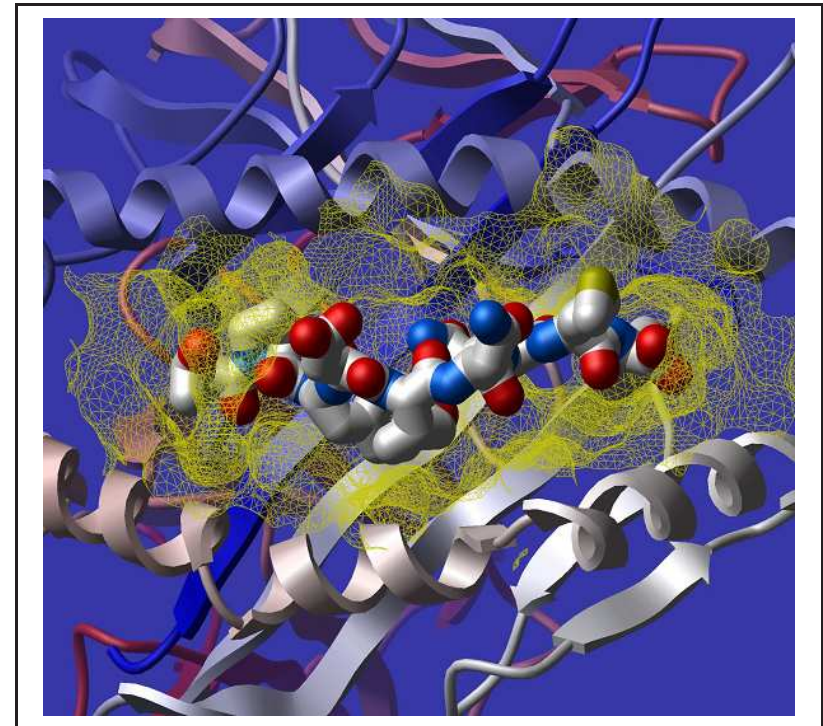


# Multiple Sequence Alignment

based on Ch. 6 from  
*Biological Sequence Analysis*  
by R. Durbin et al., 1998

Acknowledgements:  
M.Sc. student Diana Popovici  
M.Sc. student Oana Rățoi



[ MHC class I with peptide ]

MHC = Major Histocompatibility Complex

## PLAN

1. From profiles to Profile HMMs
2. Setting the parameters of a profile HMM;  
the optimal (MAP) model construction
3. Basic algorithms for profile HMMs
4. Profile HMM training from unaligned sequences:  
Getting the model and the multiple alignment simultaneously
5. Profile HMM variants for non-global alignments
6. Weighting the training sequences

## 1 From profiles to Profile HMMs

### Problem

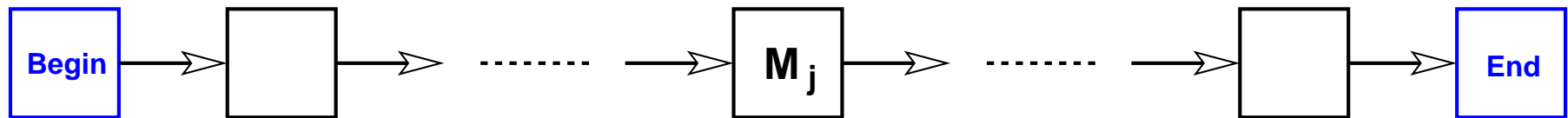
Given a multiple alignment (obtained either manually or using one of the methods presented in Ch. 6 and Ch. 7), and the profile associated to a set of marked (X = match) columns, design a HMM that would perform sequence alignments to that profile.

### Example

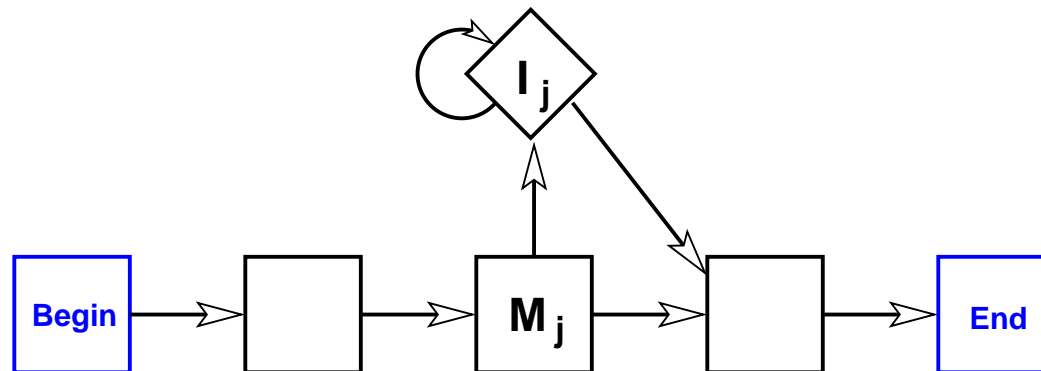
	X	X	.	.	.	X		1	2	.	.	.	3
bat	A	G	-	-	-	C	A	4/5	0				0
rat	A	-	A	G	-	C	C	0	0				4/5
cat	A	G	-	A	A	-	G	0	3/5				0
gnat	-	-	A	A	A	C	T	0	0				0
goat	A	G	-	-	-	C	-	1/5	2/5				1/5

## Building up a solution

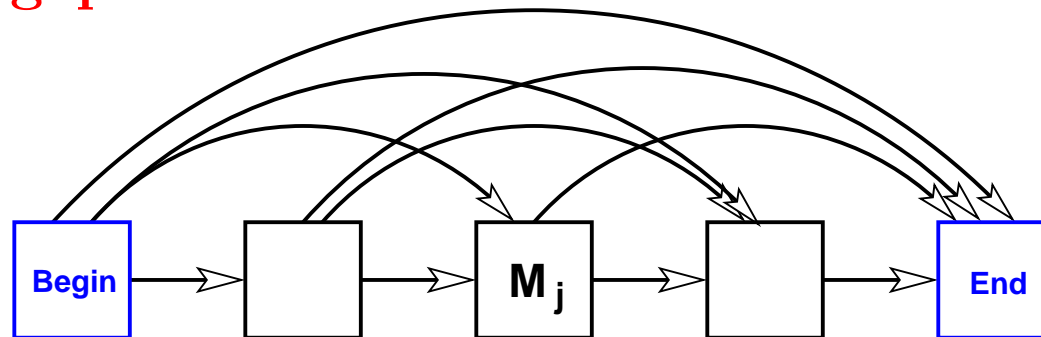
At first sight, not taking into account gaps:



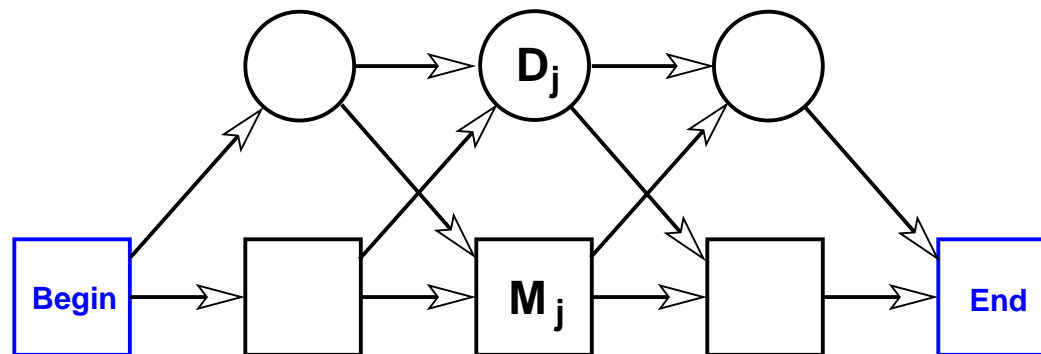
What about **insert residues**?



