BioInformática

Vítor Santos Costa

DCC FCUP

16 de setembro de 2022

Estruturas de Dados:

Processamento:

- números
- strings
- listas
- dicionários

Estruturas do Programa

- Variáveis
- Frases
- Funçõess
- Classes
- Módulos

Python: funções

```
g = 7

def sum(x,y,z):
    print(g)
    g = max(x,y,z)
    return x+y+z+g
```

Contexto: variavel criada por atribuiçõa.

Python: Parâmetros

```
g = 7

def op(x,y=0,z=0):
    print(g)
    g = max(x,y,z)
```

Parâmetros

- Posicionais
- Opcionais

Python: classes

```
class Point:
    """ Point class represents and manipulates x,y coo

    def __init__(self, x=0, y=0):
        """ Create a new point at x, y """
        self.x = x
        self.y = y

    def n1(self):
        return math.abs(x)+math.abs(y)
```

- Contexto: funções e variáveis.
- `self` refere-se ao objeto
- `__init__` inicializo

Python: Herança

```
    """ Point class represents and manipulates x,y coo

    def __init__(self, x=0, y=0):
        """ Create a new point at x, y """
        self.x = x
        self.y = y

    def n1(self):
        return math.abs(x)+math.abs(y)
```

- Herda tudo de Point, incluindo `__init__`

Dados em Python: Dicionários

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'jack': 4098, 'sape': 4139, 'guido': 4127}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'jack': 4098, 'guido': 4127, 'irv': 4127}
>>> list(tel)
['jack', 'guido', 'irv']
```

Python: Pacotes de BioInfo

Listas e Dicionários não permitem

-
- biotite-python.org
- htseq.readthedocs.io
- https://github.com/gokceneraslan/awesome-deepbio

Alinhamento de Pares de Sequências

Dada:

- Um par de sequências (DNA ou proteína)
- um método para calcular a similaridade de um par de caracteres na sequência

Faça

- Encontre as correspondências entre subsequências nas sequências que maximizam uma *função de semelhança*.

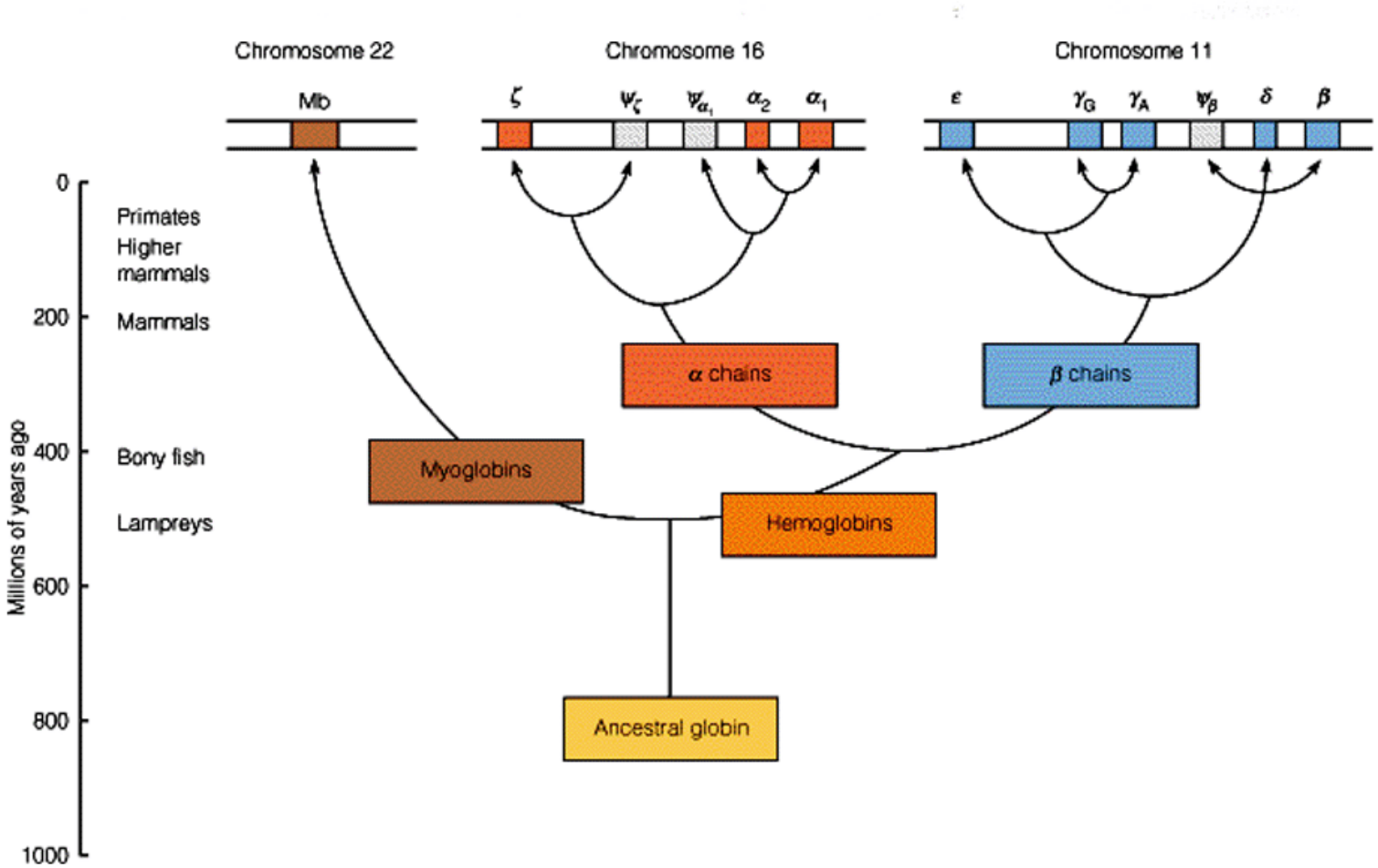
Motivação

- Comparar sequências para obter informação sobre a estrutura e função de uma sequência.
- Juntar um conjunto de fragmentos de sequência
- Comparar um segmento sequenciado por diferentes laboratórios

A Importância de Homologia

- *Homologia*: semelhança causada por descendência do mesmo antepassado
- Muitas vezes podemos inferir homologia de similaridade
- Utilidade: inferir estrutura/função a partir de similaridade.

Exemplo: Globinas



Homologia

- Sequências homologas podem ser divididas em dois grupos:
  - Ortólogas: divergiram para espécies diferentes (eg, α-globina humana e do rato)
  - Parálogas: divergiram devido a duplicação de genes na mesma espécie (eg, as várias versões da α-globina humana e da β-globina humana).

Problemas no Alinhamento de Sequências

- As sequências que vamos comparar provavelmente diferem em tamanho
- Pode haver apenas uma pequena região nas sequências que alinha
- queremos permitir alinhamentos parciais: por ex, alguns pares de amino-ácidos são mais substituíveis do que outros
- regiões de tamanho variável podem ter sido inseridas ou removidas do antepassado comum.

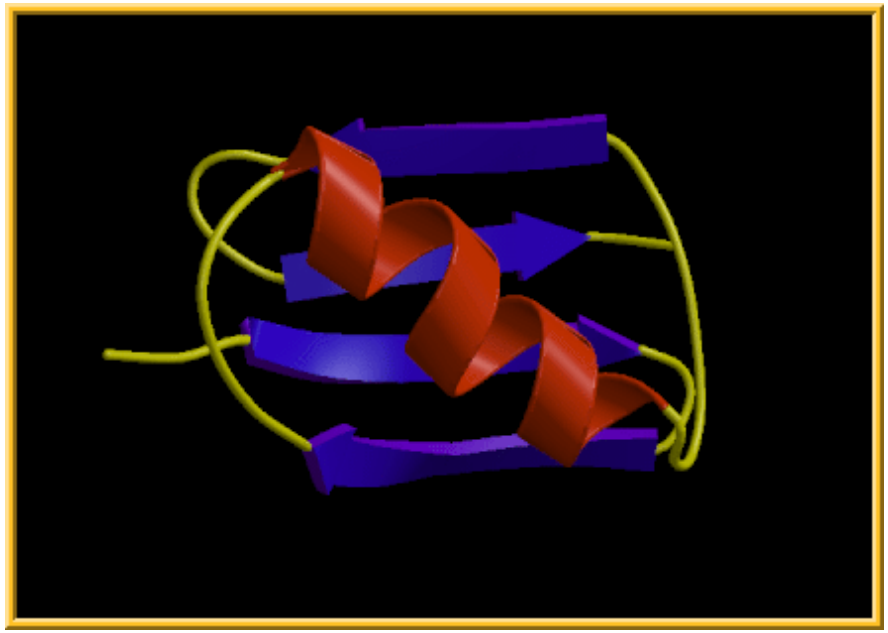
Buracos

- Sequências podem ter divergido do antepassado comum através de vários tipos de mutações:
  - Substituições: $(ACGA \longrightarrow AGGA)$
  - Inserções: $(ACGA \longrightarrow ACCGA)$
  - Remoções: $(ACGA \longrightarrow AGA)$
- apesar das matrizes de substituição serem diferentes
- encontra alinhamento óptimo
- o algoritmo exacto (e complexidade) depende da função de penalização de buracos

Inserções e Remoções vs Estrutura da Proteína

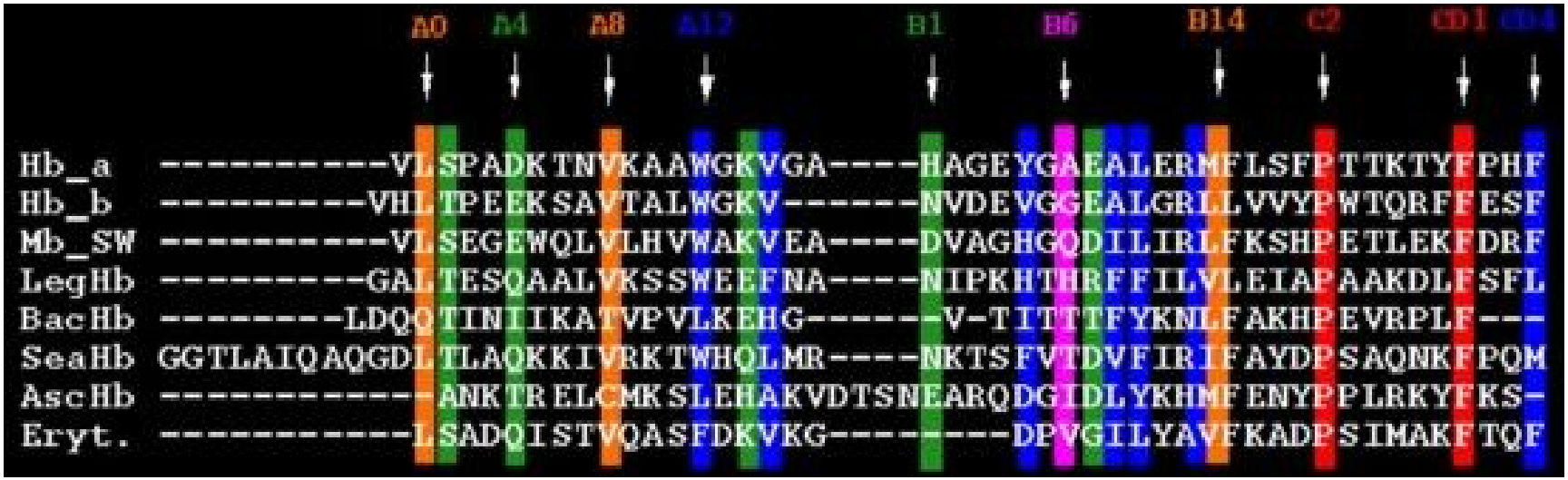
Porque é que duas sequências semelhantes podem ter muitas inserções ou remoções

- Inserções e remoções podem não afectar significativamente a estrutura da proteína.



Exemplo de Alinhamento: Globinas

- À direita, estrutura tipo de Globinas
- Em baixo, parte do Alinhamento para oito globinas



Tipos de Alinhamento

- *Global*: encontrar o melhor alinhamento entre sequências completas
- *Local*: encontrar o melhor alinhamento entre subsequências completas
- *Semi-Global*: encontrar o melhor alinhamento sem penalizar espaços brancos nas bordas do alinhamento

Como Avaliar um Alinhamento

- Matriz de substituição:
  - $s(a, b)$ indica o preço de alinhar o caracter a com o caracter b.
