Integer Programming for Phylogenetic Network Problems

D. Gusfield
University of California, Davis
Presented at the National University of Singapore, July 27, 2015.

There are many important phylogeny problems that depart from simple tree models:

- Missing entries
- Data generated by complex biology, such as recombination or recurrent mutation
- 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 simpler data.

Question

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

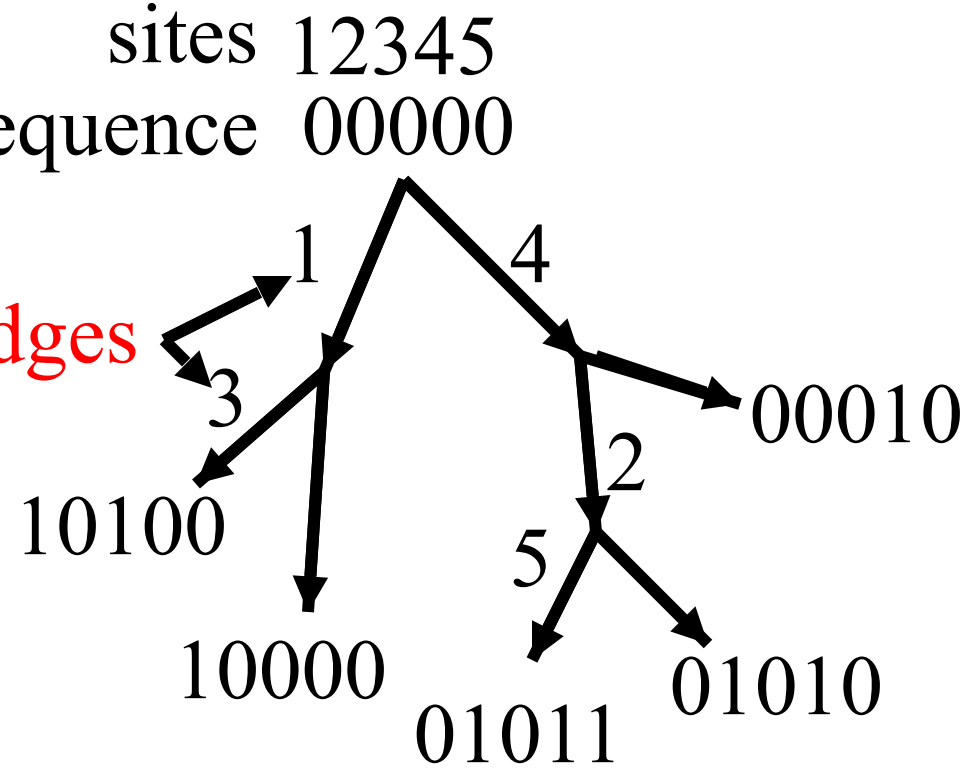
We have recently developed ILPs for many such problems and intensively studied their performance (speed, size and biological utility).

In this talk I will concentrate on
ILP problems relating to networks caused by
back mutation and recombination.

We start with the Perfect-Phylogeny Model, which is the case when neither back mutation or recombination are allowed.

Starting Model: Perfect Phylogeny (infinite sites) model for 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

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 three binary pairs:

0,1 and 1,0 and 1,1

This is the 3-Gamete Test.

Each binary pair is called a gamete.

A pair of sites that has all three gametes is called incompatible.

Problem MD: Missing Data

Given ternary sequences (0s, 1s, ?s), change the ?s to 0s and 1s in order to minimize the resulting number of incompatible pairs of sites. NP-hard.

(Special case) Perfect Phylogeny with Missing Data: Determine if the ?s can be set so that there are no resulting incompatibilities. In the case that the root is all-zero, there is an elegant poly-time solution (Pe'er, Sharan, Shamir), but we will not use it. Instead we use ILP, because it can be extended to other problems.

Simple ILP for the Missing Data problem

Create a binary variable Y(i,p) for a ? in cell (i,p), indicating whether the cell 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.

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

and create inequalities to set it 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 at site 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.

Or, we could require that the sum of the C(p,q) variables be zero, and then there is a way to set the missing values to form a Perfect Phylogeny, if and only if the ILP is feasible.

Empirically these ILPs solve very quickly, in fractions of seconds or seconds for n and m up to hundreds of rows and columns.

The software for to create the ILP formulations was written in 2006, but is paying dividends now.

