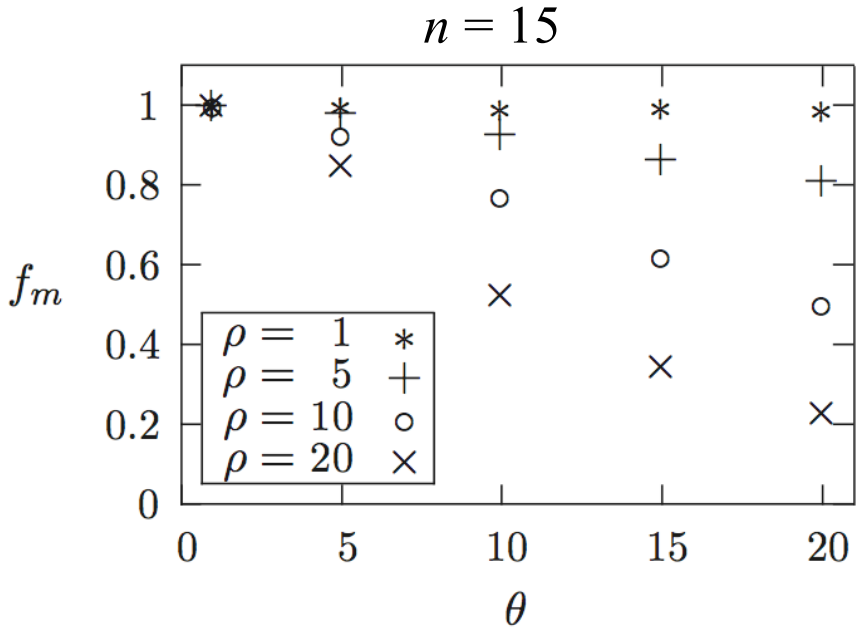
(We ignored insertion/deletion, unphased sites, and sites with missing data.)

Study on simulated data:
Exact-Match frequency for varying parameters

- θ = Scaled mutation rate
- ρ = Scaled recombination rate
- n = Number of sequences

Used Hudson's MS to generate1000 simulated datasets for each pair of θ and ρ.





For $\theta, \rho < 5$, our lower and upper bounds match more than 95% of the time.

Part III: Applications

Uniform Sampling of Min ARGs

- Sampling of ARGs: useful in statistical applications, but thought to be very challenging computationally. How to sample uniformly over the set of Min ARGs?
- All-visible ARGs: A special type of ARG
  - Built with only the input sequences
  - An all-visible ARG is a Min ARG
- We have an $O(2^n)$ algorithm to sample uniformly from the all-visible ARGs.
  - Practical when the number of sites is small
- We have heuristics to sample Min ARGs when there is no all-visible ARG.

Application: Association Mapping

- Given *case-control* data M, uniformly sample the minimum ARGs (in practice for small windows of fixed number of SNPs)
- Build the ``marginal" tree for each interval between adjacent recombination points in the ARG
- Look for non-random clustering of cases in the tree; accumulate statistics over the trees to find the best mutation explaining the partition into cases and controls.