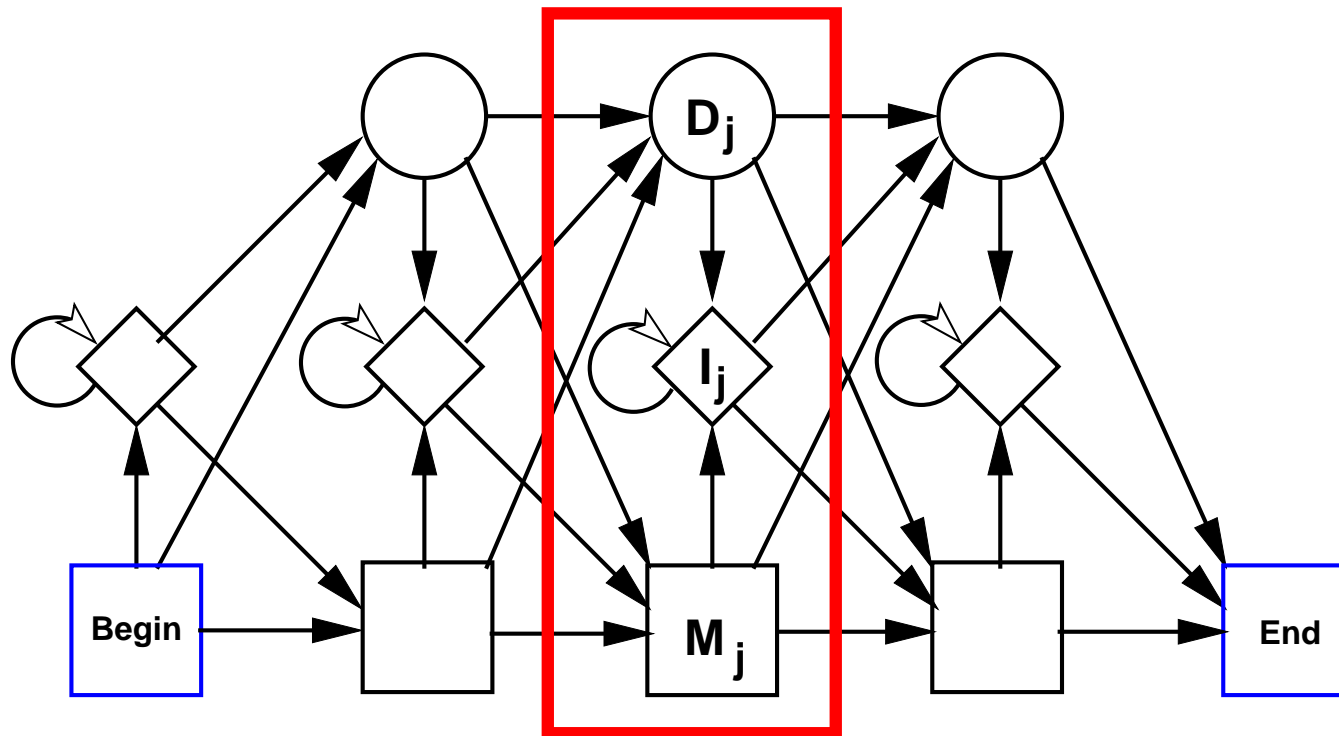
What about **gaps**?



A better treatment of gaps:



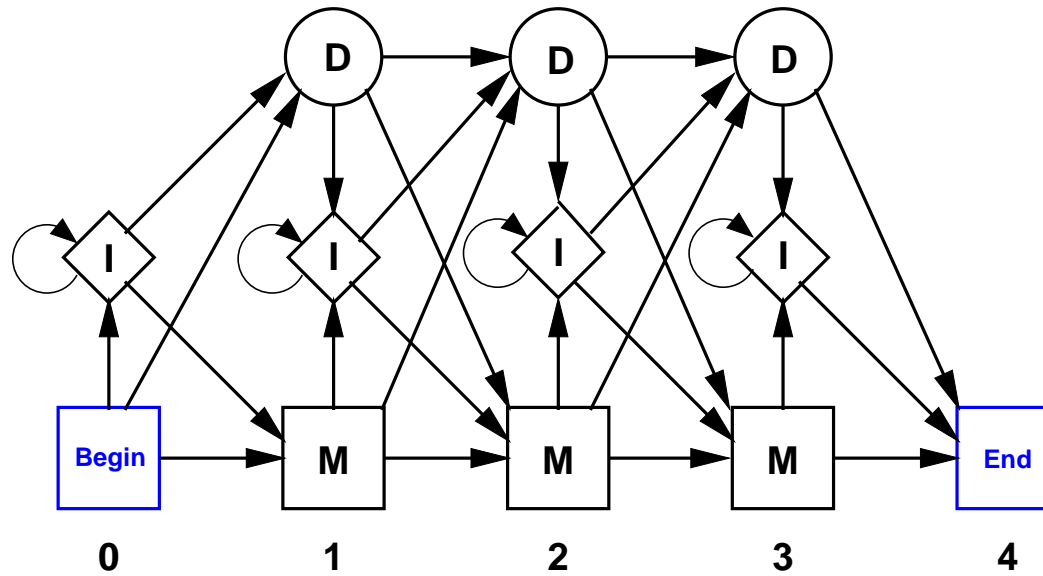
Could we put it all together?