Persistent and Dollo: Deviations from Perfect Phylogy

- Extends the Perfect Phylogeny Model by allowing each site to revert from state 1 to state 0.
- Persistent Phylogeny: Only once in the tree. So this is like the infinite sites model in for both forward and backward mutations.
- Dollo Model: Forward mutation once, but backwards any number of times.

A range of possibilities

- So given binary data either it can be generated on a Perfect Phylogeny, or a Galled-Tree, or a Persistent Phylogeny, or a Dollo Phylogeny, or none of the above - i.e., a more general network is needed.
- Given binary data, how do we determine what case we have?

The Dollo model was introduced more than 100 years ago, but the persistent phylogeny model was only introduced recently, by T. Przytycka and D. Durand, has been studied intensively by P. Bonizzoni and co-authors.

The Persistent Phylogeny Problem: Given M, determine if M can be derived on a Persistent Phylogeny.

The question of whether the Persistent Phylogeny Problem is NP-hard is open. So, we take an ILP approach.

The Persistent Phylogeny Problem

- The key to the ILP for it, is the following formalism developed by P. Bonizzoni et al. in 2013.

Definition: Given a binary matrix M, the extended matrix Me contains two columns, j1 and j2, for each column j in M.

Column j1 of Me is derived from column j in M by replacing every occurrence of `0' in column j of M with `?' in column j1 of Me.

Column j2 of Me is derived from column j1 by replacing every occurrence of `1' in j1 with `0'.

So a 0 in j becomes ??, and a 1 becomes 10.

Completing Me

A completion M'e of Me changes each ?' to either 0 or 1, with the requirement that for every pair of sites (j1, j2) in Me that originated from an entry of value 0 in cell (i,j) in M, cells (i, j1) and (i, j2) in M'e must get the same value, i.e., they either get 0,0 or 1,1.

Extension Me and Completion M'e of M

M	Me	M'e
1110	101010??	10101000
0111	??101010	11101010
0000	????????	00000000
1010	10??10??	10001000
1100	1010????	10101100
1111	10101010	10101010

For character j in M, character j1 in Me is ``a mutation of character j has occurred", and character j2 is ``a back mutation of character j has occurred".

Theorem of Bonizzoni et al.

M can be represented by a Persistent Phylogeny if and only if there is a completion M'e of Me that is a Perfect Phylogeny. And if so, the perfect phylogeny for M'e is a Persistent Phylogeny for M.

This theorem shows the way to formulate the ILP for the Persistent Phylogeny problem.