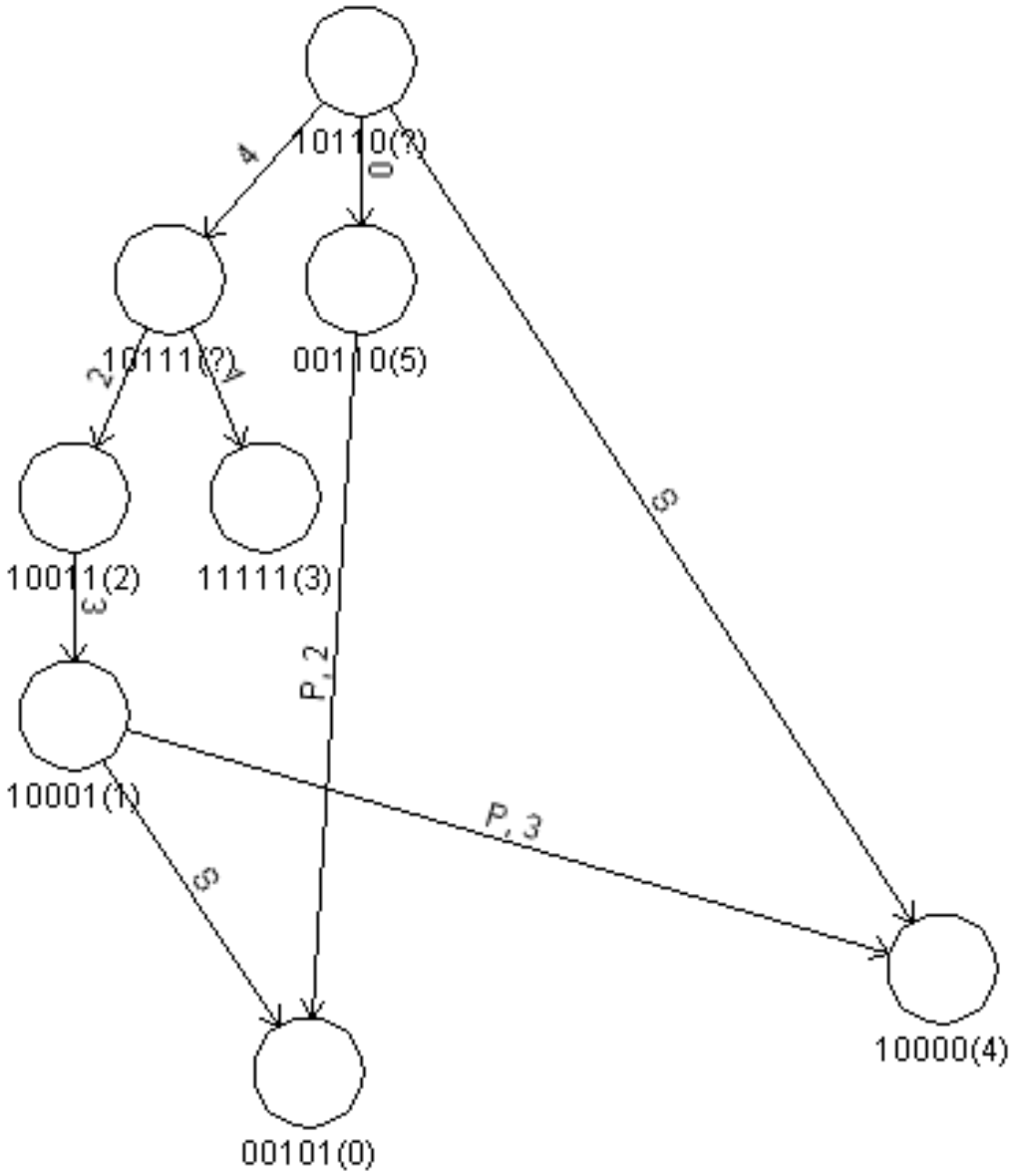
One Min ARG for the data

Input Data

00101
10001
10011

11111
10000
00110

sample



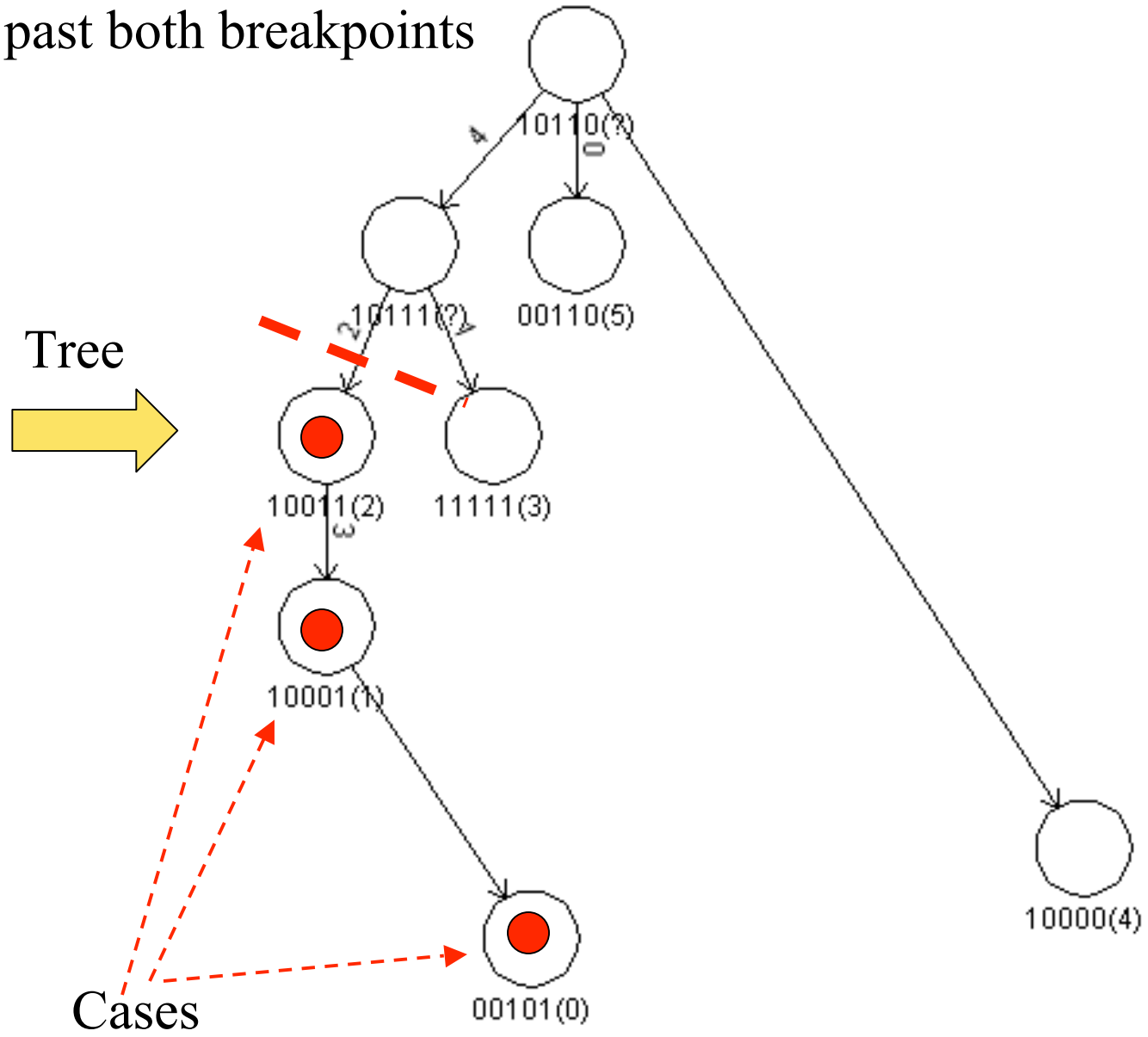
Seqs 0-2: *cases*
Seqs 3-5: *controls*

The marginal tree for the interval past both breakpoints

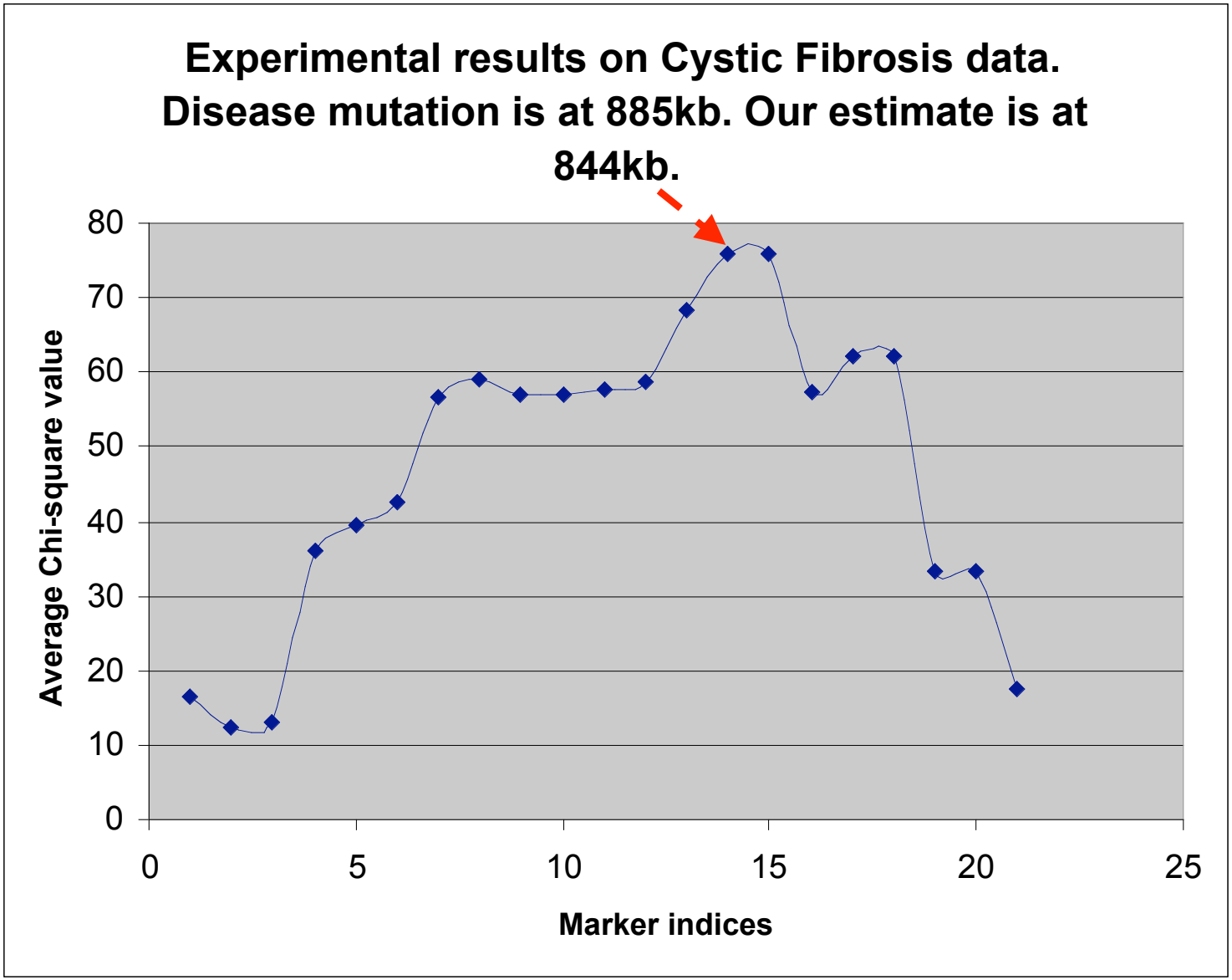
Input Data

00101
10001
10011
11111
10000
00110

Tree