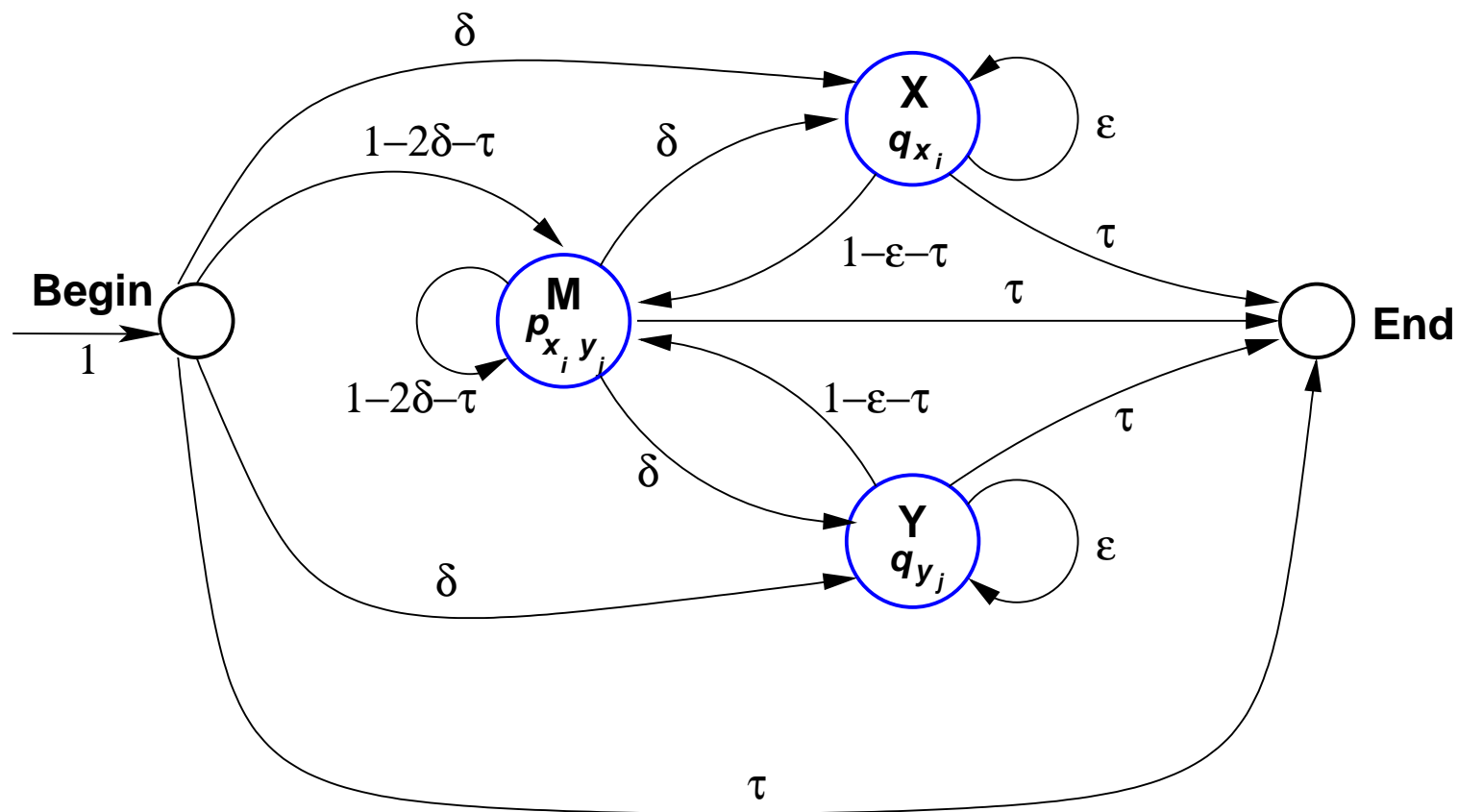
**Transition structure of a profile HMM**

Does it work?

	X	X	.	.	.	X
bat	A	G	-	-	-	C
rat	A	-	A	G	-	C
cat	A	G	-	A	A	-
gnat	-	-	A	A	A	C
goat	A	G	-	-	-	C
	1	2	.	.	.	3



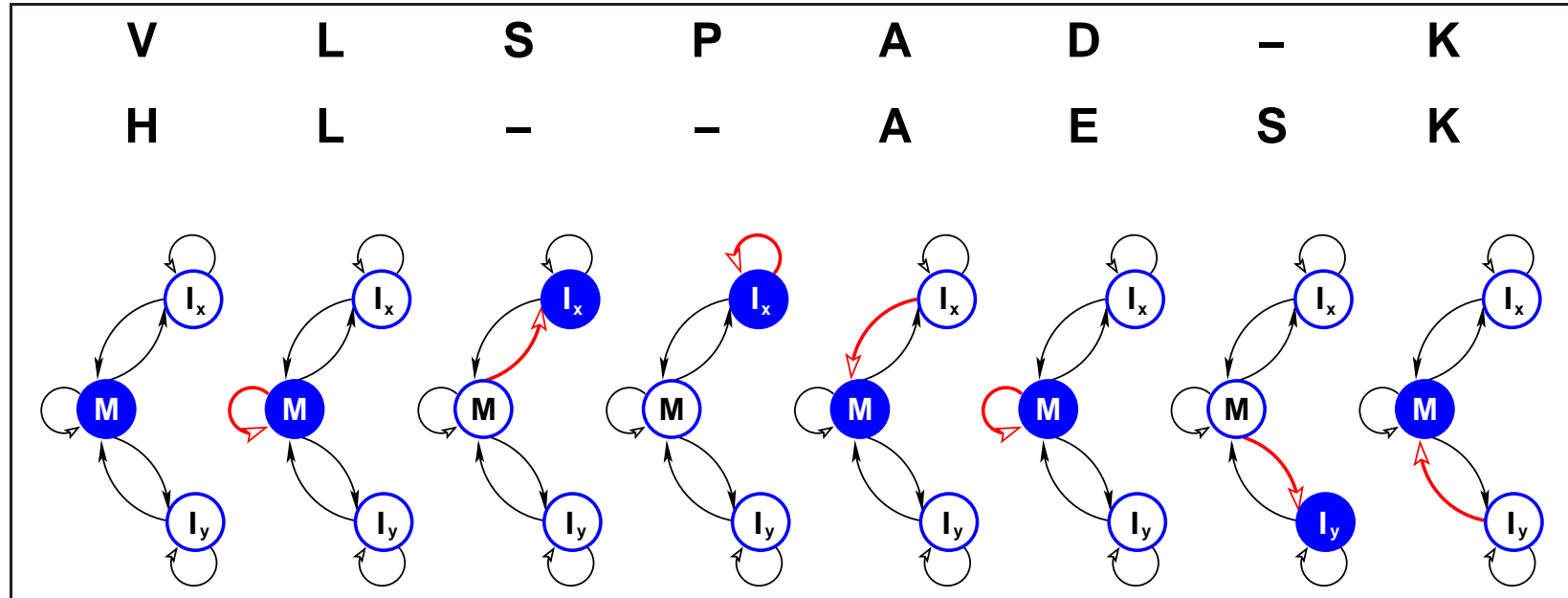
Any resemblance to pair HMMs?



Doesn't seem so...

## However, remember...

An **example** of the state assignments for global alignment using the affine gap model:



When making the extension to multiple sequence alignment,  
 think of generating only one string (instead of a pair of strings);  
 use  $I_x$  for inserting residues, and  $I_y$  to produce a gap;  
 use one triplet of states  $(M, I_x, I_y)$  for each column in the alignment;  
 finally define (appropriate) edges in the resulting FSA.