- Função de penalização de intervalos:
  - $w(k)$ indica o custo de um intervalo de tamanho k.

Função de Penalização Linear

- Diferentes funções de penalização podem requerer algoritmos de programação dinâmica diferente
  - O caso mais simples é quando usamos uma função linear:

$$w(k) = gk$$

onde g é uma constante

- Vamos começar por aqui.

Pontuação de Alinhamentos

- A pontuação de um alinhamento é:
  1. somatório dos pares de caracteres alinhados,
  2. mais pontuação para buracos
- Exemplo: dado o alinhamento

$$\begin{array}{l} \texttt{VAHV}---\texttt{D}-\texttt{-DMPNALSALSDLHAHKL} \\ \texttt{AIQLQVTGVVVTDATLKNLGSVHVSKG} \end{array}$$

- a pontuação será:

$$s(\texttt{V},\texttt{A}) + s(\texttt{A},\texttt{I}) + s(\texttt{H},\texttt{Q}) + 3g + s(\texttt{D},\texttt{G}) + \ldots$$

Alinhamento de Pares por Programação Dinâmica

- Needleman & Wunsch, Journal of Molecular Biology, 1970
- *Programação Dinâmica*: resolver uma instância de um problema usando soluções computadas para pequenas partes do problema.
- Ideia: determinar alinhamento óptimo de duas sequências determinando o melhor alinhamento para todos os prefixos.

Alinhamento de Pares por Programação Dinâmica

- Considere o último passo na computação do alinhamento de AAAC com AGC
- Três opções possíveis:

A	A	A	C
A	G		C

A	A	A	C	–
	A	G		C

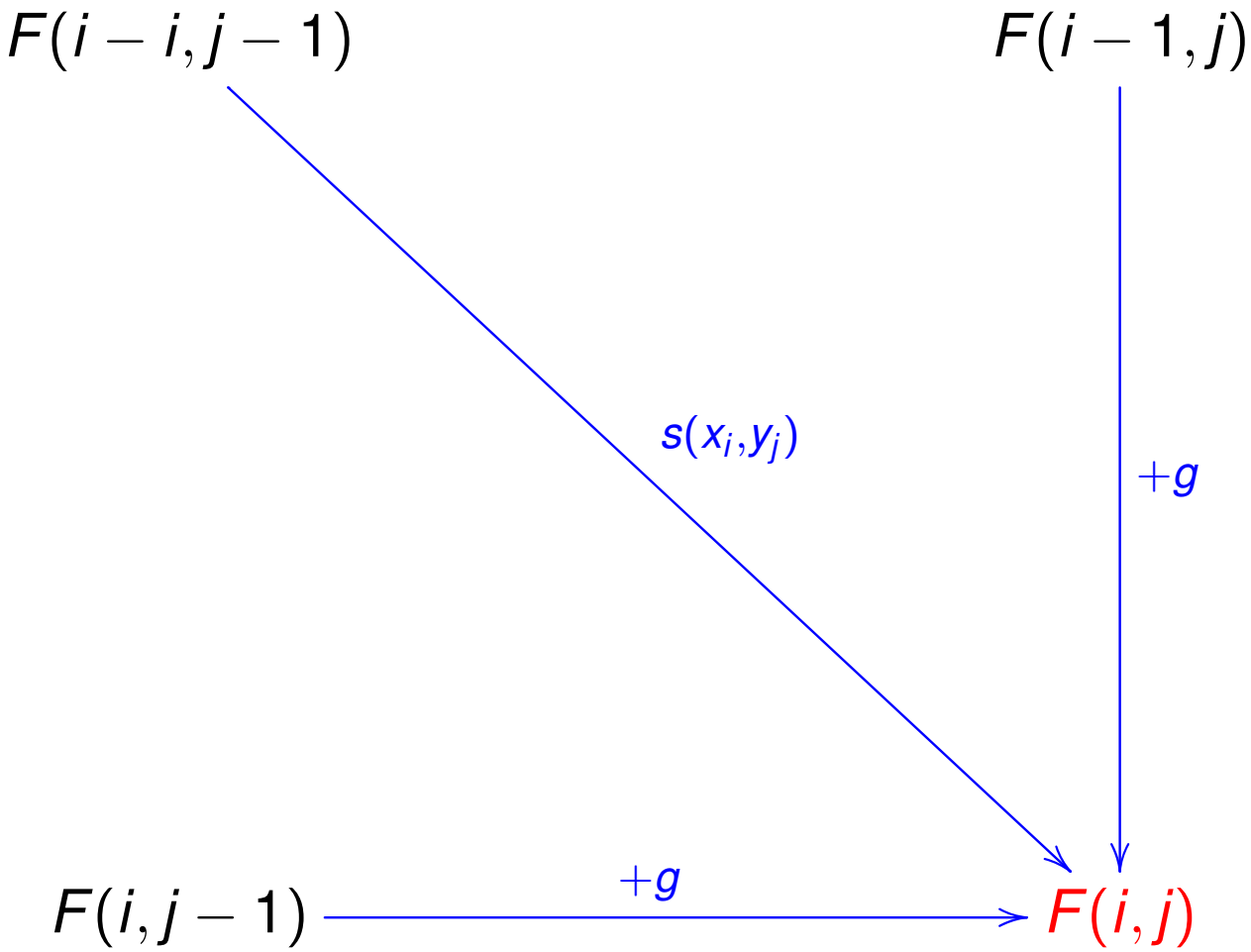
A	A	A	C
A	G	C	–

- Considere:
  1. Melhor Alinhamento dos Prefixos +
  2. Resultado do Alinhamento do par

Programação Dinâmica

1. Dada uma sequência de n caracteres x e uma sequência de m caracteres y,
2. Construa uma matriz F de dimensão $(n+1) \times (m+1)$
3. $F(i,j) =$ resultado do melhor alinhamento de $x[1 \ldots i]$ com $y[1 \ldots j]$.

Programação Dinâmica: Ideia Básica

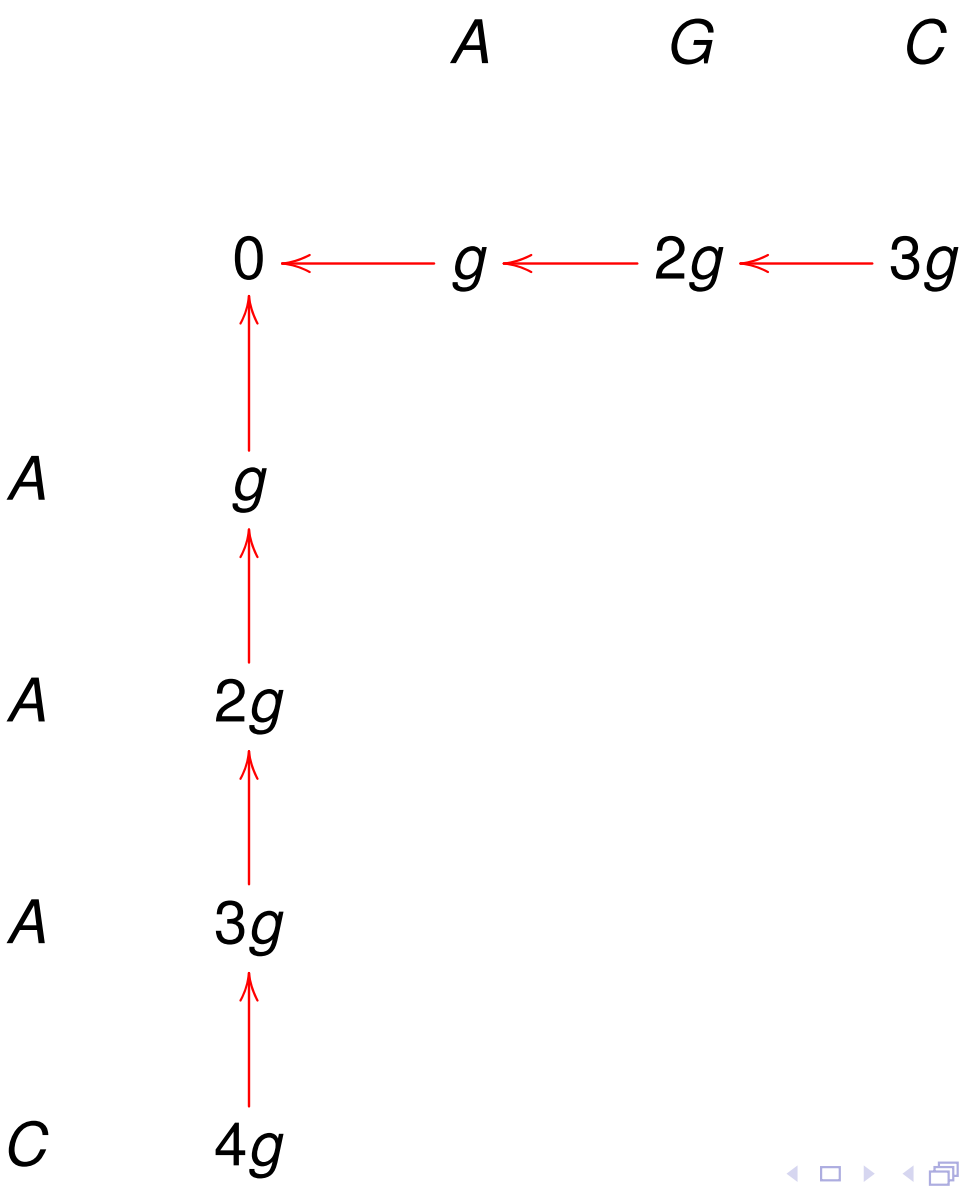


Algoritmo para Alinhamento Global com Penalização Linear de Buracos

Uma maneira é especificar a DP através da sua relação de recorrência:

$$F(i,j) = \textbf{max} \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

Inicialização da Matriz



Esquema do Algoritmo

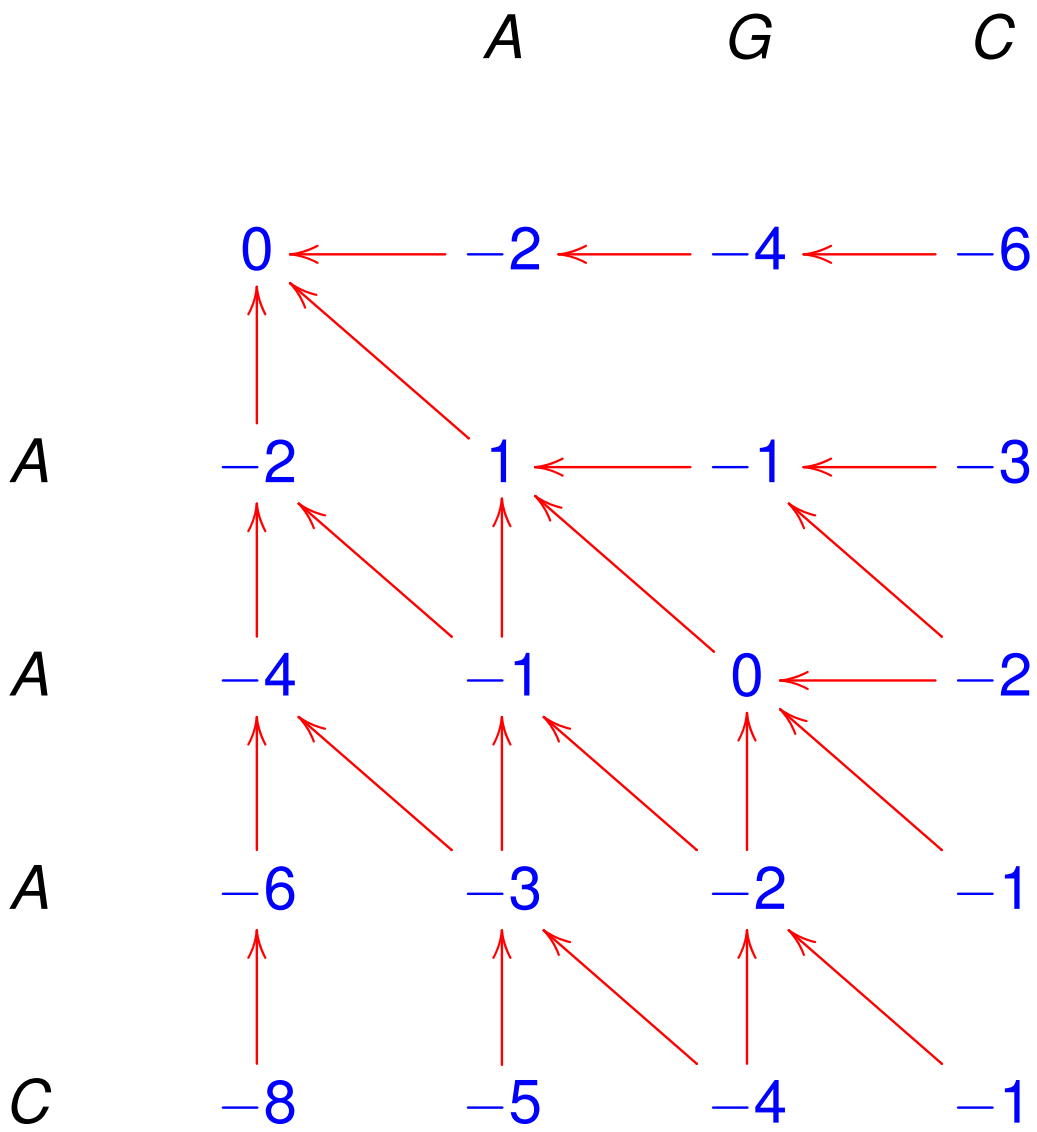
- inicializar primeira linha e coluna da matriz
- preencher o resto da matriz de cima para baixo, e esquerda para a direita
- para cada $F(i, j)$, guarde ponteiro para célula que deu o melhor resultado
- $F(m, n)$ tem a pontuação de alinhamento óptima: siga os ponteiros desde $F(m, n)$ até $F(0, 0)$ para recuperar o alinhamento.

Exemplo do Esquema do Algoritmo

Imagine que escolhíamos o seguinte esquema de pontuação:

- acerto: $+1$
- erro: -1
- g(penalidade para alinhar com um buraco) $= -2$

Exemplo do Esquema do Algoritmo

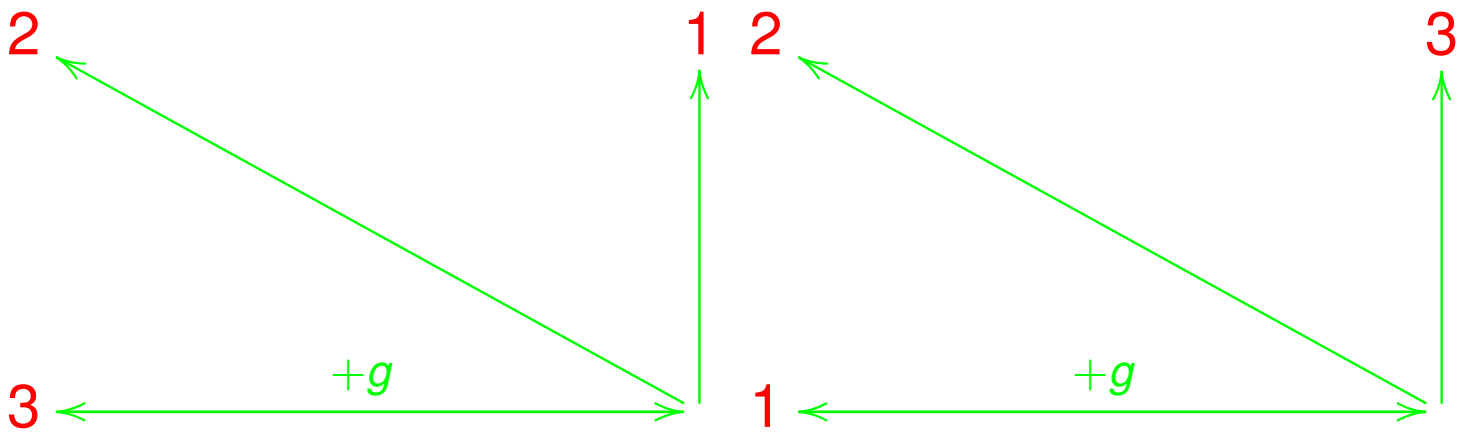


Comentários

- Funciona tanto para DNA como para sequências de amino-ácidos, apesar das matrizes de substituição serem diferentes
- encontra alinhamento óptimo
- o algoritmo exacto (e complexidade) depende da função de penalização de buracos

Alinhamentos Igualmente Ótimos

- muitos alinhamentos óptimos podem existir para um par dado de sequências
- podemos usar escolha de preferências sobre caminhos quando voltamos para trás:



- O *caminho alto* e o *caminho baixo* mostram os dois alinhamentos óptimos mais diferentes.

Análise do Algoritmo de Programação Dinâmica

- Existem

$$(2n) = \frac{(2n)!}{(n!)^2} \approx \frac{2^{2n}}{\sqrt{\pi n}}$$

  alinhamentos possíveis para 2 sequências de tamanho n
- ie, duas sequências de tamanho 1000 têm $\approx 10^{600}$ alinhamentos possíveis
- mas o algoritmo DP encontra o alinhamento óptimo eficientemente.

Complexidade Computacional

- inicialização: $O(m), O(n)$
- preenchendo o resto da matriz: $O(mn)$
- voltar para trás: $O(m+n)$
- se as duas sequências tiverem o mesmo tamanho, a complexidade computacional é:

$$O(n^2)$$

Alinhamento Local

- Até agora discutimos *alinhamento globais*, onde estamos procurando o melhor emparelhamento de duas sequências desde um fim ao outro
- mais frequentemente, queremos um *alinhamento local*, o melhor alinhamento entre subsequências de x e y

Motivação

- útil para comparar sequências de proteínas que partilham um *motivo* (padrão conservado) ou *domínio* (unidade independente enrolada) mas que diferem no resto
- útil para comparar sequências de DNA que partilham um *motivo* (padrão conservado) mas que diferem no resto