Seqs 0-2: cases
Seqs 3-5: controls


Experimental results on Cystic Fibrosis data.
Disease mutation is at 885kb. Our estimate is at 844kb.
Average Chi-square value
0
10
20
30
40
50
60
70
80
0
5
10
15
20
25
Marker indices

Haplotyping (Phasing) genotypic data using a Min ARG

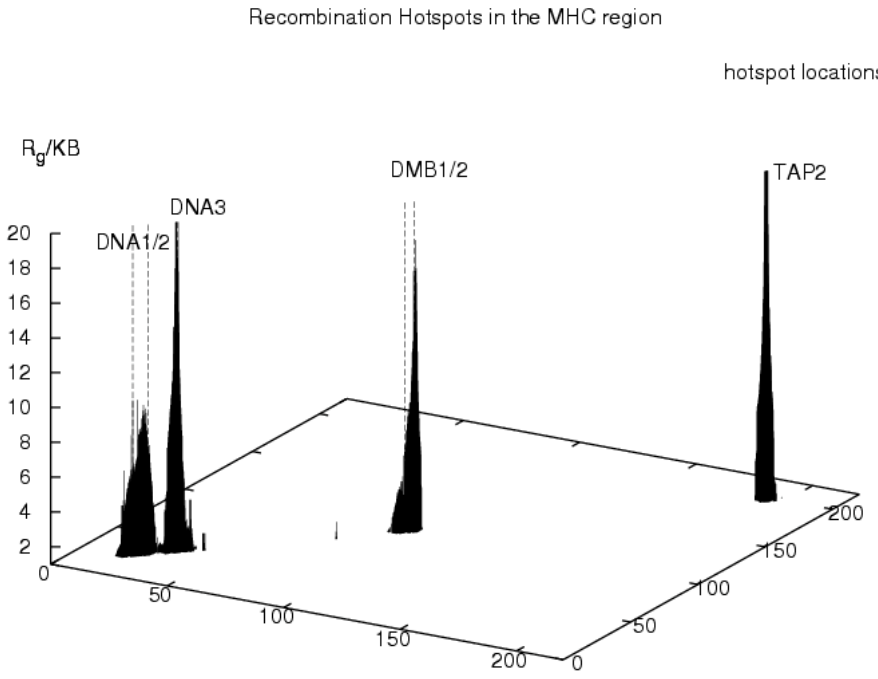
Minimizing Recombinations for Genotype Data