## Consequence

It shouldn't be difficult to re-write  
the basic HMM algorithms  
for profile HMMs!

one moment, though...

## 2 Setting the parameters of a Profile HMM

### 2.1 Using Maximum Likelihood Estimates (MLE) for transition and emission probabilities

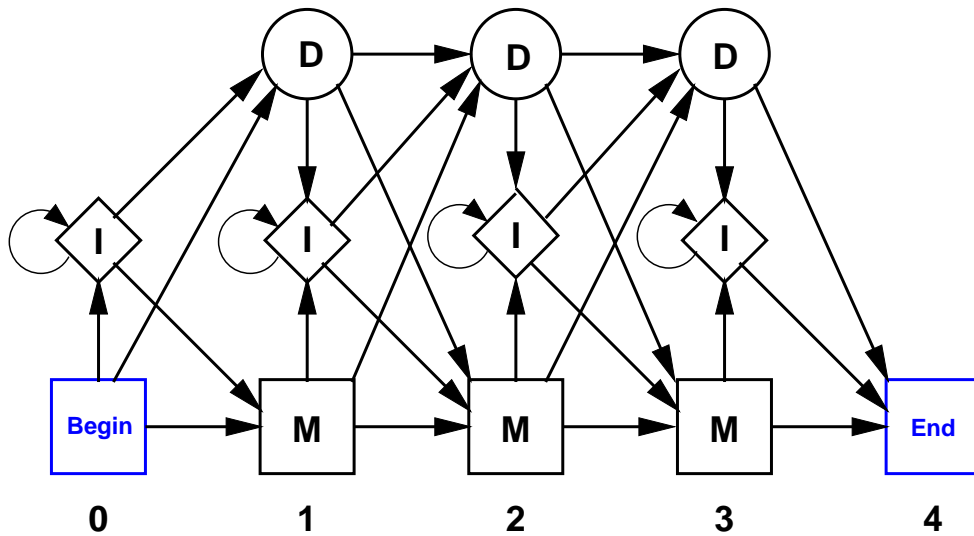
For instance — assuming a given multiple alignment with match states marked (X) —, the emission probabilities are computed as

$$e_{M_j}(a) = \frac{c_{ja}}{\sum_{a'} c_{ja'}}$$

where  $c_{ja}$  is the observed frequency of residue  $a$  in the column  $j$  of the multiple alignment.

# Counts for our example

	X	X	.	.	.	X
bat	A	G	-	-	-	C
rat	A	-	A	G	-	C
cat	A	G	-	A	A	-
gnat	-	-	A	A	A	C
goat	A	G	-	-	-	C
	1	2	.	.	.	3



	0	1	2	3	
match emissions	A	-	4	0	0
	C	-	0	0	4
	G	-	0	3	0
	T	-	0	0	0
insert emissions	A	0	0	6	0
	C	0	0	0	0
	G	0	0	1	0
	T	0	0	0	0
state transitions	M-M	4	3	2	4
	M-D	1	1	0	0
	M-I	0	0	1	0
state transitions	I-M	0	0	2	0
	I-D	0	0	1	0
	I-I	0	0	4	0
state transitions	D-M	-	0	0	1
	D-D	-	1	0	0
	D-I	-	0	2	0

## What about zero counts, i.e. unseen emissions/transitions?

One solution: **use pseudocounts** (generalising Laplace's law...):

$$e_{M_j}(a) = \frac{c_{ja} + Aq_a}{\sum_{a'} c_{ja'} + A}$$

where  $A$  is a weight put on pseudocounts as compared to real counts ( $c_{ja}$ ), and  $q_a$  is the frequency with which  $a$  appears in a random model.

$A = 20$  works well for protein alignments.

**Note:** At the intuitive level, using pseudocount makes a lot of sense:

- $e_{M_j}(a)$  is approximately equal to  $q_a$  if very little data is available, i.e. all real counts are very small compared to  $A$ ;
- when a large amount of data is available,  $e_{M_j}(a)$  is essentially equal to the maximum likelihood solution.

For **other solutions** (e.g. Dirichlet mixtures, substitution matrix mixtures, estimation based on an ancestor), you may see Durbin et al., 1998, Section 6.5.

## 2.2 Setting the $L$ parameter

- The process of **model construction** represents a way to decide which columns of the alignment should be assigned to match states, and which to insert states.
- There are  $2^L$  combinations of marking for alignment of  $L$  columns, and hence  $2^L$  different profile HMMs to choose from.

In a **marked column**, symbols are assigned to **match states** and gaps are assigned to delete states

In an **unmarked column**, symbols are assigned to **insert states** and gaps are ignored.

- There are at least **three ways** to determine the marking:
  - **manual** construction: the user marks alignment columns by hand;
  - **heuristic** construction: e.g. a column might be marked when the proportion of gap symbols in it is below a certain threshold;
  - **Maximum A Posteriori (MAP)** model construction: next slides.

## The MAP (maximum a posteriori) model construction

- **Objective:** we search for the model  $\mu$  that maximises the likelihood of the given data, namely:

$$\operatorname{argmax}_{\mu} P(\mathcal{C} \mid \mu)$$

where  $\mathcal{C}$  is a set of aligned sequences.

**Note:** The sequences in  $\mathcal{C}$  are assumed to be statistically independent.

- **Idea:** The MAP model construction algorithm recursively calculates  $S_j$ , the log probability of the optimal model for the alignment up to and including column  $j$ , assuming that the column  $j$  is marked.
- **More specifically:**  $S_j$  is calculated from smaller subalignments ending at a marked column  $i$ , by incrementing  $S_i$  with the summed log probability of the transitions and emissions for the columns between  $i$  and  $j$ .

## MAP model construction algorithm: Notations

- $c_{xy}$  — the observed state transition counts
- $a_{xy}$  — the transition probabilities, estimated from the  $c_{xy}$  in the usual fashion (MLE)

$$a_{xy} = \frac{c_{xy}}{\sum_y c_{xy}}$$

- $S_j$  — the log probability of the optimal model for the alignment up to and including column  $j$ , assuming that column  $j$  is marked
- $\mathcal{T}_{ij}$  — the summed log probability of all the state transitions between marked columns  $i$  and  $j$

$$\mathcal{T}_{ij} = \sum_{x,y \in M,D,I} c_{xy} \log a_{xy}$$

- $\mathcal{M}_j$  — the log probability contribution for match state symbol emissions in the column  $j$
- $\mathcal{L}_{i,j}$  — the log probability contribution for the insert state symbol emissions for the columns  $i + 1, \dots, j - 1$  (for  $j - i > 1$ ).