- útil para comparar sequências de proteínas contra *sequências de DNA do genoma* (longos grupos de sequências não caracterizadas)
- mais preciso para comparar sequências que divergiram muito

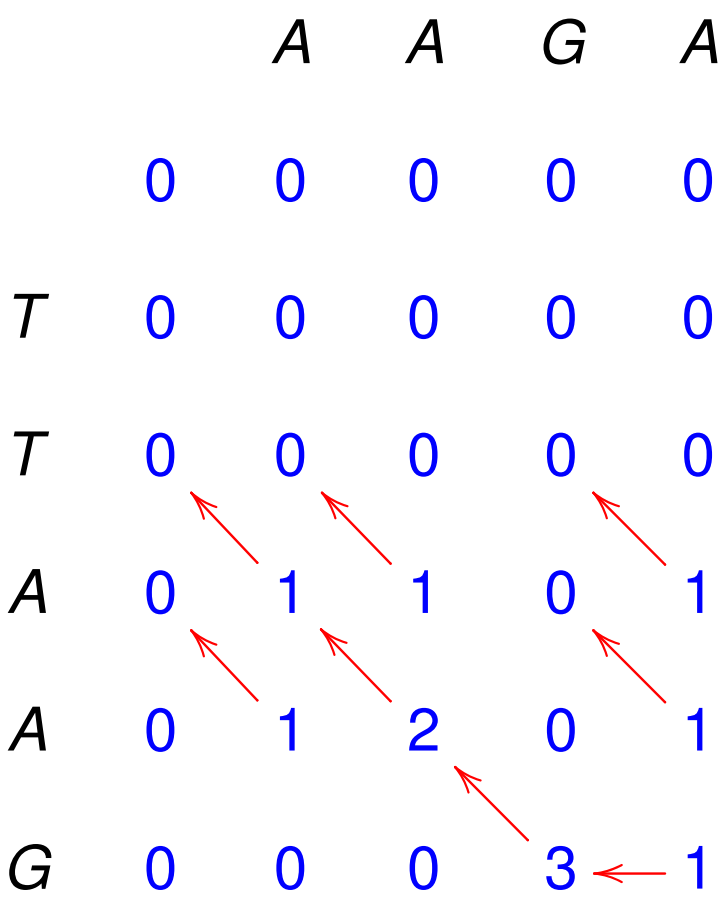
Algoritmo de Alinhamento Local por DP

- formulação original: Smith & Waterman, *Journal of Molecular Biology*, 1981
- Interpretação das matrizes é um pouco diferente:
  - $F(i, j) =$ pontuação do melhor alinhamento de um sufixo de $x[1 \ldots i]$ e um sufixo de $y[1 \ldots j]$

Algoritmo de Alinhamento Local por DP

- Inicialização: primeira linha e coluna inicializada com 0s
- Retorno:
  - encontrar valor máximo de $F(i, j)$; pode ser em qualquer posição da matriz
  - parar quando encontrar uma célula com o valor 0.

Exemplo de Alinhamento Local



x: A A G

y: A A G

Funções de Penalização de buracos

- linear
  $w(k) = gk$
- afim

$$w(k) = \begin{cases} h + gk, & k \geq 1 \\ 0, & k = 0 \end{cases}$$

- Côncava:
  $w(k + m + l) - w(k + m) \leq w(k + m) - w(k)$
  - Ex: $w(k) = h + g \times log(k)$

Programação Dinâmica para o caso afim

- Para conseguir em tempo $O(n^2)$ precisamos de 3 matrizes em vez de 1:
  - $M(i,j)$ melhor valor se $x[i]$ estiver alinhado com $y[j]$
  - $I_x(i,j)$ melhor valor se $x[i]$ estiver alinhado com um buraco
  - $I_y(i,j)$ melhor valor se $y[i]$ estiver alinhado com um buraco

DP para o caso afim, global

- $M(i,j) = max \begin{cases} M(i-1,j-1) + s(x_i, y_j) \\ I_x(i-1,j-1) + s(x_i, y_j) \\ I_y(i-1,j-1) + s(x_i, y_j) \end{cases}$
- $I_x(i,j) = max \begin{cases} M(i-1,j) + h + g \\ I_x(i-1,j) + g \end{cases}$
- $I_y(i,j) = max \begin{cases} M(i,j-1) + h + g \\ I_y(i,j-1) + g \end{cases}$
- Assumimos que é sempre melhor um match do que 2 buracos

DP para o caso afim global

- Inicialização
  - $M(0,0) = 0$
  - $I_x(i,0) = h + g \times i$
  - $I_y(0,j) = h + g \times j$
  - outras células no topo e coluna da esquerda $= -\infty$
- Voltar para trás:
  - começar no maior de $M(m,n), I_x(m,n), I_y(m,n)$
  - parar num de $M(0,0), I_x(0,0), I_y(0,0)$

DP para o caso afim local

- $$M(i,j) = max \begin{cases} M(i-1,j-1) + s(x_i, y_j) \\ I_x(i-1,j-1) + s(x_i, y_j) \\ I_y(i-1,j-1) + s(x_i, y_j) \\ 0 \end{cases}$$
- $$I_x(i,j) = max \begin{cases} M(i-1,j) + h + g \\ I_x(i-1,j) + g \end{cases}$$
- $$I_y(i,j) = max \begin{cases} M(i,j-1) + h + g \\ I_y(i,j-1) + g \end{cases}$$

DP para o caso afim local

- Inicialização
  - $M(0,0) = 0$
  - $I_x(i,0) = 0$
  - $I_y(0,j) = 0$
  - outras células no topo e coluna da esquerda $= -\infty$
- Voltar para trás:
  - começar no maior de $M(i,j)$
  - parar num $M(i,j) = 0$

DP Para o Caso Geral

Alinhamento Global:

- $F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(k,j) + \gamma(i-k) \\ F(i,k) + \gamma(j-k) \end{cases}$
- Considerar todos os elementos anteriores na linha!
- Considerar todos os elementos anteriores na coluna!

Complexidade Computacional

Dependendo da penalização de buracos:

- linear:

$$O(n^2)$$

- afim:

$$O(n^2)$$

- geral:

$$O(n^3)$$

Motivação para Uso de Heurísticas

- $O(mn)$ demasiado lento para grandes bancos de dados com muitas interrogações
- métodos heurísticos permitem aproximação rápida à programação dinâmica:
  - FASTA, de Pearson & Lipman, 1988
  - BLAST, de Altschul et al., 1990

Motivação para Alinhamento Por Heurísticas

- Imaginem procurar SWISS-PROT contra uma sequência de interrogação:
  - imaginem que a nossa pergunta tem 362 amino-ácidos
  - SWISS-PROT versão 38 contém 29.085.265 amino-ácidos
  - procurar alinhamentos locais através da programação dinâmica obrigaria a $O(10^{10})$ operações em matrizes
- muitos servidores têm que resolver milhares de tais perguntas por dia
  - NCBI $>$ 100.000

BLAST

- **B**asic **L**ocal **A**lignment **S**earch **T**ool
- BLAST usa heurísticas para encontrar *pares com pontuação alta* (HSPs):
  - Segmentos do mesmo tamanho de 2 sequências com pontuação de alinhamento estatisticamente significantes
  - ie, alinhamentos locais sem buracos
- Escolha entre precisão e velocidade

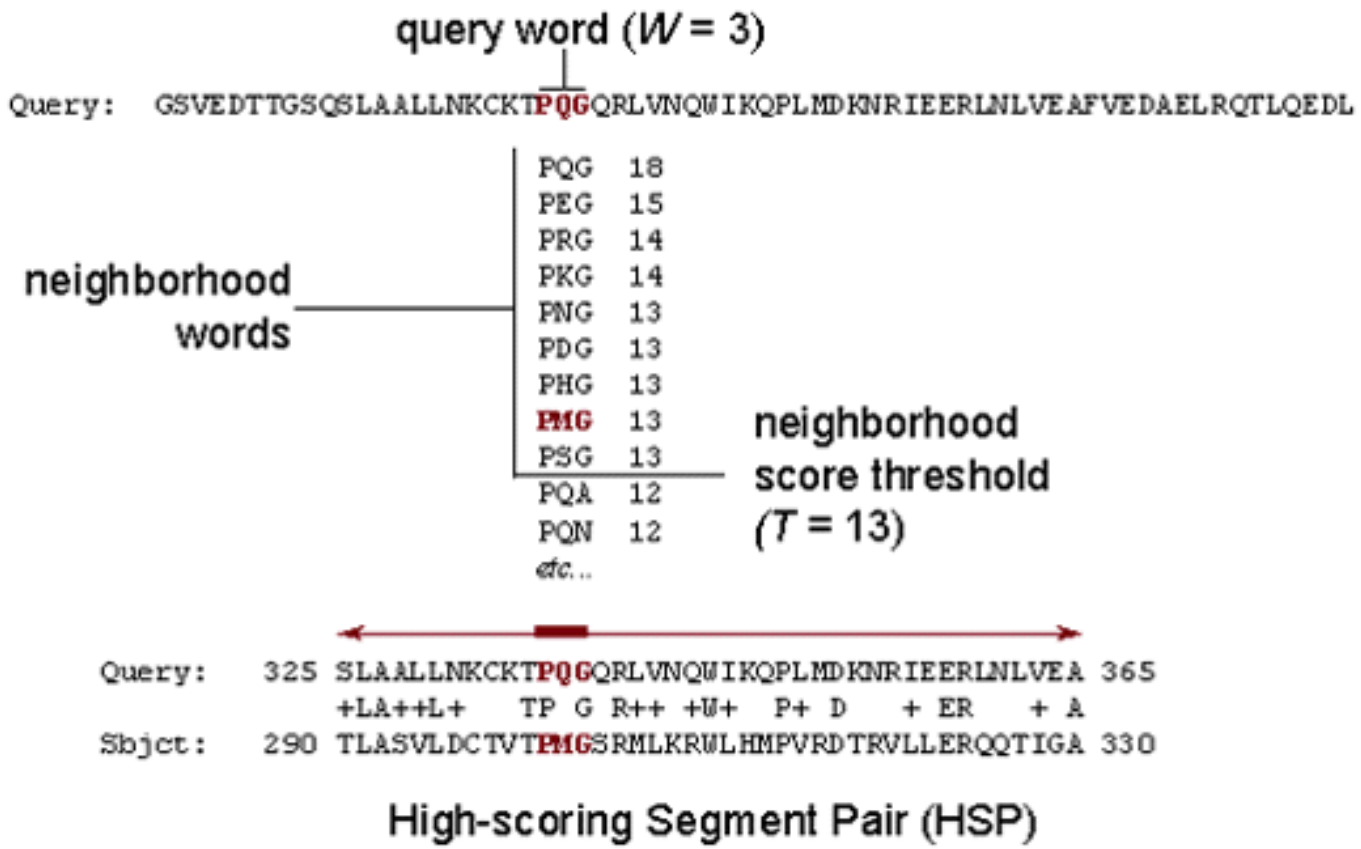
$$precisao = \frac{\#Emparelhamentos\ Significantes}{\#EmparelhamentosnaDB}$$

Ideia do BLAST

- Dada uma sequência de interrogação q tamanho de palavra w, um limite de pontuação T, e um limite de segmento S:
  - compilar uma lista de palavras que têm resultado $\geq T$ quando comparadas com palavras de q
  - percorre a BD por alinhamentos com palavras na lista
  - extender todos alinhamentos para procurar os pares de sequência com pontuação mais alta.
- resultado: pares de segmentos com resultado $\geq S$

Intuição

The BLAST Search Algorithm



Determinação de Palavras da Interrogação

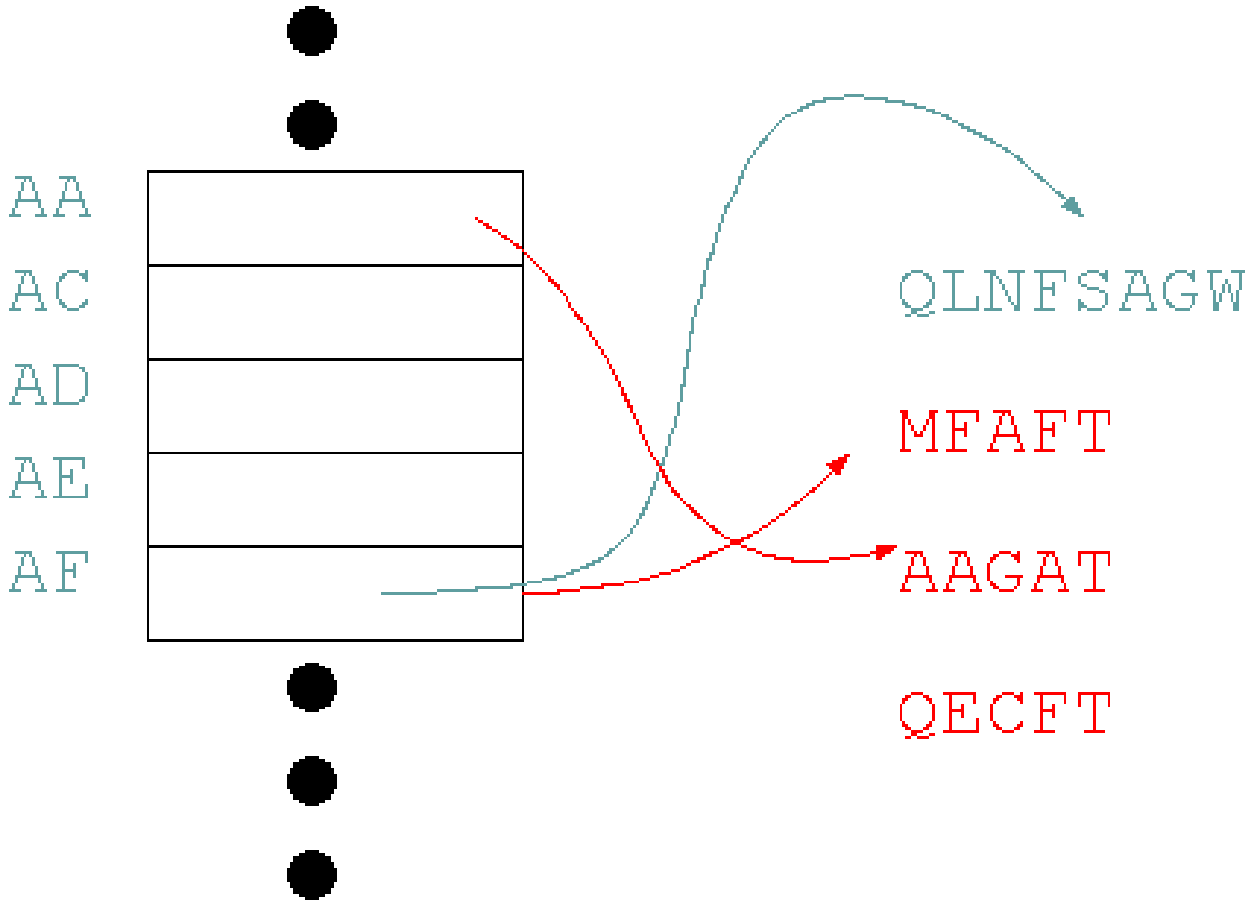
- Dada:
  - sequência de interrogação: QLNFSAGW
  - tamanho de palavra $w = 2$ (para proteína usualmente $w = 3$)
  - limite para pontuação de palavra; $T = 8$
- Passo 1: determinar todas as palavras de tamanho w na sequência de interrogação:
  - QL LN NF FS SA AG GW

Palavras Similares Query

- Passo 2
  - Procurar todas as palavras com resultado acima de limiar T
  - Usando $T = 9$
    - QL $\Rightarrow$ QL (10)
    - LN $\Rightarrow$ LN (11), LS (9)
    - FS $\Rightarrow$ FS (12)

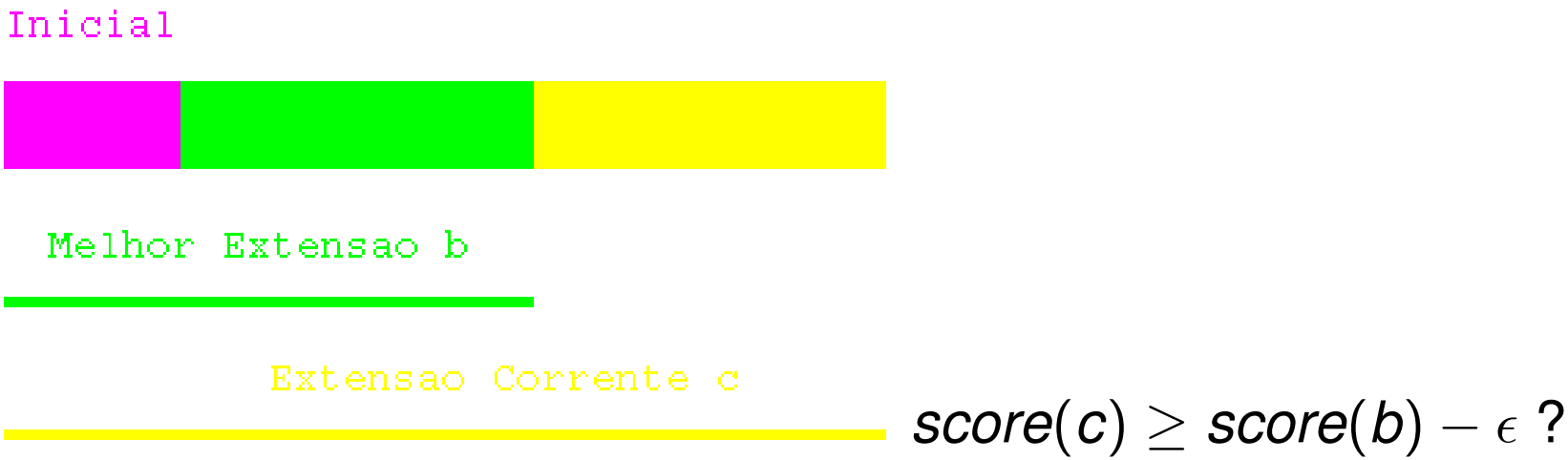
Procurando na BD

- Procurar na BD por todas as instâncias das palavras na sequência de interrogação:
- método:
  - indexar sequências na BD com *tabela de palavras*
  - procurar palavras da interrogação na tabela



Ampliar Sucessos

- Ampliar sucessos em ambas as direcções (sem permitir buracos)
- terminar a ampliação numa direcção quando a pontuação cair abaixo de certa distância abaixo pontuação óptima para pequenas extensões



$score(c) \geq score(b) - \epsilon$?

- resultado: pares de segmentos com resultado pelo menos S

P-Value

- Se na BD $\mathcal{D}$ a sequência X tem a pontuação $S(\mathcal{D}, X) = s$, então:
  - *p-value:* $P(S(D, Y)) \leq s$, onde Y é uma sequência aleatória
  - Quanto menor, melhor, eg:
    - 0.1: 1 em 10 têm pontuação $\geq$ por acaso
    - 10^{-6}: apenas 1 em 1000000!
- Como estimar *p-value* no BLAST?

Pontuação Simplificada

- Score Simplificado:
  - 1 para acerto
  - $-\infty$ para miss ou buraco
- Pontuação melhor:
  - maior alinhamento
- Se matriz de alinhamento tem tamanho $n \times m$:
  - Alinhamento pode começar $\approx n \times m$ posições

Pontuação Simplificada

- Se P(probabilidade de 2 letras à sorte serem iguais) $= p$
- Em geral, a probabilidade de alinhamento de tamanho $\geq t$:

$$(1 - p)p^t$$

  - $1 - p$: se começou aqui, antes não era alinhamento
  - $p \times \ldots \times p$
- Logo, número esperado de alinhamentos de tamanho $\geq t$:

$$\approx nm(1 - p)p^t$$

Pontuação Simplificada

- quando $n \to \infty$ e $p \to 0$ Binomial pode ser aproximada por Poisson com parâmetro $\lambda = np$
- Truque:
  - $n' = mn$
  - $p = (1-p)p^t$
- Podemos aproximar por Poisson com $\lambda =' n'p' = mn(1-p)p^t$

$$P(seq_t) = 1 - P(\neg seq_t) = 1 - \frac{\lambda^0 e^{-\lambda}}{0!} = 1 - e^{-nmp^t(1-p)}$$

- Em BLAST:

$$p - value \approx 1 - e^{-Knme^{-\lambda S}}$$

  - K e λ são parâmetros estimados.

Pontuação: Explicação

- Valor-E:

$$E = Knme^{-\lambda S}$$

- Intuição:
  - Dobrar o tamanho de qq sequência dobra o número de sequências para o score.
  - Para obter $2 \times S$ tem que duplicar o tamanho, logo E deveria decrescer exponencialmente.
  - K e λ dependem:
    - da matriz de substituição
    - da frequeência dos amino-ácidos
- Mais intuitivo do que p-value para matches fracos
  - E-value de 5, P-value de 0.993
  - E-value de 10, P-value de 0.99995
- E e p-value convergem para matches fortes

Pontuação: Bit-Score

- E-value e S dependem de K e λ

$$e^{-Knme-\lambda S}$$

- Bit-Score remove os parâmetros K e λ (divide por K, tira o algoritmo e subtrai λ):

$$S' = \frac{\lambda S - lnK}{ln2}$$

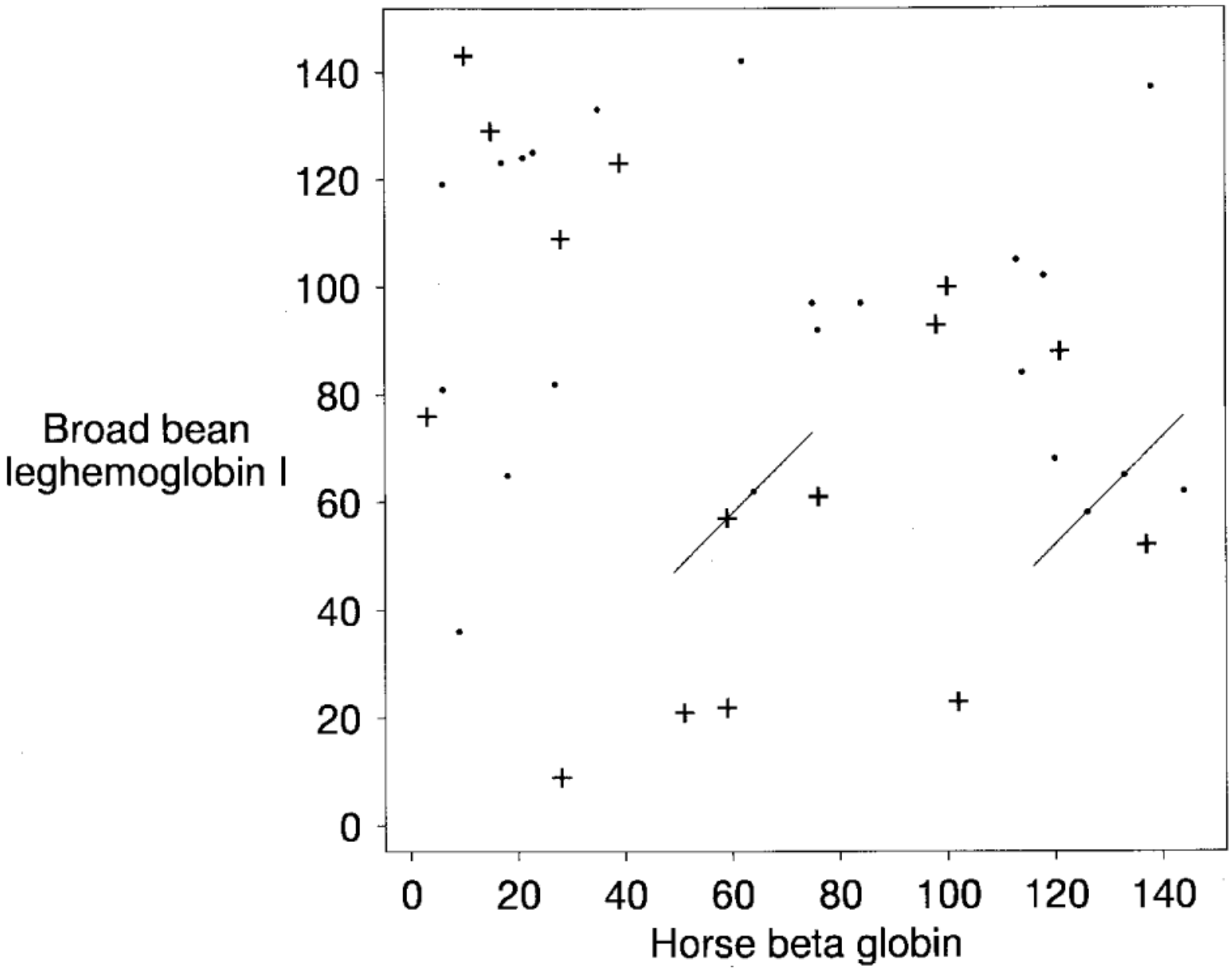
- Para entender significância precisa apenas de m, n
- Valor-E e S':

$$E = mn2^{-S'}$$

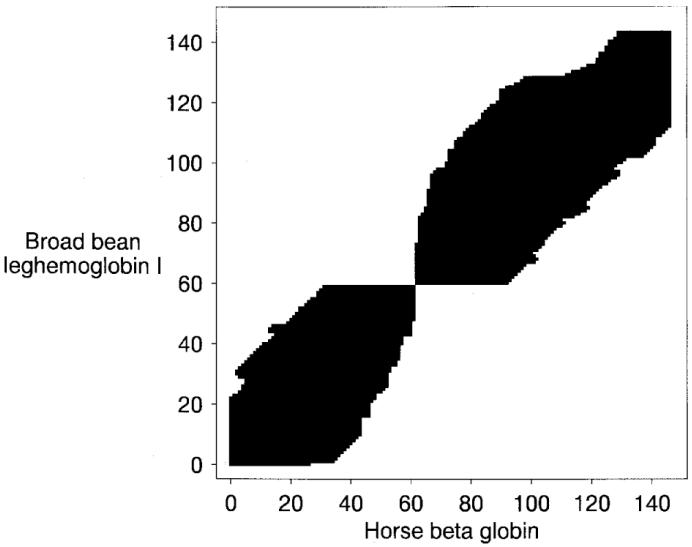
Extensões de BLAST

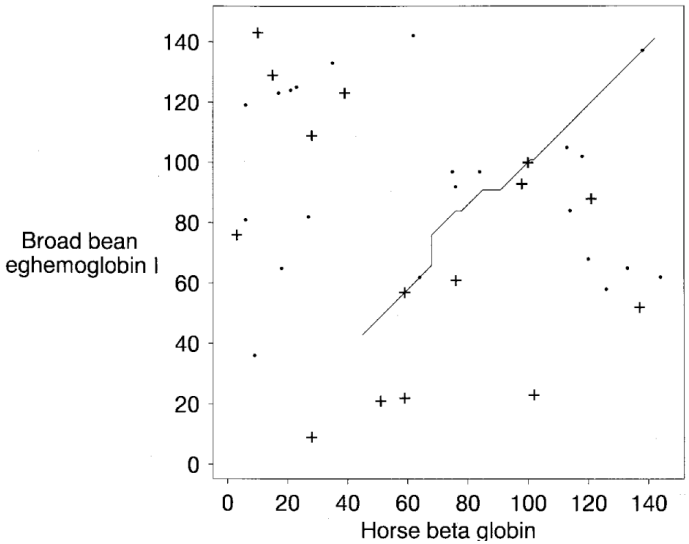
- O método *two-hit*: ampliar apenas quando há dois acertos perto e na mesma diagonal
  - permite abaixar T
- BLAST com buracos:
  - usar DP a partir de alinhamento com pontuação melhor
- PSI-BLAST: generalizar iterativamente a questão (fazê-la parecer mais como acertos) e voltar a procurar
- Todas tentam aumentar precisão enquanto limitam o tempo de execução

Método Two-Hits



Gapped Blast



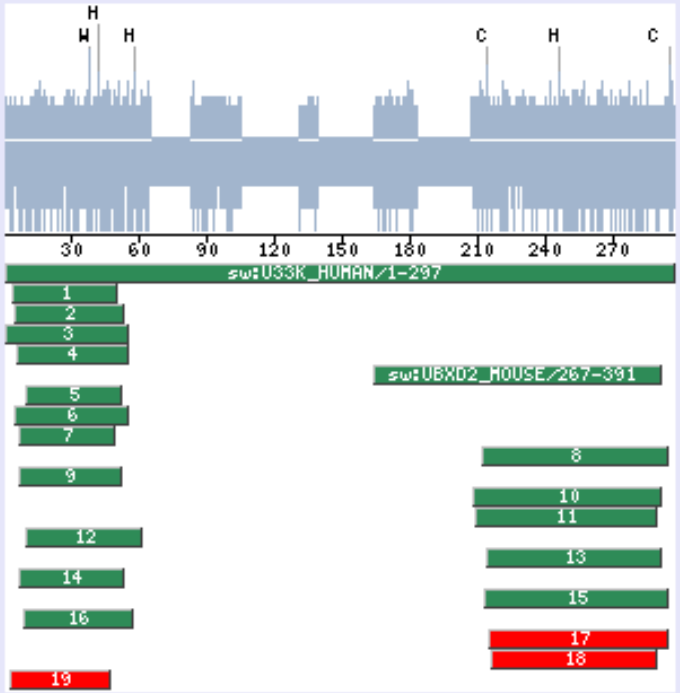