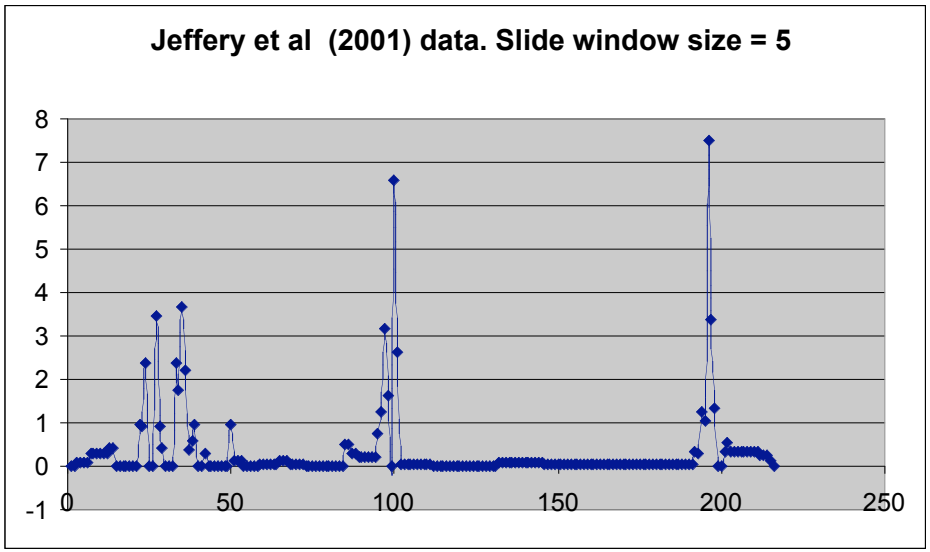
- Haplotyping (phasing genotypic data) via a Min ARG: attractive but difficult
- We have a branch and bound algorithm that builds a Min ARG for deduced haplotypes that generate the given genotypes. Works for genotype data with a small number of sites, but a larger number of genotypes.

Application: Detecting Recombination Hotspots with Genotype Data

- Bafna and Bansel (2005) uses recombination lower bounds to detect recombination hotspots with *haplotype* data.
- We apply our program on the *genotype* data
  - Compute the minimum number of recombinations for all small windows with fixed number of SNPs
  - Plot a graph showing the minimum level of recombinations normalized by physical distance
  - Initial results shows this approach can give good estimates of the locations of the recombination hotspots

Recombination Hotspots on Jeffreys, et al (2001) Data





Result from Bafna and Bansel (2005), **haplotype** data

Our result on **genotype** data

Application: Missing Data Imputation by Constructing near-optimal ARGs

For $\rho = 5$.

Datasets with about 1,000 entries

#Seq	#Sites	%missing	Accuracy
20	50	5 %	96 %
20	50	10 %	95 %
20	50	30 %	93 %
32	32	5 %	97 %
32	32	10 %	96 %
32	32	30 %	94 %
50	20	5 %	97 %
50	20	10 %	96 %
50	20	30 %	94 %

Dataets with about 10,000 entries

#Seq	#Sites	%missing	Accuracy
20	100	5 %	95 %
20	100	10 %	95 %
20	100	30 %	93 %
45	45	5 %	98 %
45	45	10 %	97 %
45	45	30 %	96 %
100	20	5 %	97 %
100	20	10 %	96 %
100	20	30 %	95 %

Haplotyping genotype data via a minimum ARG

- Compare to program PHASE, speed and accuracy: comparable for certain range of data
- Experience shows PHASE may give solutions whose recombination is close to the minimum
  - Example: In all solutions of PHASE for three sets of case/control data from Steven Orzack, recombinatons are minimized.
  - Simulation results: PHASE's solution minimizes recombination in 57 of 100 data (20 rows and 5 sites).

Algorithms to Distinguish the Role of Gene-Conversion from Single-Crossover Recombination in Populations

Y. Song, Z. Ding, D. Gusfield, C. Langley, Y. Wu

U.C. Davis

Reconstructing the Evolution of SNP (binary) Sequences

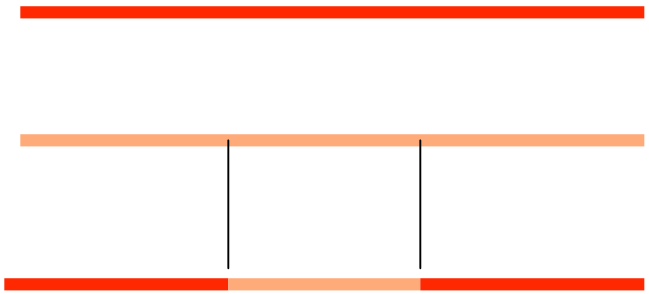
Ancestral sequence all-zeros. Three types of changes in a binary sequence:

1) Mutation: state 0 changes to state1 at a single site. At most one mutation per site in the history of the sequences. (Infinite Sites Model)

2) Single-Crossover (SC) recombination between two sequences.

3) Gene-Conversion (GC) between two sequences.

Gene Conversion



two-crossovers; two breakpoints

conversion tract

Gene Conversion (GC)

- ``Gene Conversion'' is a short two cross-over recombination that occurs in meiosis; length of the conversion tract 300 - 2000 bp.

- The extent of gene-conversion is only now being understood, due to prior lack of fine-scale molecular data, and lack of algorithmic tools. But more common than single-crossover recombination.

- Gene Conversion may be the Achilles heel of fine-scale association (LD) mapping methods. Those methods rely on monotonic decay of LD with distance, but with GC the change of LD is non-monotonic.

GC a problem for LD-mapping?

"Standard population genetics models of recombination generally ignore gene conversion, even though crossovers and gene conversions have different effects on the structure of LD." J. D. Wall

See also, Hein, Schierup and Wiuf p. 211 showing non-monotonicity.

Focus on Gene-Conversion

We want algorithms that identify the signatures of gene-conversion in SNP sequences in populations; that can quantify the extent of gene-conversion; that can distinguish GC signatures from SC signatures.

The methods parallel earlier work on networks with SC recombination, but introduce additional technical challenges.

Three types of results

- Algs. to compute lower bounds on the minimum total number of recombinations (SC + GC) needed to generate a set of sequences (with bounded and unbounded tract-length).

- Algs. to construct networks that generate the sequences with the minimum total number of recombinations, or to upper bound the min.

- Tests to distinguish the role of SC from GC.

Applications First

Assume we can compute reasonably close upper and lower bounds. How are they used?

(Naïve) Approach to Distinguish GC from SC

For a given set of sequences, let B(t) be the bound (lower or upper) on the minimum total number of recombination (SC + GC), when the tract-length is at most t.

So B(0) is the case when only single-crossovers are allowed.