## The MAP model construction algorithm

**Initialization:**  $S_0 = 0, \mathcal{M}_{L+1} = 0$

**Recurrence:**

for  $j = 1, \dots, L + 1$

$$S_j = \max_{0 \leq i < j} S_i + \mathcal{T}_{ij} + \mathcal{M}_j + \mathcal{L}_{i+1, j-1} + \lambda$$

$$\sigma_j = \arg \max_{0 \leq i < j} S_i + \mathcal{T}_{ij} + \mathcal{M}_j + \mathcal{L}_{i+1, j-1} + \lambda$$

**Traceback:**

from  $j = \sigma_{L+1}$ , while  $j > 0$ :

mark column  $j$  as a match column;  $j = \sigma_j$

**Complexity:**

$\mathcal{O}(L)$  in memory and  $\mathcal{O}(L^2)$  in time for an alignment of  $L$  columns...  
with some care in implementation!

**Note:**  $\lambda$  is a penalty used to favour models with fewer match states. In Bayesian terms,  $\lambda$  is the log of the prior probability of marking each column. It implies a simple but adequate exponentially decreasing prior distribution over model lengths.

## 3 Basic algorithms for Profile HMMs

### Notations

- $v_{M_j}(i)$  — the probability of the best path matching the subsequence  $x_{1\dots i}$  to the (profile) submodel up to the column  $j$ , ending with  $x_i$  being emitted by the state  $M_j$ ;
- $v_{I_j}(i)$  — the probability of the best path ending in  $x_i$  being emitted by  $I_j$ ;
- $v_{D_j}(i)$  — the probability of the best path ending in  $D_j$  ( $x_i$  being the last character emitted before  $D_j$ ).
- $V_{M_j}(i), V_{I_j}(i), V_{D_j}(i)$  — the log-odds scores corresponding respectively to  $v_{M_j}(i), v_{I_j}(i), v_{D_j}(i)$ .
- $f_{M_j}(i)$  — the combined probability of all alignments up to  $x_i$  that end in state  $M_j$ , and similarly  $f_{I_j}(i), f_{D_j}(i)$ .
- $b_{M_j}(i), b_{I_j}(i), b_{D_j}(i)$  — the corresponding backward probabilities.

## The Viterbi algorithm for profile HMM

### Initialization:

rename the Begin state as  $M_0$ , and set  $v_{M_0}(0) = 1$ ;  
 rename the End state as  $M_{L+1}$

### Recursion:

$$v_{M_j}(i) = e_{M_j}(x_i) \max \begin{cases} v_{M_{j-1}}(i-1) a_{M_{j-1}M_j} \\ v_{I_{j-1}}(i-1) a_{I_{j-1}M_j} \\ v_{D_{j-1}}(i-1) a_{D_{j-1}M_j} \end{cases}$$

$$v_{I_j}(i) = e_{I_j}(x_i) \max \begin{cases} v_{M_j}(i-1) a_{M_jI_j}, \\ v_{I_j}(i-1) a_{I_jI_j} \\ v_{D_j}(i-1) a_{D_jI_j} \end{cases}$$

$$v_{D_j}(i) = \max \begin{cases} v_{M_{j-1}}(i) a_{M_{j-1}D_j} \\ v_{I_{j-1}}(i) a_{I_{j-1}D_j} \\ v_{D_{j-1}}(i) a_{D_{j-1}D_j} \end{cases}$$

### Termination:

the final score is  $v_{M_{L+1}}(n)$ , calculated using the top recursion relation.

## The Viterbi algorithm for profile HMMs: log-odds version

### Initialization:

$V_{M_0}(0) = 0$ ; (the Begin state is  $M_0$ , and the End state is  $M_{L+1}$ )

### Recursion:

$$V_{M_j}(i) = \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{M_{j-1}}(i-1) + \log a_{M_{j-1}M_j} \\ V_{I_{j-1}}(i-1) + \log a_{I_{j-1}M_j} \\ V_{D_{j-1}}(i-1) + \log a_{D_{j-1}M_j} \end{cases}$$

$$V_{I_j}(i) = \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{M_j}(i-1) + \log a_{M_jI_j}, \\ V_{I_j}(i-1) + \log a_{I_jI_j} \\ V_{D_j}(i-1) + \log a_{D_jI_j} \end{cases}$$

$$V_j^D(i) = \max \begin{cases} V_{M_{j-1}}(i) + \log a_{M_{j-1}D_j} \\ V_{I_{j-1}}(i) + \log a_{I_{j-1}D_j} \\ V_{D_{j-1}}(i) + \log a_{D_{j-1}D_j} \end{cases}$$

### Termination:

the final score is  $V_{M_{L+1}}(n)$ , calculated using the top recursion relation.

## The Forward algorithm for profile HMMs

**Initialization:**  $f_{M_0}(0) = 1$

**Recursion:**

$$f_{M_j}(i) = e_{M_j}(x_i)[f_{M_{j-1}}(i-1)a_{M_{j-1}M_j} + f_{I_{j-1}}(i-1)a_{I_{j-1}M_j} + f_{D_{j-1}}(i-1)a_{D_{j-1}M_j}]$$

$$f_{I_j}(i) = e_{I_j}(x_i)[f_{M_j}(i-1)a_{M_jI_j} + f_{I_j}(i-1)a_{I_jI_j} + f_{D_j}(i-1)a_{D_jI_j}]$$

$$f_{D_j}(i) = f_{M_{j-1}}(i)a_{M_{j-1}D_j} + f_{I_{j-1}}(i)a_{I_{j-1}D_j} + f_{D_{j-1}}(i)a_{D_{j-1}D_j}$$

**Termination:**

$$f_{M_{L+1}}(n+1) = f_{M_L}(n)a_{M_LM_{L+1}} + f_{I_L}(n)a_{I_LM_{L+1}} + f_{D_L}(n)a_{D_LM_{L+1}}$$

## The Backward algorithm for profile HMMs

### Initialization:

$$b_{M_{L+1}}(n+1) = 1;$$

$$b_{M_L}(n) = a_{M_L M_{L+1}}; \quad b_{I_L}(n) = a_{I_L M_{L+1}}; \quad b_{D_L}(n) = a_{D_L M_{L+1}}$$

### Recursion:

$$b_{M_j}(i) = b_{M_{j+1}}(i+1)a_{M_j M_{j+1}}e_{M_{j+1}}(x_{i+1}) + b_{I_j}(i+1)a_{M_j I_j}e_{I_j}(x_{i+1}) + b_{D_{j+1}}(i)a_{M_j D_{j+1}}$$

$$b_{I_j}(i) = b_{M_{j+1}}(i+1)a_{I_j M_{j+1}}e_{M_{j+1}}(x_{i+1}) + b_{I_j}(i+1)a_{I_j I_j}e_{I_j}(x_{i+1}) + b_{D_{j+1}}(i)a_{I_j D_{j+1}}$$

$$b_{D_j}(i) = b_{M_{j+1}}(i+1)a_{D_j M_{j+1}}e_{M_{j+1}}(x_{i+1}) + b_{I_j}(i+1)a_{D_j I_j}e_{I_j}(x_{i+1}) + b_{D_{j+1}}(i)a_{D_j D_{j+1}}$$