```
Leghemoglobin  43 FSFLKDSAGVVDSPKLGAHAEKVFGMVRDSAVQLRATGEVV--LDGKDGS------  90
                  F  L +   V+ +PK+ AH +KV          L + GE V  LD   G+
Beta globin    45 FGDLSNPGAVMGNPKVKAHGKKV----------LHSFGEGVHHLDNLKGTFAALSE  90

Leghemoglobin  91 IHIQKGVLDP-HFVVVKEALLKTIKEASGDKWSEELSAAWEVAYDGLATAI 140
                  +H  K  +DP +F ++   L+  +    G  ++ EL A+++    G+A A+
Beta globin    91 LHCDKLHVDPENFRLLGNVLVVVLARHFGKDFTPELQASYQKVVAGVANAL 141
```

Gapped BLAST

Matches map



Legends: **1**, sw:UAS3_DROME/22-68; **2**, sw:UBP14_SCHPO/580-628; **3**, sw:UBP5_MOUSE/654-708; **4**, sw:UBP14_YEAST/612-661; **5**, sw:UBP14_ARATH/622-664; **6**, sw:UBP13_HUMAN/656-706; **7**, sw:UBPA_DICDI/634-676; **8**, sw:YOJ8_CAEEL/281-359; **9**, sw:UAS3_HUMAN/25-70; **10**, sw:UBX7_YEAST/211-290; **11**, sw:UBXD1_HUMAN/332-406; **12**, sw:UPL2_ARATH/1280-1332; **13**, sw:UBX6_YEAST/191-264; **14**, sw:UBP14_SCHPO/645-690; **15**, sw:UBXD7_HUMAN/412-488; **16**, sw:UBP5_MOUSE/730-777; **17**, sw:FAF1_MOUSE/574-648; **18**, sw:UBAX1_ARATH/350-418; **19**, sw:UBP13_HUMAN/731-775.

Papers sobre extensões do BLAST

- Altschul, S.F., Madden, T.L., et al. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25 (17), 3389-3402.
- Zhang, Z., Schwartz, S., Wagner, L. Miller, W. A greedy algorithm for aligning DNA sequences. J. Computational Biology (2000) 7:203-214.
- Schaffer, A.A., Aravind, L., Madden, T.L., Shavirin, S., Spouge, J.L., Wolf, Y.I., Koonin, E.V., Altschul, S.F. (2001) Nucleic Acids Research, July 15;29(14):2994-3005 Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements.

Comentários sobre BLAST

- é uma heurística: pode não encontrar alguns bons resultados
- rápido: empiricamente é de 10 a 50 vezes mais rápido do que Smith-Waterman
- Tem grande impacto:
  - o servidor do NCBI recebe mais de 100.000 interrogações por dia
  - é o programa mais usado em bioinformática

Parâmetros Default do BLAST

- *M*: ganho por match de nucleotídos (5)
- *N*: perca por mismatch de DNA (-4)
- Buracos: usa afim com 11/1
- Matriz BLOSUM62
- Tamanho de palavra:
  - 11 para nucleotídos
  - 3 para AAs

Conclusões

- apresentamos alinhamentos: *locais* e *globais*
- o algoritmo exacto com DP depende de ser local/global e da função da penalização de buracos
- ao permitir buracos permitimos um número exponencial de alinhamentos
- com programação dinâmica a complexidade é $O(mn)$
- algoritmos funcionam tanto para proteínas como DNA
- heurísticas como BLAST são mais rápidas mas não tão precisas.

Modelos Para Alinhamentos

- Na aula passada usamos pontuação como $\equiv$ comprimento
- Não faz sempre sentido
- Como avaliar alinhamento?
  - Probabilidades do Alinhamento
- Vamos assumir que cada posição é independente

Modelos de Sequências

- Independentes:

$$Pr(x, y|U) = \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} q_{y_i}$$

- Dependentes:

$$Pr(x, y|R) = \prod_{i=1}^{n} p_{x_i, y_i}$$

Comparando Modelos

- Qual o mais provável?
- Comparar a verosimilhança dos 2:

$$\frac{Pr(x,y|R)}{Pr(x,y|U)} = \frac{\prod_{i=1}^{n} p_{x_i,y_i}}{\prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{n} bq_{y_i}} = \prod_{i=1}^{n} \frac{p_{x_i,y_i}}{q_{x_i} q_{y_i}}$$

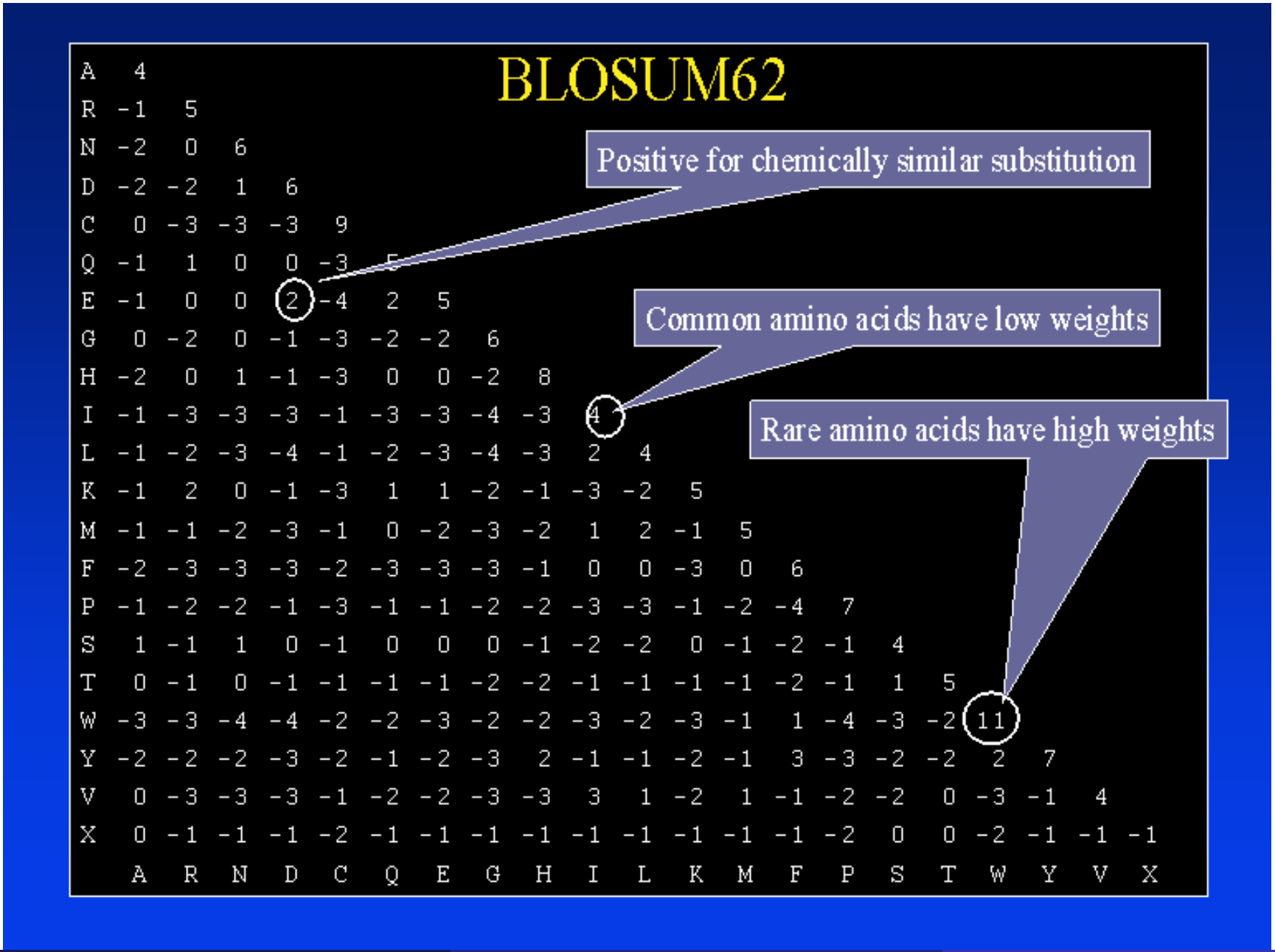
- É mais fácil trabalhar com o log:

$$log \frac{Pr(x,y|R)}{Pr(x,y|U)} = \sum_{i} \frac{p_{x_i,y_i}}{q_{x_i} q_{y_i}}$$

Matrizes

- PSSM (PSI-BLAST)
  - Estimar uma probabilidade do AA em cada coluna
  - Dividir pela probabilidade do AA
  - Tirar log: se positivo é provável
- Matrizes de Substituição:
  - PAM [Dayhoff et al, 1978]
  - BLOSUM [Henikoff & Henikoff, 1992]

BLOSUM62


BLOSUM62
Positive for chemically similar substitution
Common amino acids have low weights
Rare amino acids have high weights
A 4
R -1 5
N -2 0 6
D -2 -2 1 6
C 0 -3 -3 -3 9
Q -1 1 0 0 -3 5
E -1 0 0 2 -4 2 5
G 0 -2 0 -1 -3 -2 -2 6
H -2 0 1 -1 -3 0 0 -2 8
I -1 -3 -3 -3 -1 -3 -3 -4 -3 4
L -1 -2 -3 -4 -1 -2 -3 -4 -3 2 4
K -1 2 0 -1 -3 1 1 -2 -1 -3 -2 5
M -1 -1 -2 -3 -1 0 -2 -3 -2 1 2 -1 5
F -2 -3 -3 -3 -2 -3 -3 -3 -1 0 0 -3 0 6
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4 7
S 1 -1 1 0 -1 0 0 0 -1 -2 -2 0 -1 -2 -1 4
T 0 -1 0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1 1 5
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1 1 -4 -3 -2 11
Y -2 -2 -2 -3 -2 -1 -2 -3 2 -1 -1 -2 -1 3 -3 -2 -2 2 7
V 0 -3 -3 -3 -1 -2 -2 -3 -3 3 1 -2 1 -1 -2 -2 0 -3 -1 4
X 0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2 0 0 -2 -1 -1 -1
A R N D C Q E G H I L K M F P S T W Y V X

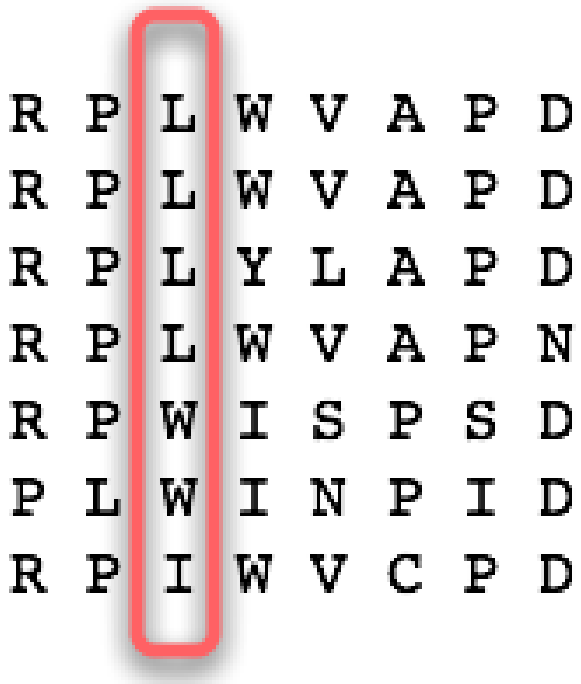
BLOSUM62

- Começar por BD com segmentos bem conservados
- Exemplo: BLOCKS