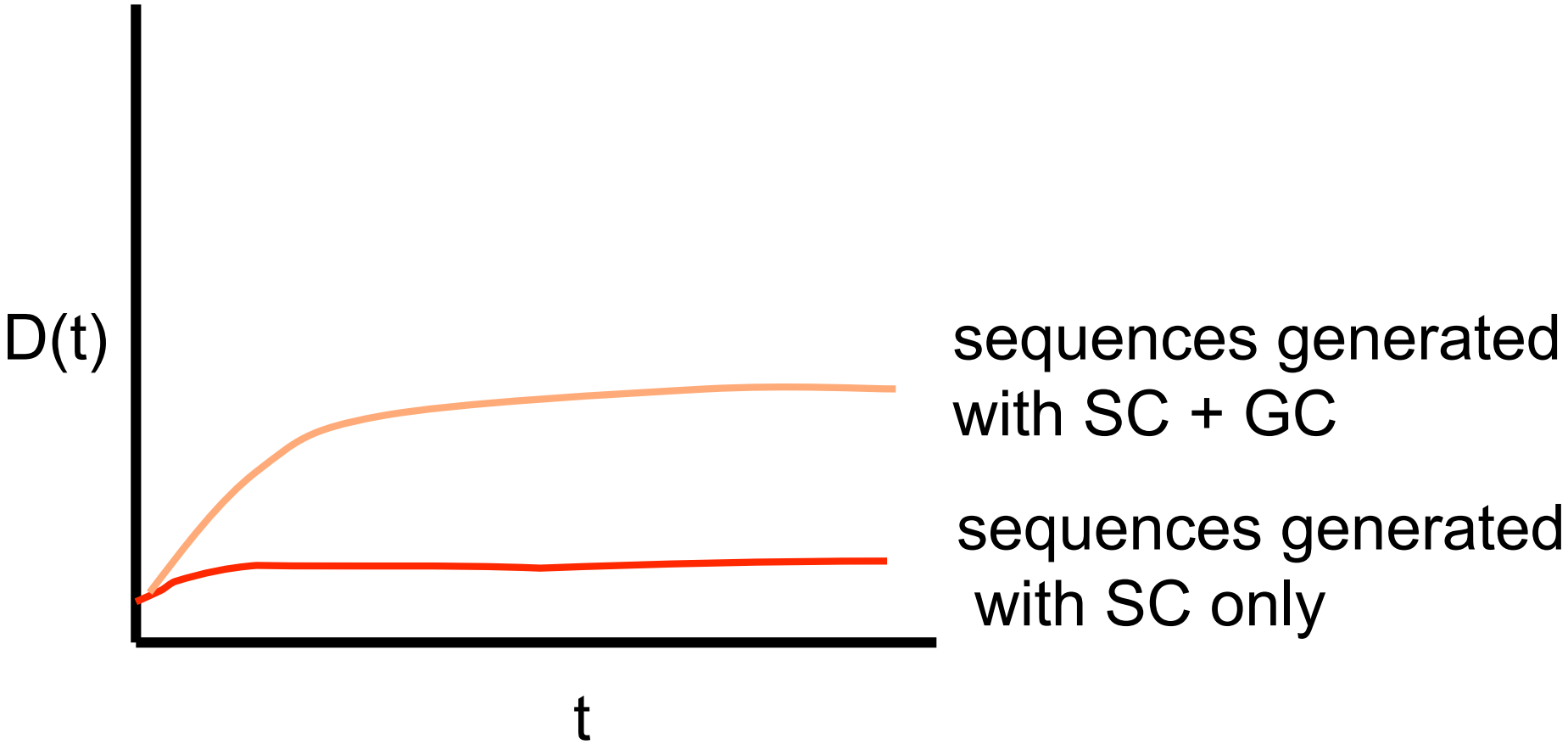
Note that B(t) <= B(0) and B(t) "decreases" with t.

Define D(t) = B(0) - B(t). D(t) "increases" with t.

We expect that D(t) will be larger and will grow faster when the sequences are generated using gene-conversion and crossovers compared to when they are generated with crossovers only.

And we expect that D(t) will be convex in simulations where GC tract-length is chosen from a geometric distribution - at some point past the mean tract length, larger t does not help reduce B(t).

$$D(t) = B(0) - B(t)$$



Naïve expectation

Actually, we compute the minimum number of GCs, call it GC(t), among all solutions that use B(t) total recombinations. Then we take the ratio GC(t)/B(t). The ratio indicates the

relative importance of GCs in the bound.