**The Baum-Welch (Expectation-Maximization) algorithm  
for Profile HMMs: re-estimation equations**

**Expected emission counts from sequence  $x$ :**

$$E_{M_j}(a) = \frac{1}{P(x)} \sum_{i|x_i=a} f_{M_j}(i) b_{M_j}(i)$$

$$E_{I_j}(a) = \frac{1}{P(x)} \sum_{i|x_i=a} f_{I_j}(i) b_{I_j}(i)$$

**Expected transition counts from sequence  $x$ :**

$$A_{X_j M_{j+1}} = \frac{1}{P(x)} \sum_i f_{X_j}(i) a_{X_j M_{j+1}} e_{M_{j+1}}(x_{i+1}) b_{M_{j+1}}(i+1)$$

$$A_{X_j I_j} = \frac{1}{P(x)} \sum_i f_{X_j}(i) a_{X_j I_j} e_{I_j}(x_{i+1}) b_{I_j}(i+1)$$

$$A_{X_j D_{j+1}} = \frac{1}{P(x)} \sum_i f_{X_j}(i) a_{X_j D_{j+1}} b_{D_{j+1}}(i)$$

where  $X_j$  is one of  $M_j$ ,  $I_j$ , and  $D_j$ .

## Avoiding local maxima

- The Baum-Welch algorithm is guaranteed to find a local maximum on the probability “surface” but there is no guarantee that this local optimum is anywhere near the global optimum.
- A more involved approach is to **use some form of stochastic search** algorithm that “bumps” Baum-Welch off from local maxima:
  - noise injection during Baum-Welch re-estimation,
  - simulated annealing Viterbi approximation of EM,
  - Gibbs sampling.

For **details**, see Durbin et al. Section 6.5, pages 154–158.

## 4 Getting simultaneously the model and the multiple alignment

### Profile HMM training from unaligned sequences

#### Initialization:

Choose the **length of the profile HMM** (i.e., the number of match states), and initialize the transition and emission parameters.

A **commonly used rule** is to set the profile HMM's length to be the average length of the training sequences.

#### Training:

Estimate the model using the **Baum-Welch algorithm** or its Viterbi alternative.

Start Baum-Welch from multiple different points to see if it all converges to approximately the same optimum.

If necessary, use a **heuristic method for avoiding local optima**. (See the previous slide.)

#### Multiple alignment:

Align all sequences to the final model using the Viterbi algorithm and build a multiple alignment.

## Further comments on profile HMM's initial parameters:

- One possibility:  
guess a multiple alignment of some or all given sequences.
- A further possibility:  
derive the model's initial parameters from the Dirichlet prior over parameters (see Ch. 11).
- Alternatively:  
use frequencies derived from the prior to initialise the model's parameters, then use this model to generate a small number of random sequences, and finally use the resulting counts as 'data' to estimate an initial model.
- Note:  
The model should be encouraged to use 'sensible' transitions; for instance transitions into match states should be large compared to other transition probabilities.

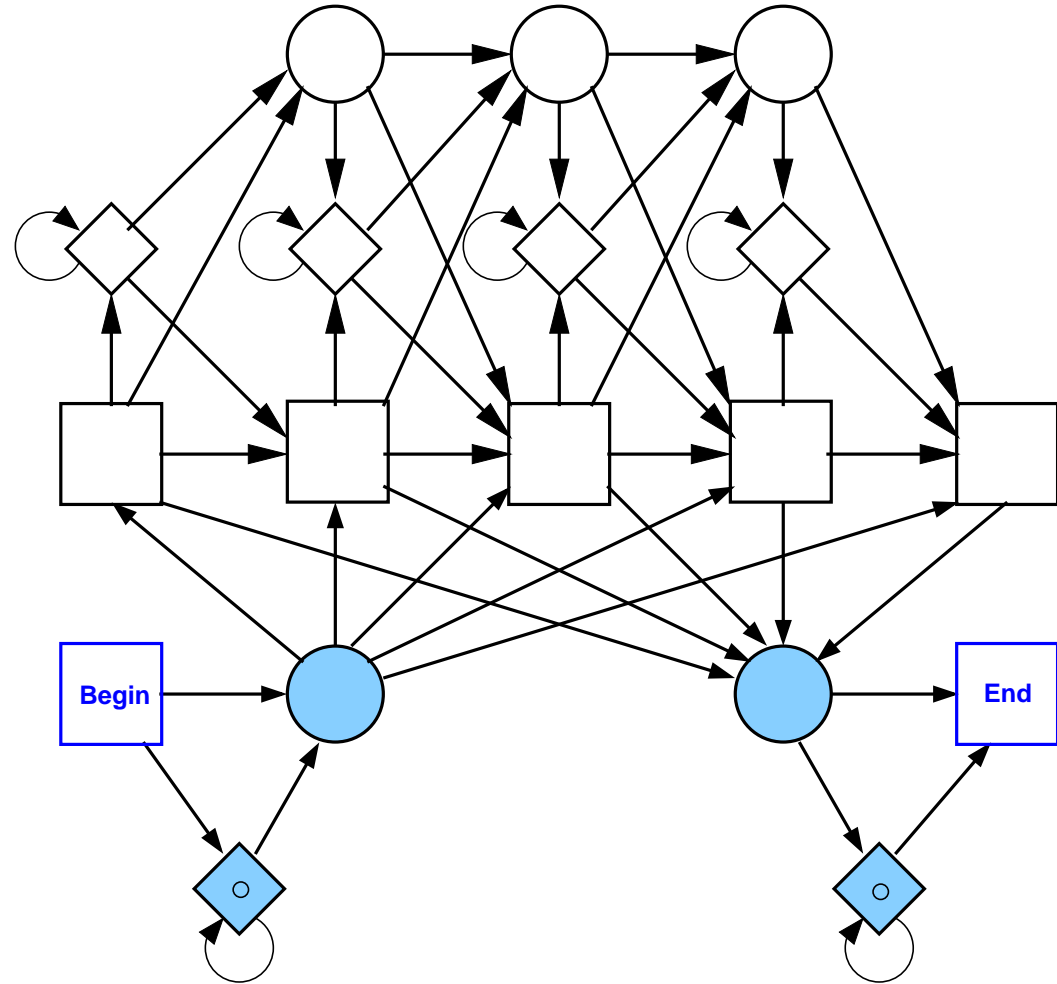
## Model surgery

After training a model we can **analyse the alignment** it produces:

- From **counts** estimated by the forward-backward procedure we can see how much a certain transition is used by the training sequences.
- The **usage of a match state** is the sum of counts for all letters emitted in the state.
- If a certain **match state** is used by less than half the number of given sequences, the corresponding module (triplet of match, insert, delete states) should be **deleted**.
- Similarly, if more than half (or some other predefined fraction) of the sequences use the transitions into a certain **insert state**, this should be **expanded** to some number of new modules (usually the average number of insertions).

## 5 Profile HMMs for non-global alignments

Local  
multiple alignment



The looping probabilities of the flanking (blue diamond) states should be close to 1; let's set them to  $1 - \eta$ .