The ILP

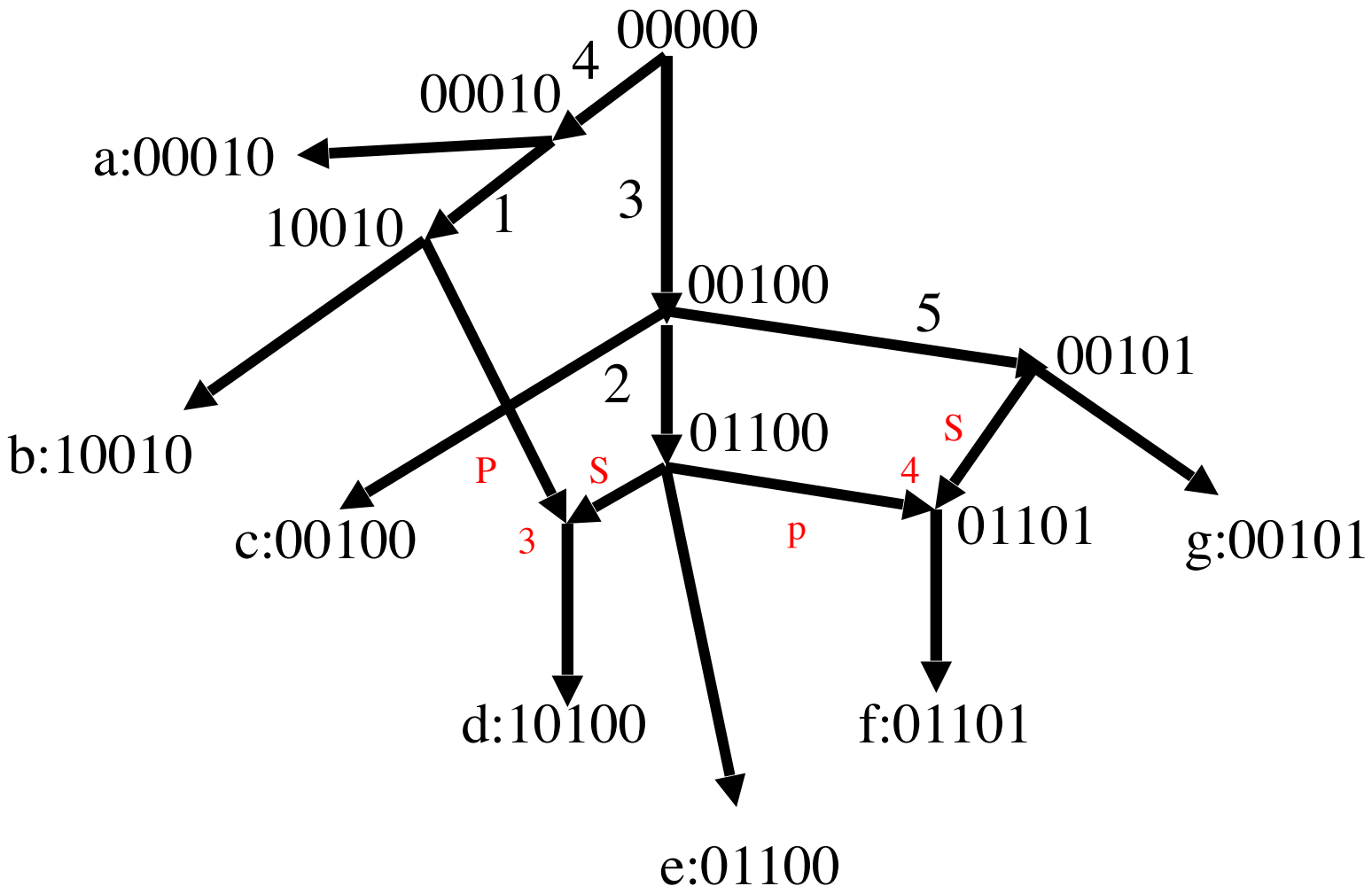
Given M, we form Me and treat that as input to problem MD, but for every pair of sites (j1, j2) in Me that originated from an entry of value 0 in cell (i, j) in M, we add the constraint: Y(i,j1) = Y(i,j2).

Then the ILP has optimal value zero if and only if M has a persistent phylogeny.

ARGs and Galled-Trees

- ARGs incorporate recombination, but still only allow one mutation per site in the ARG.
- The problem of finding the ARG for M that minimizes the number of recombination nodes is NP-hard.

ARG


00000
4
00010
a:00010
1
10010
3
00100
5
00101
2
b:10010
01100
S
P
S
4
c:00100
3
p
01101
g:00101
d:10100
f:01101
e:01100