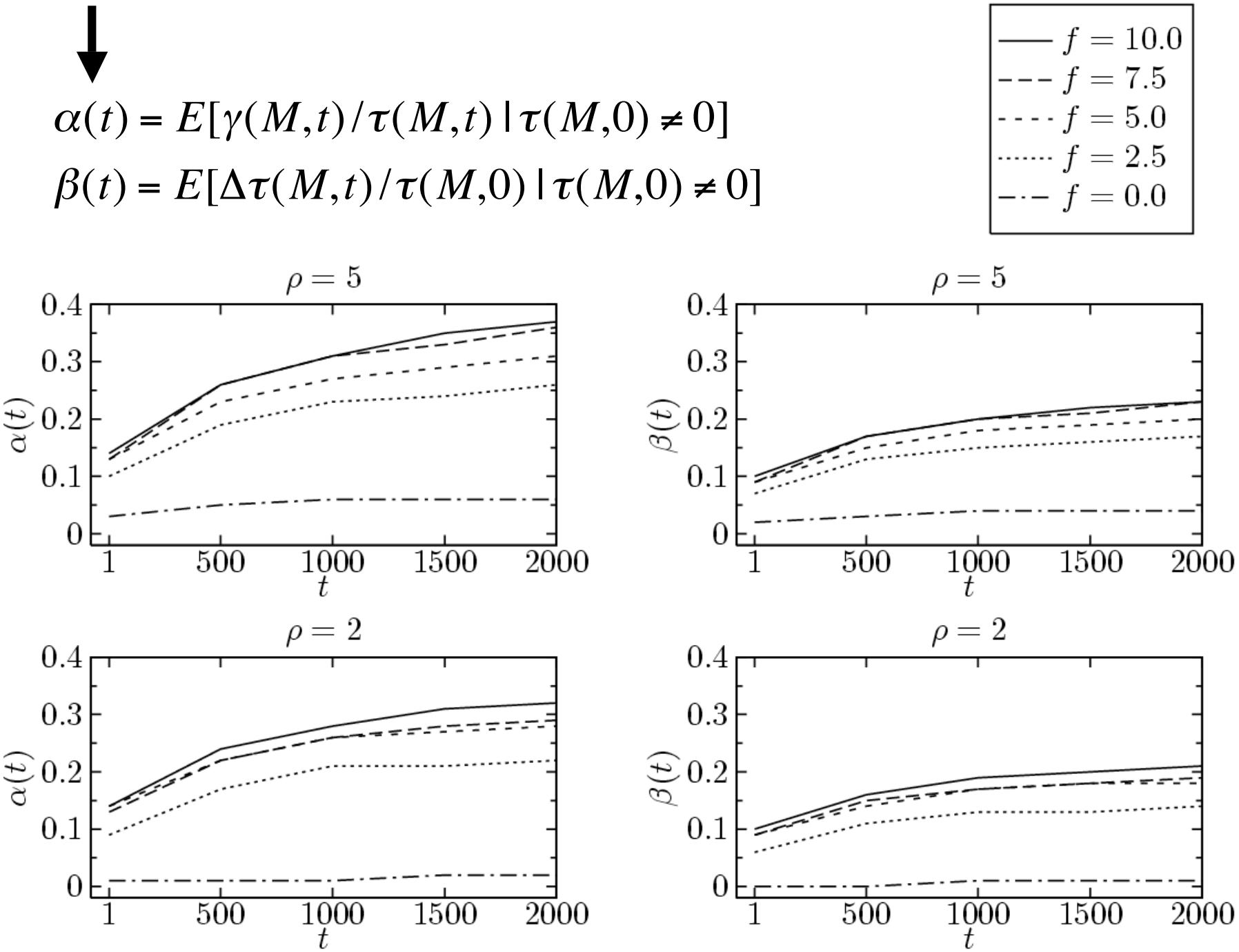
Results for average GC(t)/B(t):

1) Little change (as a function of t) for sequences generated with SC only.

2) Ratio increase with t for sequences generated with GC also, and the difference is greater when more GCs were used to generate the sequences.

$$\alpha(t) = E[\gamma(M,t)/\tau(M,t) \mid \tau(M,0) \neq 0]$$

$$\beta(t) = E[\Delta\tau(M,t)/\tau(M,0) \mid \tau(M,0) \neq 0]$$



Take-home message

The upper and lower bound algorithms cannot ``make-up'' gene-conversions.

The ability to use GCs in computing upper and lower bounds doesn't help much unless the sequences were actually generated with GCs.

Gene-Conversion Presence Test

The results just shown are averages.

Unfortunately, the variance is large, so we need a different test on any single data set. The simplest is whether $GC(t) > 0$ for a given t.

That is, in order for the algorithm to get the best bound it can, are some GC's needed? GC(t) can be based either on upper or lower bounds or we can require both be non-zero - which is what we do.

It works, pretty well. Extreme examples

1. Recombination rate, 5; no gene-conversion, percent of data passing test 9.6 % (false positive).

- Recombination rate 5, gene-conversion ratio f = 10 (high gene conversion), percent of simulated data passing test 95.8%.
- Both test use upper and lower bounds.

Gene-Conversions in *Arabidopsis thaliana*

- 96 samples, broken up into 1338 fragments (Plagnol, Norberg *et al.*, Genetics, in press)
- Each fragment is between 500 and 600 bps.
- Plagnol et al. identified four fragments as containing clear signals for gene-conversion.

  Essentially, they found fragments where exactly one recombination is needed, but it must be a GC.
- In contrast, 22 fragments passed our test: $GC(t) > 0$.
- Of these 22 fragments, three coincided with those found by Plagnol et al.

Lower Bounds: Review of composite methods for SC (S. Myers, 2003)

- Compute local lower bounds in (small) overlapping intervals. Many types of local bounds are possible.

- Compose the local bounds to obtain a global lower bound on the full data.

Example: Haplotype Local Bound (Myers 2003)

- Rh = Number of distinct sequences (rows) - Number of distinct sites (columns) -1 <= minimum number of recombinations (SC) needed.

- The key to proving that Rh is a lower bound, is that each recombination can create at most one new sequence. This holds for both SC and GC.

The better Local Bounds

- haplotype, connected component, history, ILP bounds, galled-tree, many other variants.

- Each of the better local bounds for SC also hold for both SC and GC. Different justifications for different bounds.

- Some of the local bounds are bad, even negative, when used on large intervals, but good when used as on small intervals, leading to very good global lower bounds, with a sufficient number of sites.

Composition of local bounds

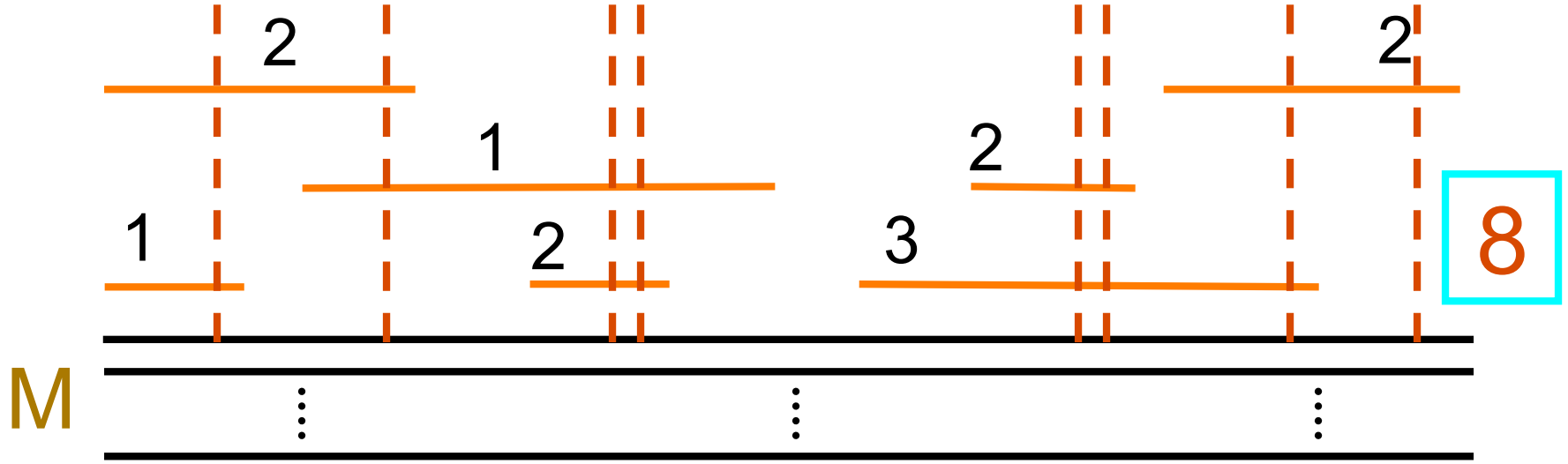
Given a set of intervals on the line, and for each interval I, a local bound N(I), define the composite problem: Find the minimum number of vertical lines so that every interval I intersects at least N(I) of the vertical lines. The result is a valid global lower bound for the full data.