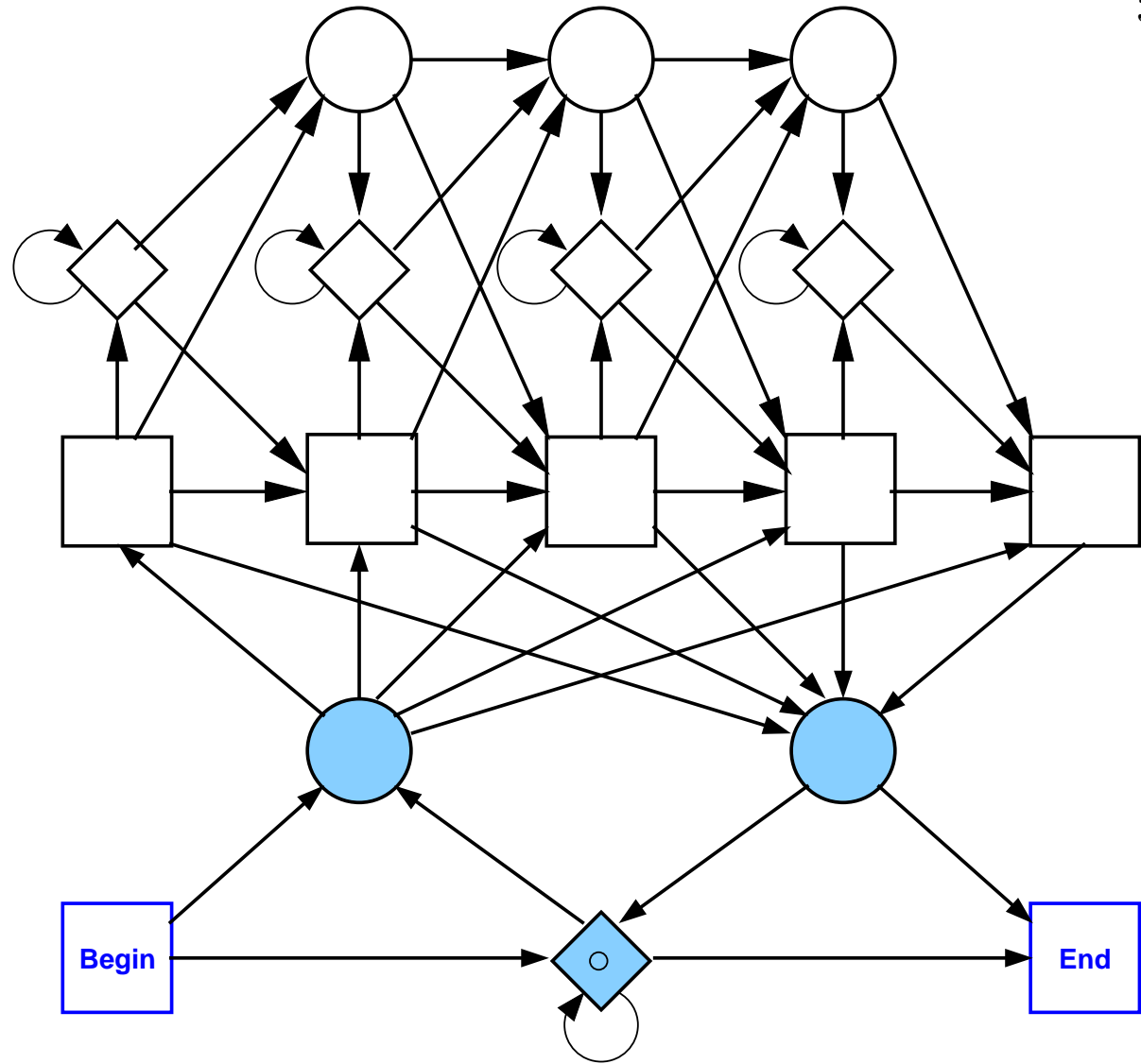
Transition probabilities from the left flanking state to the match states:

one option: all  $\eta/L$

another option:  $\eta/2$  for the transition into the first match state, and  $\eta/2(L - 1)$  for the other positions.



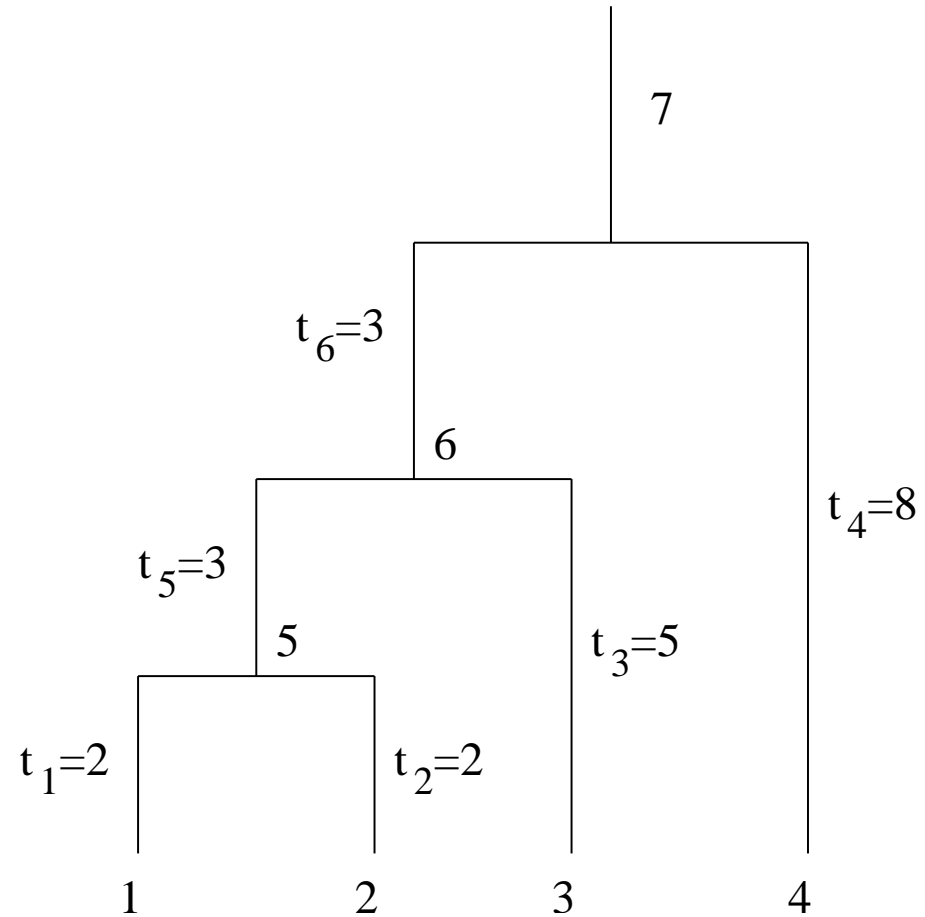
A profile HMM for repeat matches to subsections of the profile model