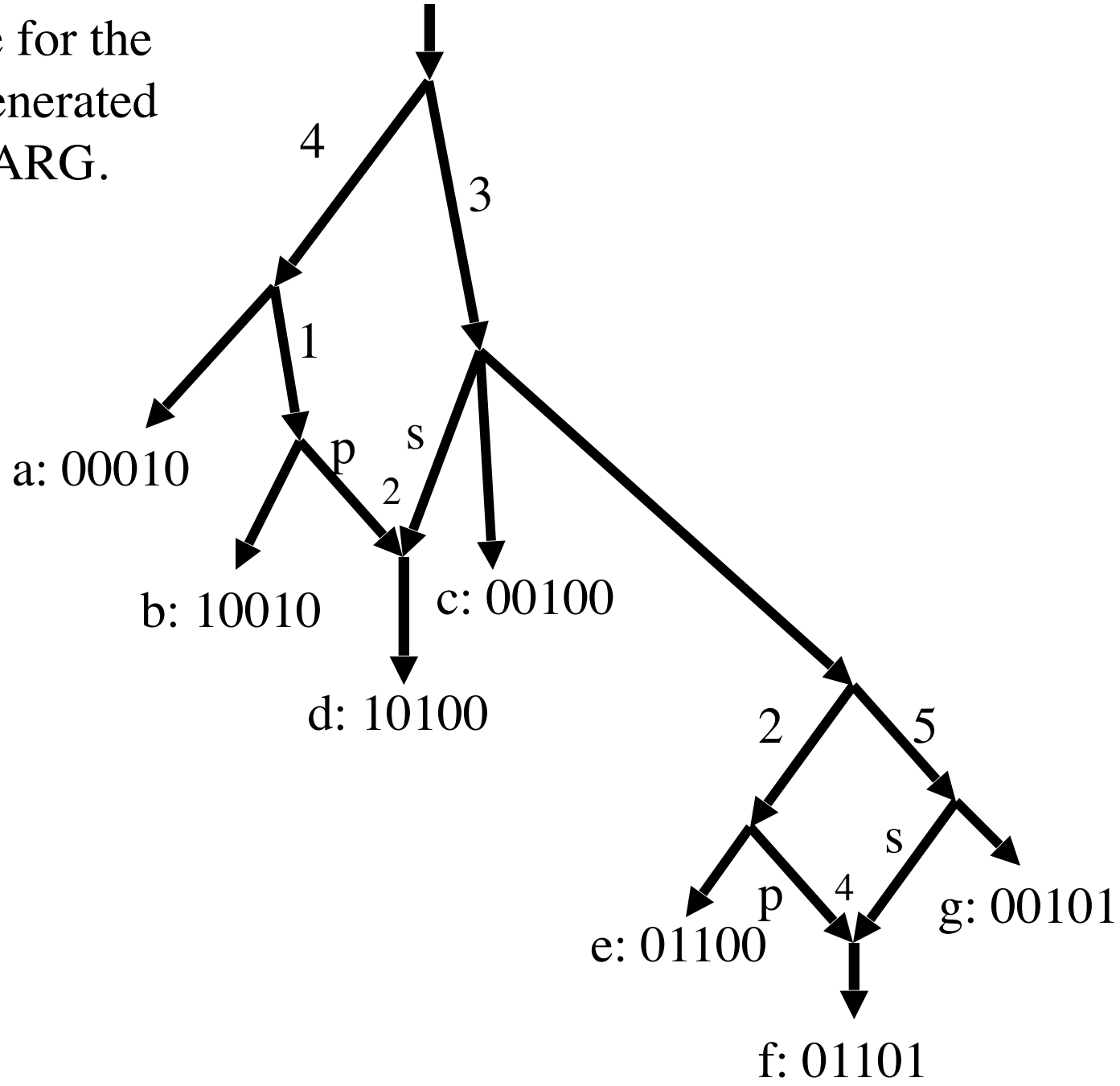
Recombination Cycles

- In an ARG, 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”.

Allowing Limited Recombination: Galled-Trees

- An ARG where no recombination cycles share an edge is called a galled tree.
- A cycle in a galled-tree is called a gall.

A galled-tree for the sequences generated by the prior ARG.



Galled-Trees and Persistent Phylogeny

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

- Practical software exists to determine if M can be derived on a galled tree.

- Theorem: If M can be derived on a galled tree, it can be derived on a Persistent Phylogeny. Hence, this is a special case where the Persistent Phylogeny problem can be solved in polynomial time.

Empirical Results: It Works!

r; c	br	data types perf; gt; pers	conflicts gt; pers	ILP-data inf; int; tm-gt; tm-pers
40, 30	0.02	39, 10, 1	2.29, 3	0, 0, 0, 0
40, 30	0.05	29, 17, 4	1.7, 3.75	0, 0, 0, 0
40, 30	0.1	20, 23, 7	1.73, 5	0, 0, 0, 0.01
40, 30	0.2	7, 21, 22	1.95, 6.13	0, 0, 0, 0
40, 100	0.02	12, 17, 21	7.35, 12.85	0, 0, 0.01, 0.04
40, 100	0.05	3, 6, 41	4.66, 24.56	0, 0, 0, 0.16
40, 100	0.1	0, 0, 50	0, 33.15	0, 0, 0, 0.34
40, 100	0.2	0, 0, 50	0, 48.7	0, 0, 0, 0.62
60, 30	0.02	39, 10, 1	1.8, 3	0, 0, 0, 0
60, 30	0.05	26, 17, 7	2.11, 4.28	0, 0, 0, 0
60, 30	0.1	21, 22, 7	1.63, 4.71	0, 0, 0, 0
60, 30	0.2	12, 11, 27	1.81, 5.03	0, 0, 0, 0.01
60, 100	0.02	11, 23, 16	4.56, 13.56	0, 0, 0, 0.06
60, 100	0.05	1, 6, 43	6, 18.83	0, 0, 0.01, 0.13
60, 100	0.1	0, 2, 48	5.5, 36.29	0, 0, 0.03, 0.47
60, 100	0.2	0, 1, 49	12, 56.04	0, 0, 0.1, 0.95
100, 30	0.02	41, 8, 1	2.62, 5	0, 0, 0, 0