The composite problem is easy to solve by a left-to-right myopic placement of vertical lines.

The Composite Method (Myers & Griffiths 2003)

1. Given a set of intervals, and
2. for each interval **I**, a number **N(I)**

Composite Problem: Find the minimum number of vertical lines so that every **I** intersects at least N(I) vertical lines.



Trivial composite bound on SC + GC

If L(SC) is a global lower bound on the number of SC recombinations needed, obtained using the composite method, then the total number of SC + GC recombinations is at least L(SC)/2.

Can we get higher lower bounds for SC + GC using the composition approach?

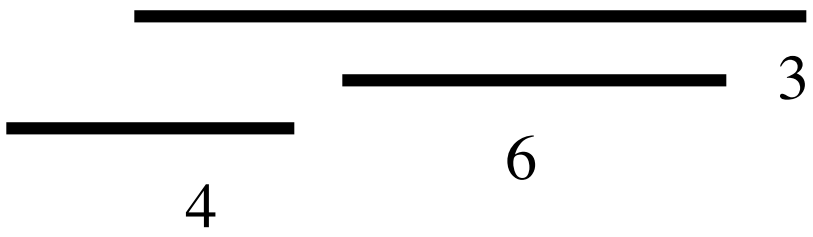
Extending the Composite Method to Gene-Conversion

- All previous methods for local bounds also provide lower bounds on the number of SC + GC recombinations in an interval.

- Problem: How to compose local bounds to get a global lower bound for SC + GC?

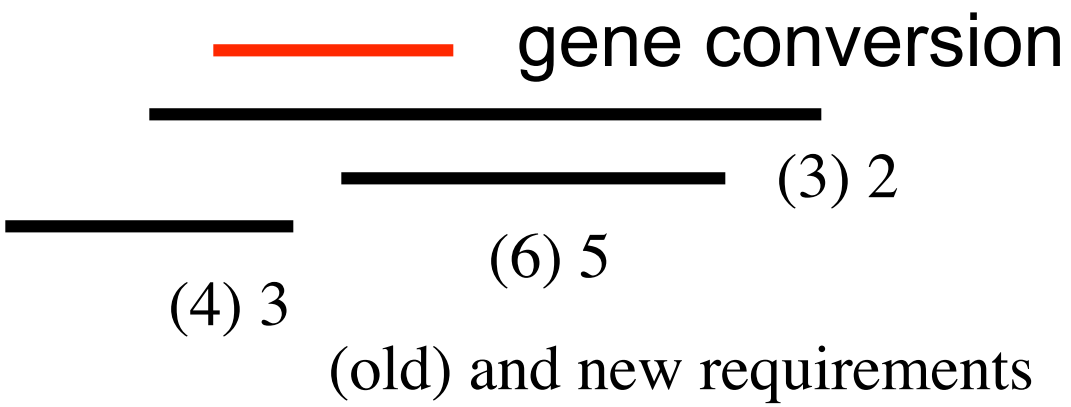
How composition with GC differs from SC

A single gene-conversion counts as a recombination in every interval containing a breakpoint of the gene-conversion.



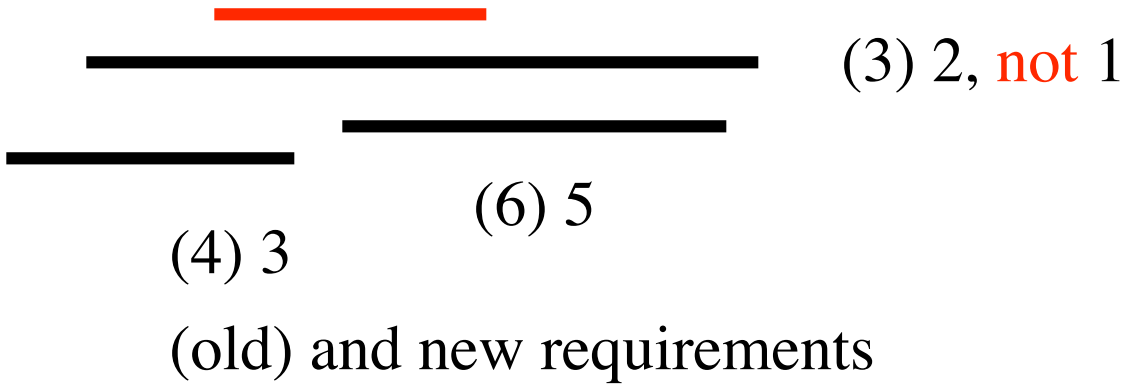
local bounds

So one gene-conversion can sometimes act like two single-crossover recombinations:



However …

A GC never counts as two recombinations in any single interval, even if it contains both breakpoints.



(old) and new requirements

The reason depends on the particular local bound.

The reasons depend on the specific local bound. For example, the haplotype bound for SC is based on the fact that a single crossover in an interval can create one new sequence. However, two crossovers in the interval, from the same GC, can also only create one new sequence.