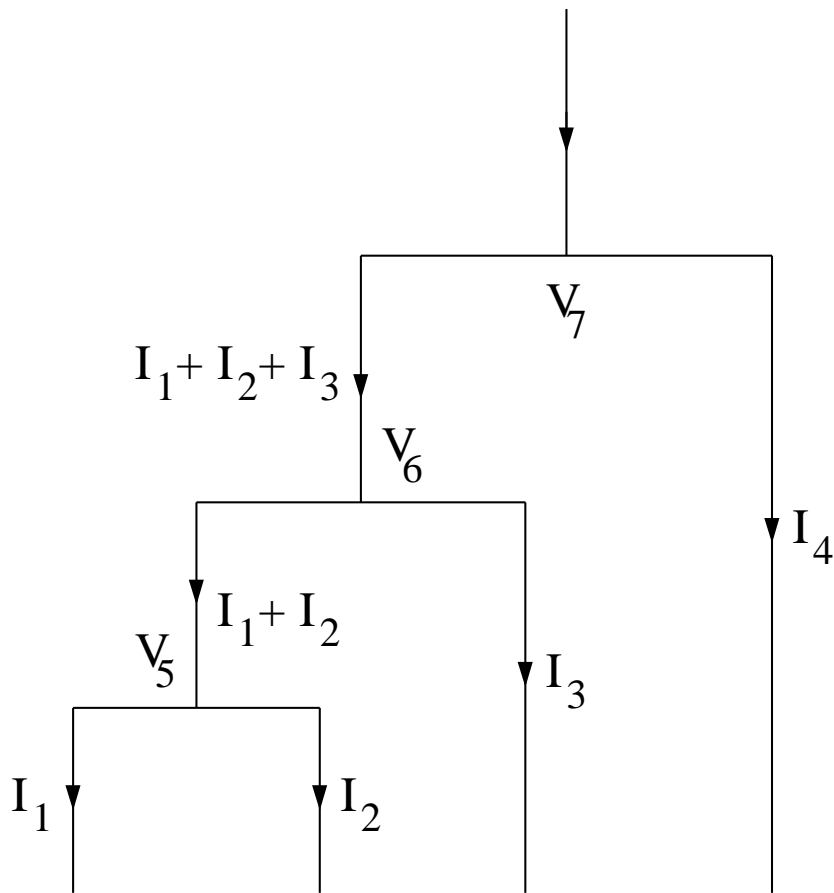
## 6 Weighting the training sequences

When the (training) sequences in the given multiple alignment are **not statistically independent**, one may use some simple **weighting schemes** derived from a **phylogenetic tree**.

**Example** of a phylogenetic tree:



## 6.1 A weighting scheme using Kirchhoff's laws [Thompson et al., 1994]



- Let  $I_n$  and  $V_n$  be the current and respectively the voltage at node  $n$ . We can set the resistance equal to the edge length/time.
- Equations:  

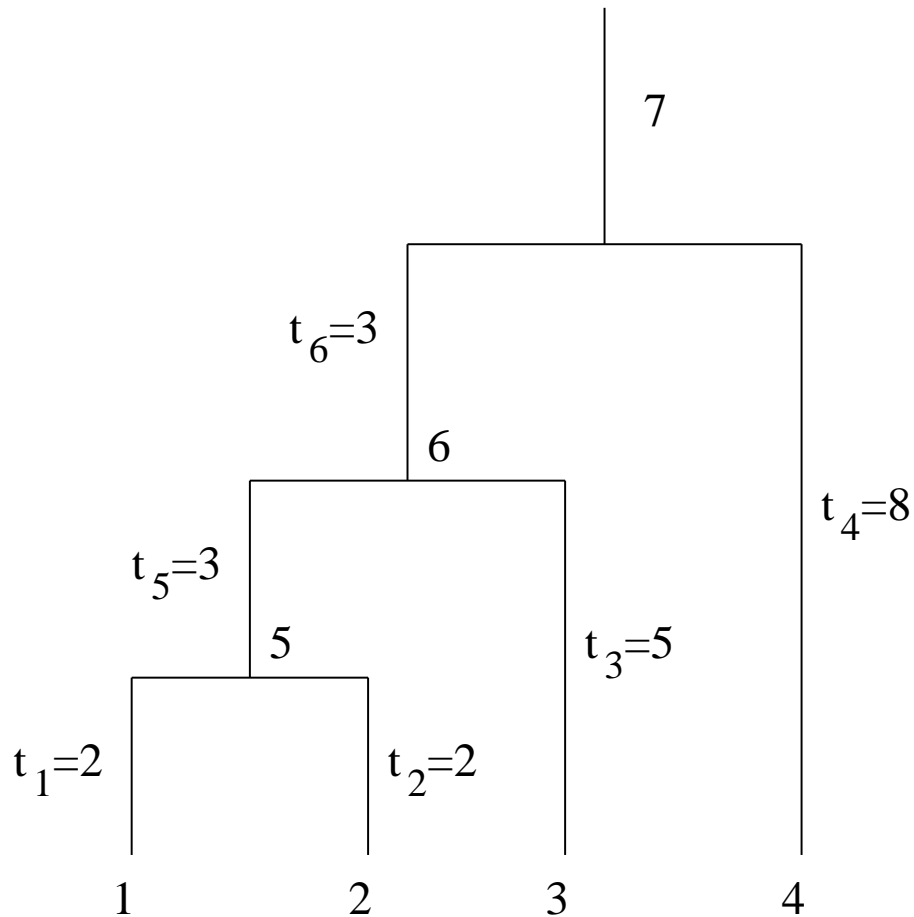
$$V_5 = 2I_1 = 2I_2$$

$$V_6 = 2I_1 + 3(I_1 + I_2) = 5I_3$$

$$V_7 = 8I_4 = 5I_3 + 3(I_1 + I_2 + I_3)$$
- Result:  

$$I_1 : I_2 : I_3 : I_4 = 20 : 20 : 32 : 47$$

## 6.2 Another simple algorithm [Gerstein et al., 1994]



- Initially the weights are set to the edge lengths of the leafs:  
 $w_1 = w_2 = 2, w_3 = 5, w_4 = 8.$

- Then

$$\Delta\omega_i = t_n \frac{\omega_i}{\sum_{\text{leaves } k \text{ below } n} \omega_k}$$

So, at node 5:

$$w_1 = w_2 = 2 + 3/2 = 3.5$$

At node 6:

$$w_1 = w_2 = 3.5 + 3 \times 3.5/12,$$

$$w_3 = 5 + 3 \times 5/12$$

- Result:

$$w_1 : w_2 : w_3 : w_4 = 35 : 35 : 50 : 64$$

For details on other, more involved weighting schemes,

- Root weights from Gaussian parameters
- Voronoi weights
- Maximum discrimination weights
- Maximum entropy weights

you may see Durbin et al., Section 5.8, pages 126–132

## Examples

(including the use of weights in the computation of parameters for the HMM profile):

pr. 5.6, 5.10 in [Borodovsky, Ekisheva, 2006]