```
ID   XRODRMPGMNTB; BLOCK
AC   PR00851A; distance from previous block=(52,131)
DE   Xeroderma pigmentosum group B protein signature
BL   adapted;   width=21; seqs=8; 99.5%=985; strength=1287
XPB_HUMAN|P19447   (  74) RPLWVAPDGHIFLEAFSPVYK  54
XPB_MOUSE|P49135   (  74) RPLWVAPDGHIFLEAFSPVYK  54
P91579             (  80) RPLYLAPDGHIFLESFSPVYK  67
XPB_DROME|Q02870   (  84) RPLWVAPNGHVFLESFSPVYK  79
RA25_YEAST|Q00578  ( 131) PLWISPSDGRIILESFSPLAE 100
Q38861             (  52) RPLWACADGRIFLETFSPLYK  71
O13768             (  90) PLWINPIDGRIILEAFSPLAE 100
O00835             (  79) RPIWVCPDGHIFLETFSAIYK  86
//
```

BLOSUM: Pares

- Contar Pares por Colunas
- $L/L : 3 + 2 + 1 = 6$
- $L/W : 4 + 4 = 8$
- $L/I : 4$
- $W/W : 1$

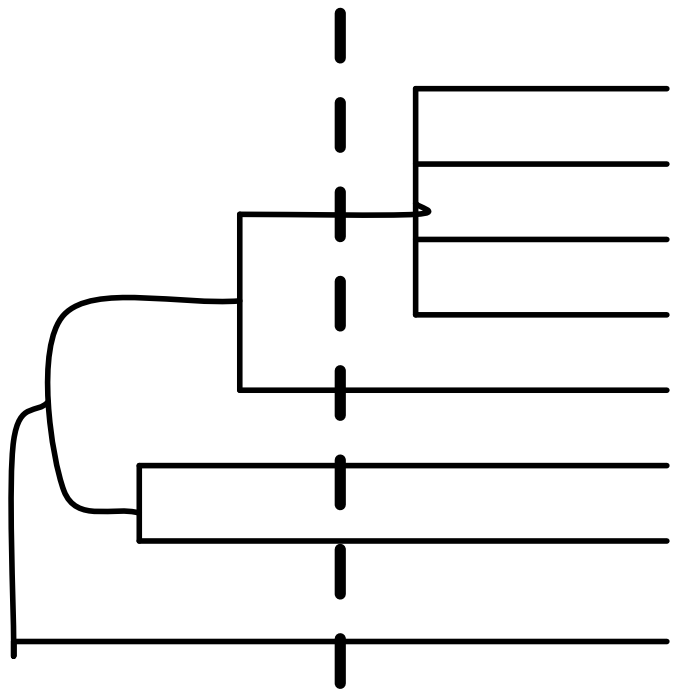


BLOSUM: Algoritmo

- Contar Pares f_{ab} em Todas Colunas da BD (> 20000)
- Calcular $p_{ab} = \frac{f_{ab}}{\sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}}$
- Calcular $q_a = \sum_{b=1}^{20} (\frac{f_{ab}}{\sum_{i=1}^{20} \sum_{j=1}^{i} f_{ij}})$
- $s(a, b) = log_2(\frac{p_{ab}}{q_a q_b})$
- Fazer arredondamento e ajustar (half-bits)

BLOSUM*XX*: Distância Evolutiva

- Agrupar sequências que estão perto evolucionariamente
- Juntos os elementos dos cluster $>$ *XX* valem 1



BLOSUM: Análise

- A melhor pontuação é para W (triptofano)
  - Relativamente raro
- O pior para * com AA, e exs como D e L
  - Ácido aspártico está a 2 mutações de Leucina
  - Polaridade e Carga Diferentes

BLOSUM vs PAM

- Menos tolerante de substituições de hidrfílicos
- Mais tolerante de mudanças entre hidrofibicidade
- Mais tolerante de falhas com cisteína e triptofan.

Alinhamentos Múltiplos

- caracterizar um conjunto de sequências (ie, uma classe de sinais DNA)
- caracterizar uma família de proteínas:
  - o que é conservado
  - o que varia
- geração de perfis para procura

Uma matriz de perfis

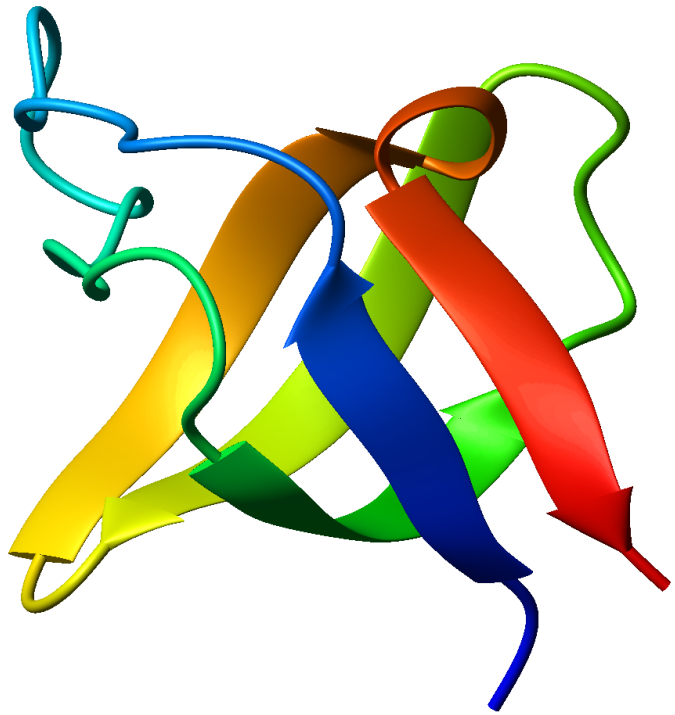
- um perfil é uma descrição de um conjunto de sequências
- colunas representam posições em sequências
- linhas representam caracteres em sequências
- elementos representam a abundância de um caracter numa posição

amino – ácidos

	1	2	3	4	5	6	7	8	•••
A			0						
R			0						
D			0.5						
N			0.2						
C			0						
						•			

SH3

- Domínio envolvido em tradução de sinal para cito-esqueleto, RAS
- Estrutura típica de barril-beta parcialmente aberto
- $\approx$ 60 AAs



Alinhamento Múltiplo em SH3