Composition Problem with GC

Definition: A point p covers an interval I if p is contained in I. A line segment, s, covers I if one or both of the endpoints of s are contained in I.

Problem: Given intervals I with local bounds N(I), find the minimum number of points, P, and line segments S, so that each I is covered at least N(I) times by P U I. The result is a lower bound on the minimum number of SC + GC.

The Hope

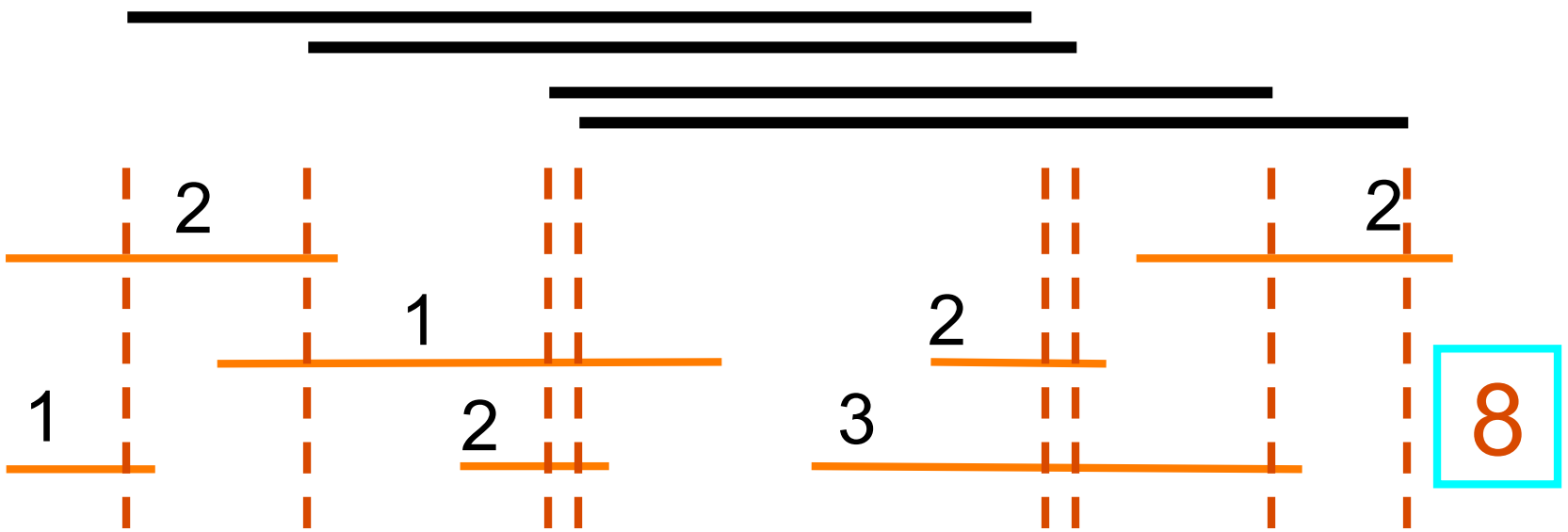
Because of combinatorial constraints, we hope(d) that not every GC could replace two SC recombinations, so that the resulting global bound would be greater than the trivial L(SC)/2.

Unfortunately …

Theorem: If L(SC) is the lower bound obtained by the composite method for SC only, and the tract length of a GC is unconstrained, then it is always possible to cover the intervals with exactly

Max [L(SC)/2, $\max_I$ N(I)] points and line segments.

So, with unconstrained tract length, we essentially can only get trivial lower bounds (wrt L(SC)) using the composite method, but those bounds can be computed efficiently.



Four gene-conversions suffice in place of 8 SCs.
The breakpoints of the GCs align with the SCs.

How to beat the trivial bounds

- Constrain the tract length. Biologically realistic, but then the composition problem is computationally hard. It can be effectively solved by a simple ILP formulation.

- Encode combinatorial constraints that come from GC but not SC.

Lower Bounds with bounded tract length t

- Solve the composition problem with ILP. Simple formulation with one variable K(p,q) for every pair of sites p,q with the permitted length bound. K(p,q) indicates how many GCs with breakpoints p,q will be selected.
- For each interval I,

  Σ [k(p,q)] >= N(I), for p or q in I

“Four-Gamete” Constraints on Composition

a b c	All three intervals [a,b], [a,c]
0 0 0	and [b,c] have (haplotype) local
0 0 1	bound of 1, and a single GC
1 1 0	covers these local bounds.
1 0 1	But the pair a,c have all four

binary combinations, and no single GC with both breakpoints in [a,c]

can generate those four combinations. So more constraints can be added to the ILP that raise the lower bound. New constraints for every “incompatible” pair of sites.

Constructing Optimal Phylogenetic Networks

Optimal = minimum number of recombinations. Called Min ARG.

The method is based on the coalescent viewpoint of sequence evolution. We build the network backwards in time.

Definition: A column is non-informative if all entries are the same, or all but one are the same.

The key tool

- Given a set of rows A and a single row r, define w(r | A - r) as the minimum number of recombinations needed to create r from A-r (well defined in our application).
- w(r | A-r) can be computed in polynomial time by an algorithm recently published by N. Mabrouk et al.

Upper Bound Algorithm

1) Set W = 0
2) Collapse identical rows together, and remove non-informative columns. Repeat until neither is possible.
3) Let A be the data at this point. If A is empty, stop, else remove some row r from A, and set W = W + W(r | A-r). Go to step 2).

Note that the choice of r is arbitrary in Step 3), so the resulting W can vary.

An execution gives an upper bound W and specifies how to construct a network that derives the sequences using exactly W recombinations.

Each step 2 corresponds to a mutation or a coalescent event; each step 3 corresponds to a recombination event.

We can find the lowest possible W with this approach in O(2^n) time by using Dynamic Programming, and build the Min ARG at the same time.

In practice, we can use branch and bound to speed up the computation, and we have also found that branching on the best local choice, or randomizing quickly builds near-optimal ARGs.

Program: SHRUB-GC

Papers and
Software on wwwcsif.cs.ucdavis.edu/~gusfield