Table : Table for 40, 60, 100 and 150 taxa representable by a persistent phylogeny. All times are in seconds.

r; c	br	data types perf; gt; pers	conflicts gt; pers	ILP-data inf; int; tm-gt; tm-pers
200, 100	0.02	6, 0, 44	0, 11.45	0, 0, 0, 0.04
200, 100	0.05	1, 0, 49	0, 21.69	0, 0, 0, 0.22
200, 100	0.1	0, 0, 50	0, 34.86	0, 0, 0, 0.64
200, 100	0.2	0, 0, 50	0, 44.34	0, 0, 0, 0.93
200, 200	0.02	0, 0, 50	0, 54.9	0, 0, 0, 1.33
200, 200	0.05	0, 0, 50	0, 101.78	0, 0, 0, 3.67
200, 200	0.1	0, 0, 50	0, 138.56	0, 0, 0, 9.42
200, 200	0.2	0, 0, 50	0, 173.78	0, 0, 0, 13.67
200, 250	0.02	0, 0, 50	0, 86.86	0, 0, 0, 4.01
200, 250	0.05	0, 0, 50	0, 181.5	0, 0, 0, 11.67
200, 250	0.1	0, 0, 50	0, 266.95	0, 0, 0, 20.42
200, 250	0.2	0, 0, 50	0, 350.8	0, 0, 0, 38.34
400, 200	0.05	0, 0, 50	0, 98.26	0, 0, 0, 4.37
400, 200	0.1	0, 0, 50	0, 172.72	0, 0, 0, 10.15
400, 200	0.2	0, 0, 50	0, 186.36	0, 0, 0, 16.49
400, 300	0.05	0, 0, 50	0, 277.72	0, 0, 0, 21.77
400, 300	0.1	0, 0, 50	0, 400.8	0, 0, 0, 47.98

Table : Table for 200 through 1000 taxa representable by a persistent phylogeny. Galled-tree computations were not run on these data, due to size limitations in the galled-tree program. All times are in seconds.

Next Topic: Computing the History Bound

1. The History Lower Bound

Given a set of sequences M, the *History Bound* of Myers and Griffiths is a lower bound on the number of needed recombinations in any ARG that creates M (with all-zero ancestral sequence).

It is defined only by the algorithms that compute it.

ARGs, Reticulation Networks, the History Bound and ILPs

- The history bound is the best available lower bound on the minimum number of recombinations needed in an ARG that derives M.
- But, the history bound takes exponential time to compute, as a function of the number of rows of M.
- Can an ILP approach do better?

The computation of the history bound uses three Rules

Initially, set $\tilde{M}$ to the input M. As the algorithm proceeds, rows and columns of M will be deleted. Let $\tilde{M}$ denote the current *remaining* submatrix of M as the algorithm executes. The algorithm executes three Rules.

> **Rule Dc:** *If a column c of $\tilde{M}$ contains at most one entry with value 1, then remove column c from $\tilde{M}$.*
>
> *Rule* **Dt:** *If neither Rule* **Dc** *nor* **Dr** *can be applied, pick a row $\tilde{r}$ in the current $\tilde{M}$ (other than the all-zero row that corresponds to the ancestral sequence) and remove row $\tilde{r}$ from $\tilde{M}$.*

Example

	1	2	3	4	5	6
r_1	0	0	1	0	0	0
r_2	0	1	1	0	1	0
r_3	0	1	0	0	1	0
r_4	1	0	0	0	0	1
r_5	1	0	0	1	0	1
r_6	0	0	0	1	0	0

Figure: The input M. No application of Rule **Dc** or **Dr** is possible. So pick a row, say r_6, for Rule **Dt**.

Example

	1	2	3	4	5	6
r_1	0	0	1	0	0	0
r_2	0	1	1	0	1	0
r_3	0	1	0	0	1	0
r_4	1	0	0	0	0	1
r_5	1	0	0	1	0	1

Figure: Now apply Rule **Dc** to column 4.

Example

	1	2	3	5	6
r_1	0	0	1	0	0
r_2	0	1	1	1	0
r_3	0	1	0	1	0
r_4	1	0	0	0	1
r_5	1	0	0	0	1

Figure: Now apply Rule **Dr**, and remove row r_5.

Example

	1	2	3	5	6
r_1	0	0	1	0	0
r_2	0	1	1	1	0
r_3	0	1	0	1	0
r_4	1	0	0	0	1

Figure: Now apply Rule **Dc** twice to remove columns 1 and 6.