```
G G W W R G d y . g g k k q L W F P S N Y V
I G W L N G y n e t t g e r G D F P G T Y V
P N W W E G q l . . n n r r G I F P S N Y V
D E W W Q A r r . . d e q i G I V P S K - -
G E W W K A q r . . t g q e G F I P F N F V
G D W W L A r s . . s g q t G Y I P S N Y V
G D W W D A e l . . k g r r G K V P D N Y L
- D W W E A r s l s s g h r G Y V P S N Y V
G D W W Y A r s l i t n s e G Y I P S T Y V
G E W W K A r s l a t r k e G Y I P S N Y V
G D W W L A r s l v t g r e G Y V P S N F V
G E W W K A q t . k n g q . G W V P S N Y I
S D W W R V v n l t t r q e G L I P L N F V
L P W W R A r d . k n g q e G Y I P S N Y I
R D W W E F r s k t v y t p G Y Y E S G Y V
E H W W K V k d . q l g n v G Y I P S N Y V
I H W W R V q d . r n g h e G Y V O S S Y L
K D W W K V e v . . n d r q G F V P A A Y V
V G W M P G l n e r t r q r G D F P G T Y V
P D W W E G e l . . n g q r G V F P A S Y V
E N W W N G e i . . g n r k G I F P A T Y V
E E W L E G e c . . k g k v G I F P K V F V
G G W W K G d y . g t r i q Q Y F P S N Y V
D G W W R G s y . . n g q v G W F P S N Y V
Q G W W R G e i . . y g r v G W F P A N Y V
G R W W K A r r . a n g e t G I I P S N Y V
G G W T Q G e l . k s g q k G W A P T N Y L
G D W W E A r s n . t g e n G Y I P S N Y V
N D W W T G r t . . n g k e G I F P A N Y V
```

Avaliação de Alinhamentos Múltiplos

- Questão Principal: como estimar a qualidade de um alinhamento entre sequências múltiplas?
- Assumimos que as *colunas* individuais de alinhamentos são independentes.
- Discutiremos dois métodos:
  - Soma de Pares (SP)
  - Entropia Mínima

Soma de Pares (SP)

- Computar a soma das pontuações entre pares:

$$Score(m_i) = \sum_{k<l} s(m_i^k, m_i^l)$$

- m_i^k = caracter da sequência k e coluna i
- s = matriz de substituição

Entropia Mínima

- Ideia: **minimizar** a *entropia* de cada coluna
- Ou coluna que pode ser apresentada com menos bits de informação é melhor
- Teoria da Informação: um código óptimo usa $-\log_2 p$ para codificar uma mensagem de probabilidade p.

Entropia Mínima

- Neste caso as mensagens são os caracteres numa certa coluna
- a entropia de uma coluna é dada por:

$$Score(m_i) = -\sum_a c_{ia} \log_2(p_{ia})$$

- m_i = a coluna i de um alinhamento m
- c_{ia} = número de caracteres a na coluna i
- p_{ia} = probabilidade do caracter a na coluna i

Programação Dinâmica

- Pode-se encontrar alinhamentos óptimos usando programação dinâmica
- Generalização de métodos para alinhamento de pares:
  - Matriz de dimensão-k para k sequências (substituindo uma matriz bidimensional)
  - cada entrada na matriz representa um alinhamento para k subsequências (em vez de 2 subsequências)
- dadas k sequências de tamanho n
  - Complexidade espacial é:

$$O(n^k)$$

Programação dinâmica

- Dadas k sequências de tamanho n:
  - Complexidade temporal é:

$$\begin{cases} O(k^2 2^k n^k) & \text{se usarmos SP} \\ O(k 2^k n^k) & \text{se as pontuações de colunas puderem ser computadas} \end{cases}$$

Métodos Heurísticos para Alinhamento

- Como a complexidade de DP é exponencial...
- *Alinhamento Progressivo*: construa uma sucessão de alinhamentos entre pares:
  - CLUSTALW
  - estrela
  - ...
- Refinamento Iterativo:
  - dado um alinhamento (eg, dado por um método progressivo)
    - remover uma sequência, realinhá-la ao perfil de outras sequências
    - repetir até convergir.

Alinhamento em Estrela

- dadas: k sequências para serem alinhadas,

$$x_1, \ldots, x_k$$

  - seleccione uma sequência x_c como sendo o *centro*
  - para cada sequência x_i determine um alinhamento óptimo entre x_i e x_c
  - agregar alinhamentos entre pares
- resultado: alinhamentos múltiplos resultando do agregado

Estrela: O Centro

- tente cada sequência como o centro, retornar o melhor alinhamento múltiplo
- computar todos os alinhamentos entre pares e seleccionar a sequência x_c que maximize:

$$\sum_{i \neq c} sim(x_i, x_c)$$

Estrela:Agregação

- Se um buraco, sempre buraco
- Deslocar colunas inteiras quando se incorporam buracos.

Estrela: Exemplo

Dados:

1. ATTGCCATT
2. ATGGCCATT
3. ATCCAATTTT
4. ATCTTCTT
5. ATTGCCGATT

Estrela: Alinhamentos

1. ATTGCCATT
   ATGGCCATT
2. ATTGCCATT--
   ATC-CAATTTT
3. ATTGCCATT
   ATCTTC-TT
4. ATTGCC-ATT
   ATTGCCGATT

Estrela: Junção

1. ATTGCCATT
   ATGGCCATT

   ATTGCCATT
   ATGGCCATT

2. ATTGCCATT--
   ATC-CAATTTT

   ATTGCCATT--
   ATGGCCATT--
   ATC-CAATTTT

3. ATTGCCATT
   ATCTTC-TT

   ATTGCCATT--
   ATGGCCATT--
   ATC-CAATTTT
   ATCTTC-TT--

4. ATTGCC-ATT
   ATTGCCGATT

   ATTGCC-ATT--
   ATGGCC-ATT--
   ATC-CA-ATTTT
   ATCTTC--TT--
   ATTGCCGATT-

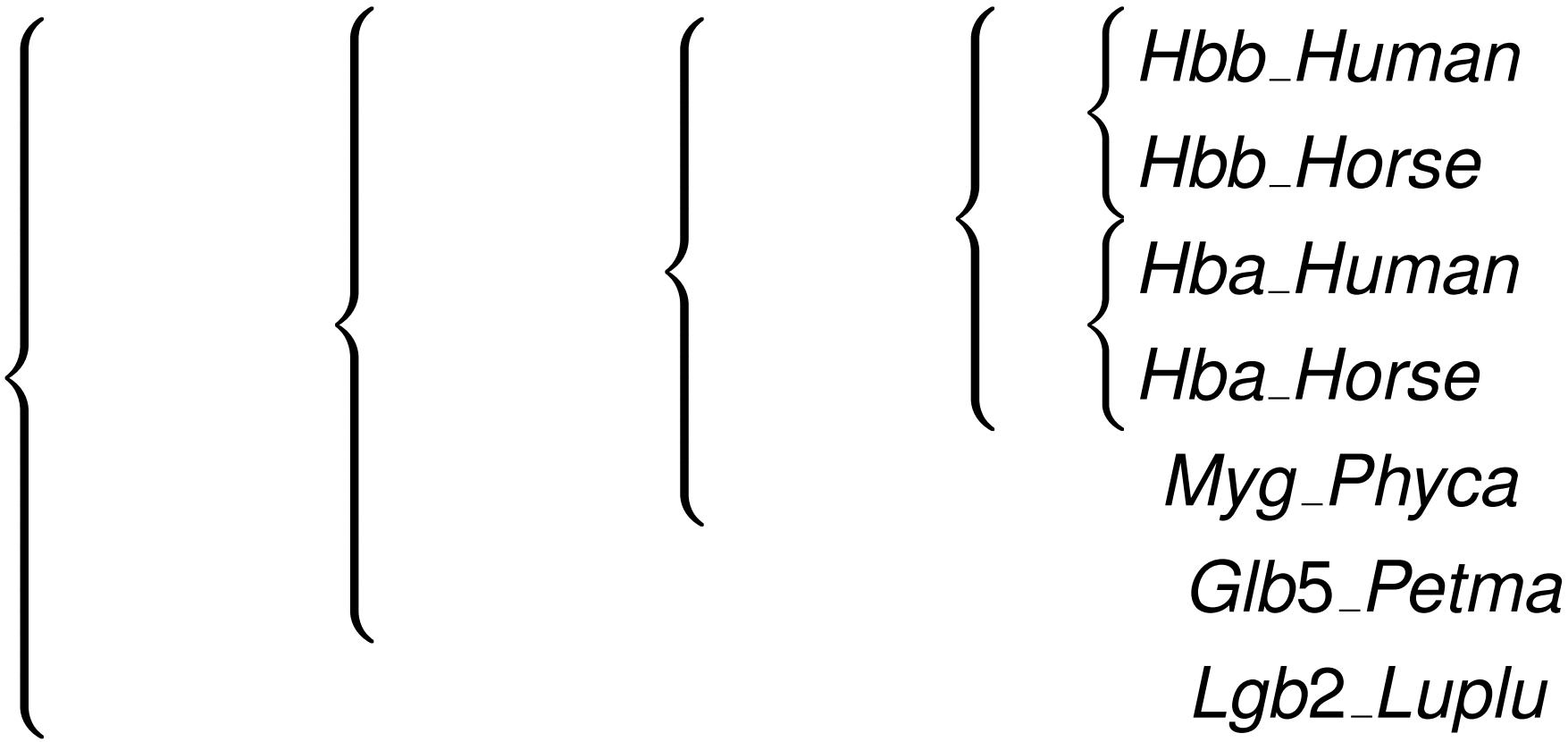
Alinhamento em Árvore

- Ideia básica: organizar alinhamentos múltiplos de sequências usando uma *árvore guia*
  - folhas representam *sequências*
  - nós internos representam alinhamentos
- falaremos sobre algoritmos para determinar árvores mais tarde
- determinar alinhamentos desde o fundo da árvore para cima
- variante comum: o algoritmo CLUSTALW de [Thompson et al. 1994].

Ideias de CLUSTALW

- dadas: k sequências a alinhar
  - construir a matriz de distância de todos os pares usando DP entre os pares
  - converter medidas de semelhança em distâncias
  - construir uma árvore guia das distâncias
  - alinhar os nós internos progressivamente em ordem de semelhança decrescente
- resultado: alinhamentos múltiplos na raíz da árvore

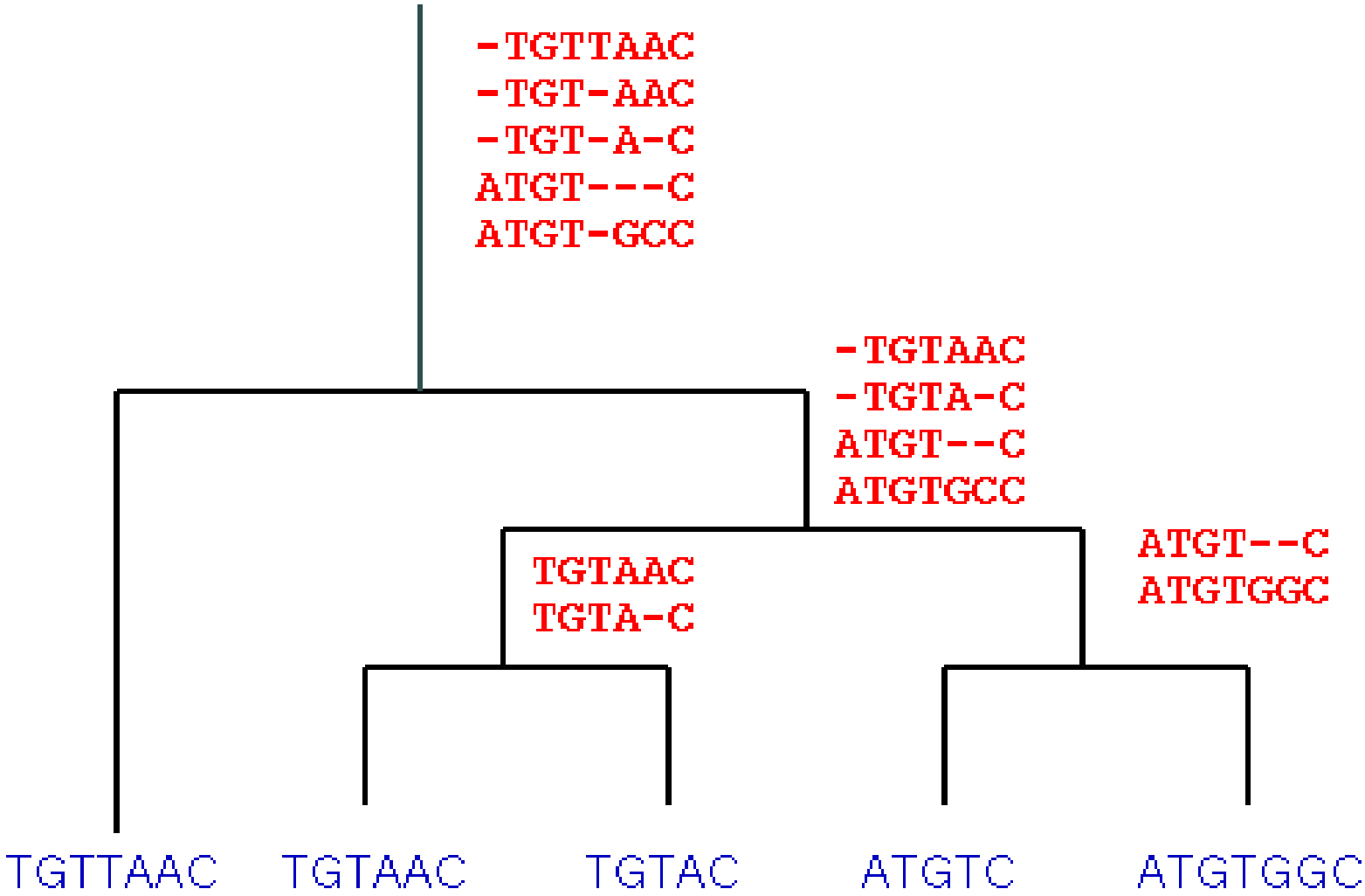
Exemplo de Árvore Guia



Alinhamento Progressivo em CLUSTALW

- dependendo do nó interno na árvore, podemos ter que alinhar:
  - uma sequência com uma sequência
  - uma sequência com um *perfil*
  - um *perfil* com um *perfil*
- em todos os casos podemos usar programação dinâmica
  - no caso de perfis, usamos Soma de Pares

SH3


-TGTTAAC
-TGT-AAC
-TGT-A-C
ATGT---C
ATGT-GCC
-TGTAAC
-TGTA-C
ATGT--C
ATGTGCC
TGTAAC
TGTA-C
ATGT--C
ATGTGGC
TGTTAAC
TGTAAC
TGTAC
ATGTC
ATGTGGC

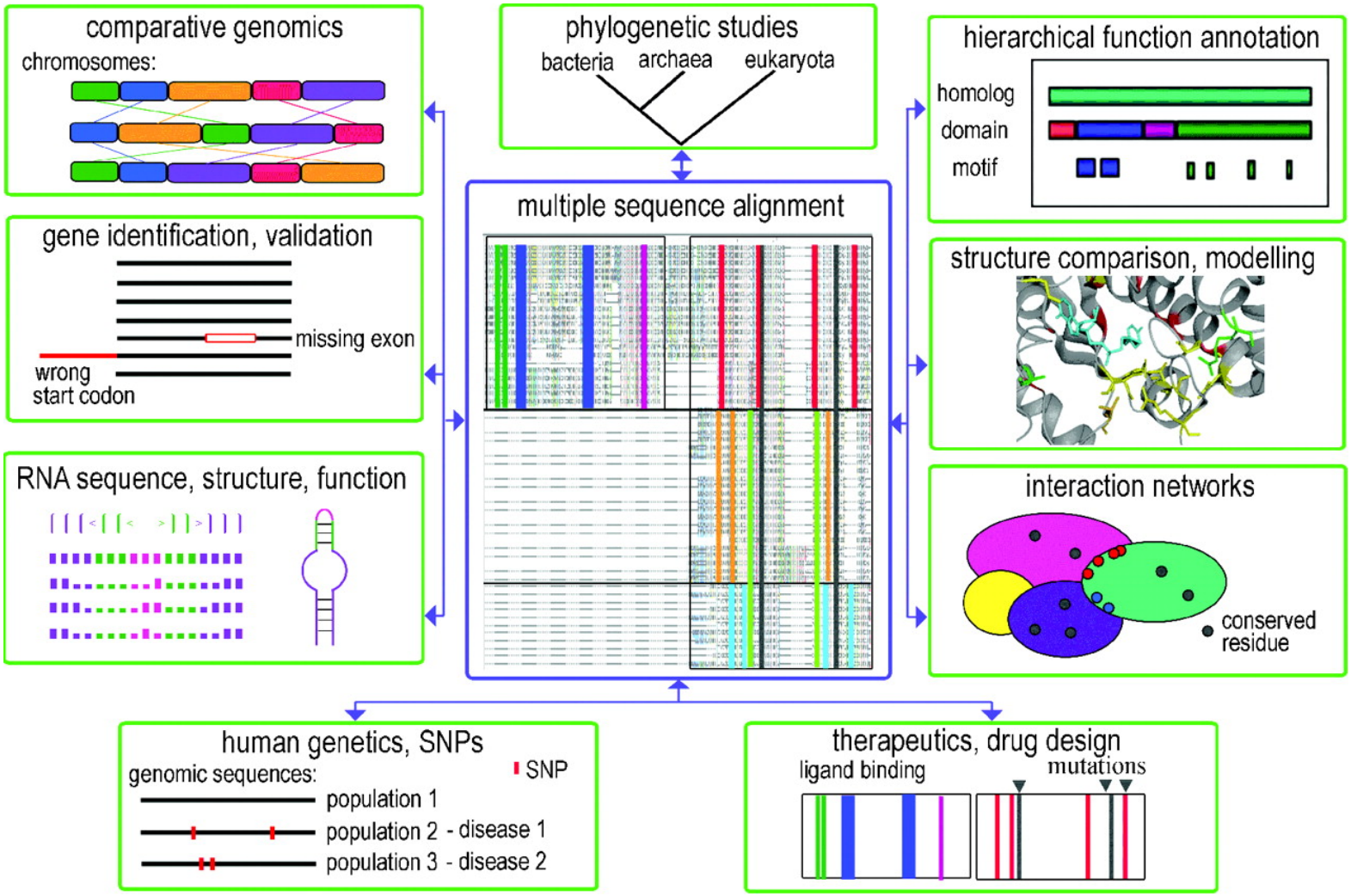
Outras Variações

- Alinhamento de perfis baseado em alinhamentos de pares
  - considere todos os alinhamentos de pares
  - deixar que o melhor alinhamento entre pares determine o alinhamento de sequência múltiplos
- Refinamento iterativo
  - dado um alinhamento múltiplo
    - remover uma sequência, realinhá-la ao perfil das outras sequências
    - repetir até convergir (ou até ao computador cansar)

Métodos para Alinhamento de Múltiplas Sequências

método	**tipos de alinhamento**	**procura**
programação dinâmica multi-dimensional	global/local	programação dinâmica
Estrela	global	guloso com alinhamento de pares
CLUSTALW (árvore)	global	guloso com alinhamento de pares
HMMs com perfis	global/local	Baum-Welch (EM) para aprender modelo e Viterbi recuperar alinhamentos
EM/MEME	local	EM

Aplicações de Alinhamento de Múltiplas Sequências [Thompson et al 2005]



Exemplo de Sistemas

- CLUSTALW
- T-COFFEE
- ALIGN-M
- MAFFT
- MUSCLE
- PROBCONS

Definição de Probabilidade

- *Interpretação Frequentista:* a probabilidade de um evento é a proporção de eventos temporais desse tipo que vão ocorrer passado muito tempo.
- Exemplos:
  - a probabilidade que o meu voo para o Porto saia a tempo
  - a probabilidade que o bilhete ganhe a lotaria
  - a probabilidade que chova amanhã
- Sempre um número entre $[0, 1]$
  - 0 significa que nunca vai acontecer
  - 1 significa que acontece sempre

Espaço de Amostragem

- *Espaço de Amostragem*: o conjunto de resultados possíveis para o evento
- exemplos:
  - voo para o Porto: {certo, atrasado }
  - lotaria: {bilhete 1 ganha, bilhete 2 ganha,. . .,bilhete *n* ganha}
  - tempo amanhã:
    - {chove,não chove}
    - {sol, chuva}
    - {sol, nublado, chuvisco, chuva, tempestade}

Variáveis Aleatórias

- *Variável Aleatória:* a variável representa o resultado de uma experiência
- exemplo:
  - X representa o resultado do meu voo para o Porto
  - escrevemos a probabilidade do voo estar certo como $P(X = acerto)$
  - quando for claro no que estamos a pensar, podemos escrever $P(acerto)$

Notação

- Letras maiúsculas e palavras capitalizadas denotam variáveis aleatórias
- Letras minúsculas e palavras não-capitalizadas denotam valores
- vamos escrever um valor para uma variável como sendo:

$$P(X = x) \qquad P(Febre = \texttt{verdade})$$

- Podemos usar uma notação compacta

$$P(x) \text{ em vez de } P(X = x)$$

- para variáveis booleanas, podemos usar:
  - $P(\texttt{febre})$ para $P(Febre = \texttt{verdade})$