Example

		2	3	5	
r_1		0	1	0	
r_2		1	1	1	
r_3		1	0	1	
r_4		0	0	0	

Figure: Now apply Rule **Dt** to remove row 4.

Example

		2	3	5	
r_1		0	1	0	
r_2		1	1	1	
r_3		1	0	1	

Figure: Now apply Rule **Dt** again to remove row 3.

Example

		2	3	5	
r_1		0	1	0	
r_2		1	1	1	

Figure: Now apply Rule **Dc** twice to remove columns 2 and 5.

Example

			3		
r_1			1		
r_2			1		

Figure: Now apply Rule **Dr** to remove row 2.

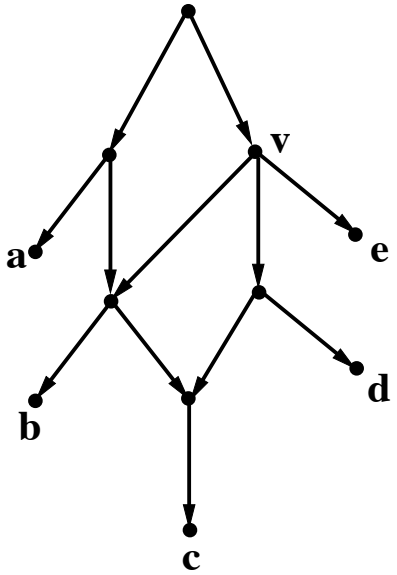
Example

			3		
r_1			1		

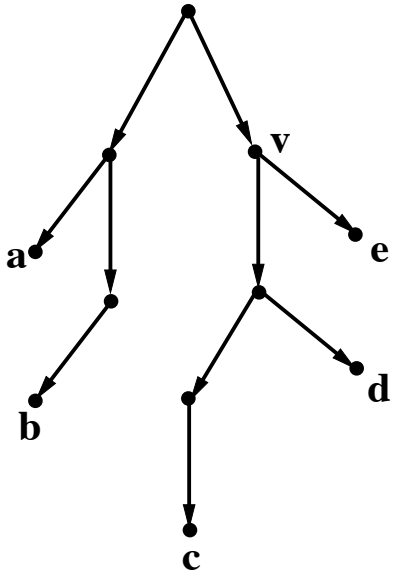
Figure: Now apply Rule **Dc** to remove column 3 to obtain a single row with no entries, so this execution of Algorithm *CHB* ends.

History Bound and Reticulation Networks

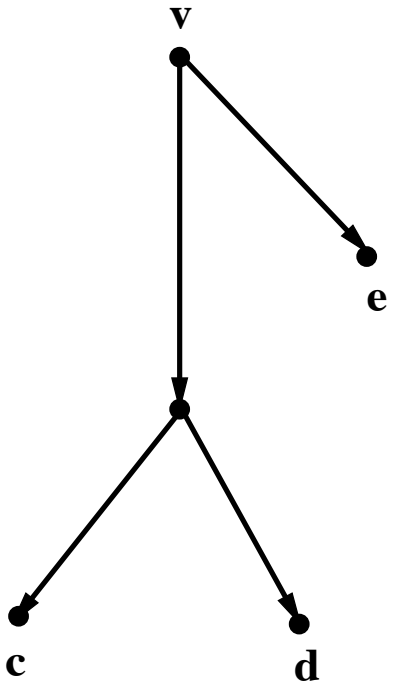
- Also, the history bound is equal to the minimum number of reticulation nodes needed in any reticulation network that derives M, in the soft-wired sense.
- So, the ILP for the history bound also solves this problem about reticulation networks.



(a) Reticulation network $\mathcal{D}$, with two reticulation nodes. Each leaf is labeled with the rows of input matrix M.



(b) A spanning subtree $T_{\mathcal{D}}$ of $\mathcal{D}$, displaying one split for each edge. For example, split $\{e, c, d\}$.



(c)

Figure 1: A column c of input data M is displayed in reticulation network DD (in the softwired sense) if there is a spanning tree T_{DD} of $\mathcal{D}$, and an edge (u, v) such that the removal of edge (u, v) separates the rows of c labeled 0 from the rows of c labeled 1.

We (Yufeng Wu and I) now have an ILP formulation for the history bound, and Julia Matsieva has implemented it, and is optimizing it. Currently, it is about as fast as the DP for computing the history bound. The DP does not scale. The goal is to compute history bound with the ILP for data with many more rows than can be handled by the DP.