  - $P(\neg\texttt{febre})$ para $P(Febre = \texttt{falso})$

Distribuição de Probabilidade

- Se X for uma variável aleatória, a função dada por $P(X = x)$ para cada x é a distribuição de probabilidade de X
- Requisitos:
  - $P(X = x) \geq 0$ para qualquer x
  - $\sum_x P(x) = 1$

Distribuição Conjunta

- *Probabilidade Conjunta* a função dada por $P(X = x, Y = y)$
- ler como "X vale x e Y vale y"
- exemplo:

x, y	$P(X = x, Y = y)$
sol, certo	0.20
chuva, certo	0.30
sol, atrasado	0.40
chuva, atrasado	0.10

Distribuição Marginal

- a *distribuição marginal* de X é definida como:

$$Pr(x) = \sum_{y} P(x, y)$$

  "a distribuição de X ignorando outras variáveis"
- Esta distribuição generaliza para mais do que 2 variáveis, e.g.

$$P(x) = \sum_{y} \sum_{z} P(x, y, z)$$

Exemplo de Distribuição Marginal

Distribuição Conjunta

x, y	$P(X = x, Y = y)$
sol, certo	0.20
chuva, certo	0.30
sol, atrasado	0.40
chuva, atrasado	0.10

Distribuição Marginal para X

x	$P(X = x)$
sol	0.60
chuva	0.40

Distribuições Condicionais

- A *distribuição condicional* de X dado Y é definida como:

$$P(X = x | Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)}$$

"A distribuição de X dado que sabemos Y"

Exemplo de Distribuição Condicional

Distribuição Conjunta

x, y	$P(X = x, Y = y)$
sol, certo	0.20
chuva, certo	0.30
sol, atrasado	0.40
chuva, atrasado	0.10

Distribuição Condicional para X dado que $Y =$ certo

x	$P(X = x \mid Y = \texttt{certo})$
sol	0.20/0.50 = 0.40
chuva	0.30/0.50 = 0.60

Independência

Duas variáveis aleatórias X e Y são *independentes* se

$$P(x, y) = P(x) \times P(y) \quad \text{para qualquer } x \text{ e } y$$

Exemplo de Distribuição Marginal

Distribuição Conjunta

x, y	$P(X = x, Y = y)$
sol, certo	0.20
chuva, certo	0.30
sol, atrasado	0.40
chuva, atrasado	0.10

Distribuição Marginal para X

x	$P(X = x)$
sol	0.60
chuva	0.40

y	$P(Y = y)$
sol	0.50
chuva	0.50

Independência Condicional

- Duas variáveis X e Y são condicionalmente independentes dado Z se

$$P(X|Y,Z) = P(X|Z)$$

"se soubermos o valor de Z, saber Y não faz diferença para saber X"

- Alternativamente:

$$P(X,Y|Z) = P(X|Z) \times P(Y|Z) \text{ para qualquer } `x, y, z$$

Exemplo de Independência Condicional

Gripe	Febre	Vomitar	**Pr**
V	V	V	0.04
V	V	F	0.04
V	F	V	0.01
V	F	F	0.01
F	V	V	0.009
F	V	F	0.081
F	F	V	0.081
F	F	F	0.729

- Febre e Vomitar não são independentes:
  $P(\texttt{febre}, \texttt{vomitar}) \neq P(\texttt{febre}) \times P(\texttt{vomitar})$
- Febre e vomitar são condicionalmente independentes dado gripe:
  - $P(\texttt{febre}, \texttt{vomitar} | \texttt{gripe}) \neq$
    $P(\texttt{febre} | \texttt{gripe}) \times P(\texttt{vomitar} | \texttt{gripe})$
  - $P(\texttt{febre}, \texttt{vomitar} | \neg\texttt{gripe}) \neq$
    $P(\texttt{febre} | \neg\texttt{gripe}) \times P(\texttt{vomitar} | \neg\texttt{gripe})$

Teorema de Bayes

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} = \frac{P(y|x)P(x)}{\sum_x P(y|x)P(x)}$$

- Este teorema é extremamente útil
- Há muitos casos em que é difícil estimar $P(x|y)$, mas não é difícil estimar $P(y|x)$ e $P(x)$.

Exemplo do Teorema de Bayes

- Habitualmente médicos não conseguem estimar facilmente $P(Doenca|Sintoma)$
- É mais fácil para eles estimar $P(Sintoma|Doenca)$
- Se pudermos estimar $P(\texttt{febre}|\texttt{gripe})$ e $P(\texttt{gripe})$ podemos usar o teorema de Bayes para fazer o diagnóstico:

$$P(\texttt{gripe}|\texttt{febre})$$

$$=$$

$$\frac{P(\texttt{febre}|\texttt{gripe})P(\texttt{gripe})}{P(\texttt{febre}|\texttt{gripe})P(\texttt{gripe}) + P(\texttt{febre}|\neg\texttt{gripe})P(\neg\texttt{gripe})}$$

Valores Esperados

- O *valor esperado* de uma variável aleatória que toma valores numéricos é definido como:

$$E[X] = \sum_x x \times P(x)$$

  É o mesmo que a média
- Podemos também falar do valor esperado de um função de variável aleatória:

$$E[g(X)] = \sum_x g(x) \times P(x)$$

Exemplo de Valor Esperado

- Clientes de Sapataria:

$$E[\mathit{TamanhoDoPe}] = 38 \times P(\mathit{TamanhoDoPe} = 38) + \ldots$$

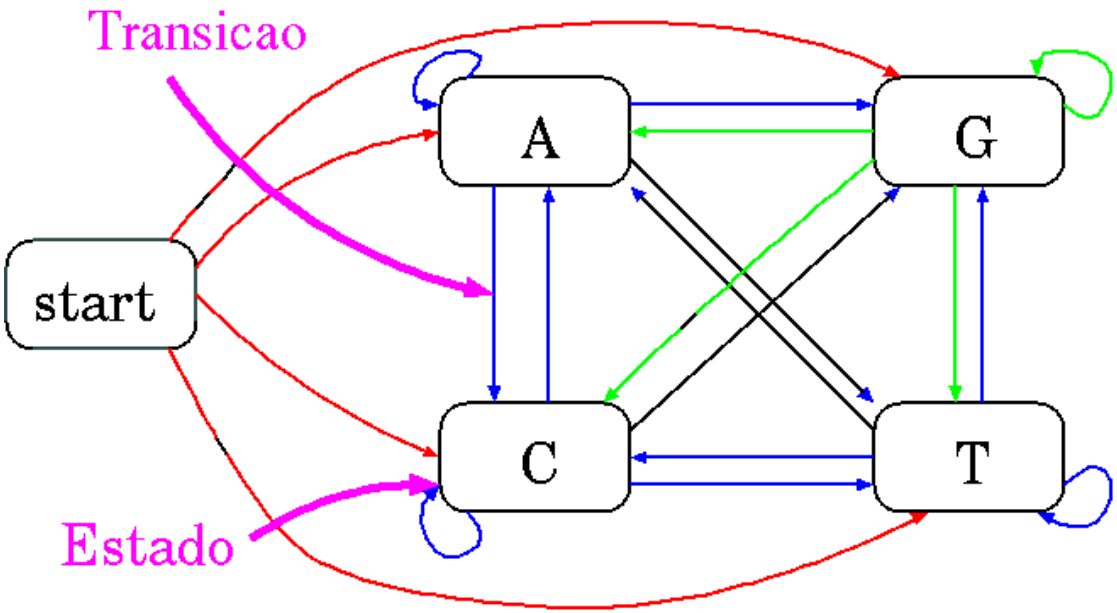
- Lotaria Simples:

$$\begin{aligned} E[\mathit{lucro}(\mathit{Lotaria})] \quad & = \\ & \mathit{ganho}(\texttt{premiado})P(\texttt{premiado}) \\ & + \mathit{ganho}(\neg\texttt{premiado})P(\neg\texttt{premiado}) \\ & = \\ & \$100 \times 0.001 - \$1 \times 0.999 \\ & = \\ & -\$0.8999 \end{aligned}$$

Cadeias de Markov

- Há muitos casos em que queremos representar regularidades estatísticas de certos tipos de sequências:
  - genes
  - locais reguladores em DNA (eg, onde a polimerase do RNA e os factores de transcrição associam)
  - proteínas de uma família
- Modelos de Markov são muito apropriados para este tipo de tarefa.

Cadeias de Markov



probabilidades de transição

$$P(x_i = a|x_{i-1} = g) = 0.16$$
$$P(x_i = c|x_{i-1} = g) = 0.34$$
$$P(x_i = g|x_{i-1} = g) = 0.38$$
$$P(x_i = t|x_{i-1} = g) = 0.12$$

Modelos de Cadeia de Markov

- Um modelo de cadeia de Markov é definido como:
  - um conjunto de estados
  - alguns estados emitem símbolos
  - outros estados (eg, estado `begin`) são silenciosos
  - um conjunto de transições com probabilidades associadas
- as transições saindo de um estado definem uma distribuição sobre os estados seguintes possíveis.

Modelos de Cadeia de Markov

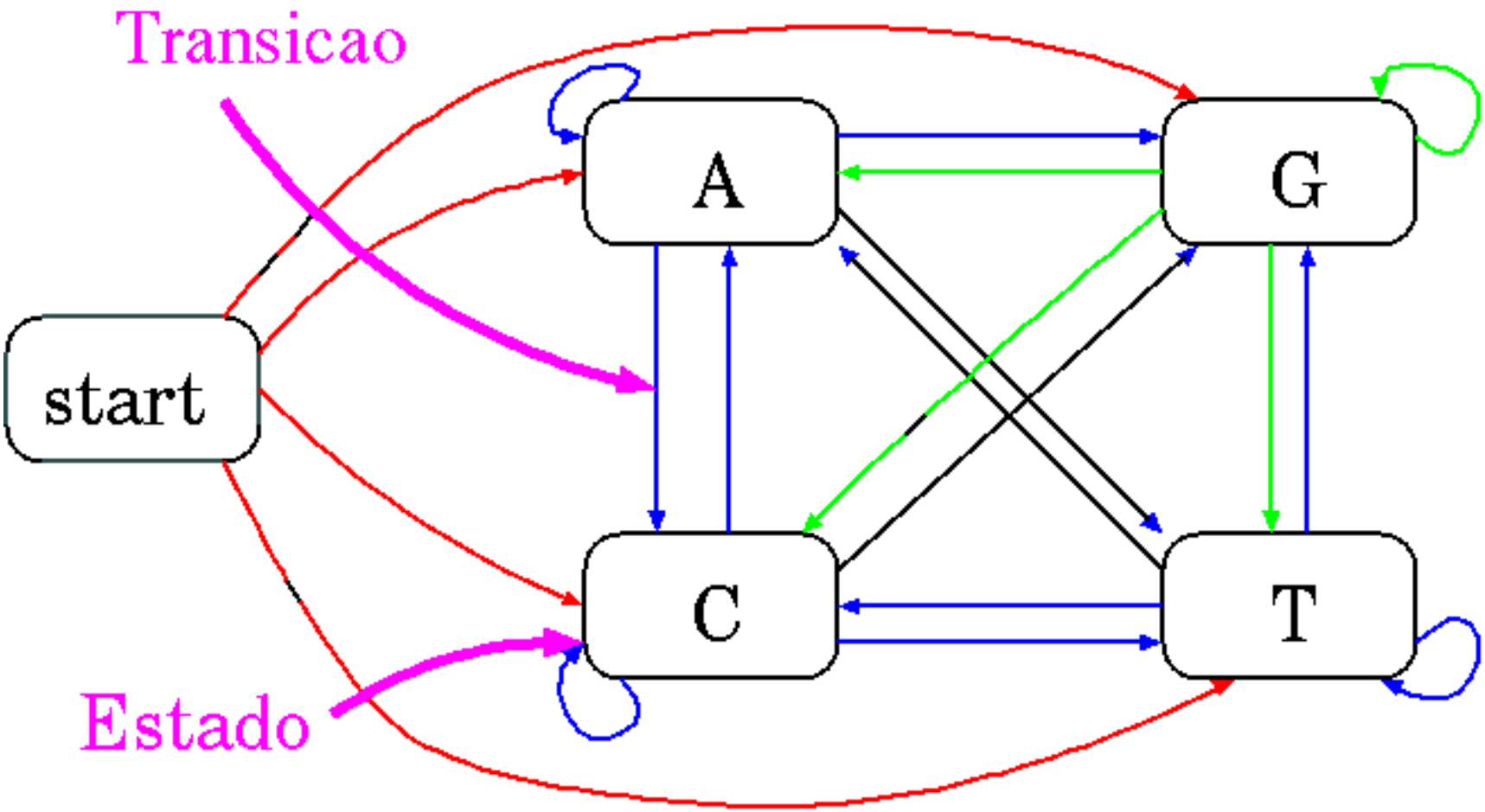
- Dada uma sequência x de comprimento L, queremos saber qual a probabilidade dado o nosso modelo
- Para qualquer modelo probabilístico de sequências, podemos escrever esta probabilidade como:

$$\begin{aligned} P(x) = \quad & P(x_L, x_{L-1}, \ldots, x_1) = \\ & P(x_L | x_{L_1}, \ldots, x_1) P(x_{L-1} | x_{L-2}, \ldots, x_1) \ldots P(x_1) \end{aligned}$$

- propriedade chave das cadeias de Markov da primeira ordem: a probabilidade de cada x_i depende apenas de x_{i-1}:

$$\begin{aligned} P(x) = \quad & P(x_L | x_{L-1}) P(x_{L-1} | x_{L-2}) \ldots P(x_2 | x_1) P(x_1) \\ & P(x_1) \prod_{i=2}^{L} P(x_i | x_{i-1}) \end{aligned}$$

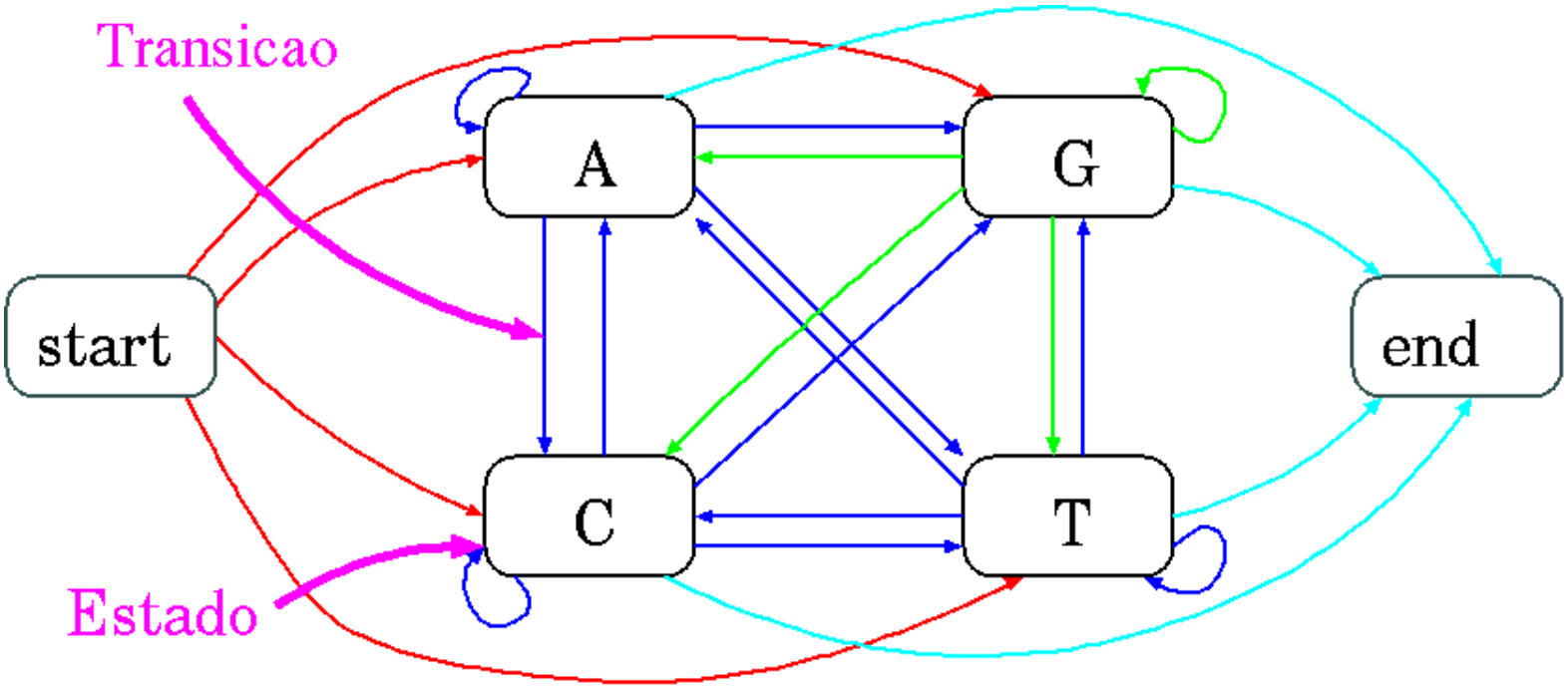
Probabilidade de uma Sequência



$$P(cggt) = P(c)P(g|c)P(g|g)P(t|g)$$

Modelos de Cadeia de Markov

- Podem ter um estado final: permitem ao modelo representar:
  - uma distribuição sobre sequências de tamanhos diferentes
  - preferências para terminar sequências com certos tamanhos



Notação de Cadeias de Markov

- Os parâmetros de transição podem ser denotados como $a_{x_{i-1}x_i}$ onde:

$$a_{x_{i-1}x_i} = P(x_i|x_{i=1})$$

- podemos denotar a probabilidade de uma sequência x como sendo

$$a_{Bx_1} \prod_{i=2}^{L} a_{x_{i-1}x_i} = P(x_1) \prod_{i=2}^{L} P(x_i|x_{i-1})$$

onde a_{Bx_1} representa a transição do estado `begin`.

Exemplo de Aplicação

- Ilhas CpG:
  - dinucleotídos *CG* são mais raros em genomas eucarióticos do que esperado dada as probabilidade marginais de *C* e de *G*
  - mas as regiões acima de um gene são mais ricas em dinucleotídos *CG* do que em qualquer outro lado: *ilhas CpG*
  - informação útil para encontrar genes
- podemos prever ilhas CpG com cadeias de Markov:
  - uma para representar ilhas CpG
  - outra para representar o resto do genoma.

Estimando os Parâmetros do Modelo

- Obtido um conjunto de dados, (eg, um conjunto de sequências de ilhas CpG), como determinar os parâmetros do nosso modelo?
- Uma solução: `estimação com parecença máxima`:
  - Dado um conjunto de dados D
  - obter os parâmetros θ para maximizar

$$P(D|\theta)$$

  - ie, fazer com que os dados pareçam o mais possível de acordo com o modelo.

Estimação de Máxima Semelhança (Likelihood)

- suponhamos que queremos estimar os parâmetros $P(a)$, $P(c)$, $P(g)$, e $P(t)$
- e recebemos as sequências:
  - `accgcgctta`
  - `gcttagtgac`
  - `tagccgttac`
- A estimação de máxima semelhança será:

$$P(a) = \frac{6}{30} = 0.2 \quad P(g) = \frac{7}{30} = 0.233$$

$$P(c) = \frac{9}{30} = 0.3 \quad P(t) = \frac{8}{30} = 0.267$$

O Método Bayesiano

- Em vez de estimarmos os parâmetros estreitamentos dos dados, podemos começar com alguma crença prévia
- por exemplo, podemos usar estimativas de Laplace:

$$P(a) = \frac{n_a + 1}{\sum_i (n_i + 1)}$$

- $n_a + 1$ é uma *pseudo-contagem*
- e n_i representa o número de ocorrências do caracter i
- usando estimativas de Laplace com as sequências:
  - `gccgcgcttg`
  - `gcttggtggc`
  - `tggccgttgc`

O Método Bayesiano

- uma forma mais geral: *m-estimates*

$$P(a) = \frac{n_a + p_a m}{\sum_i (n_i) + m}$$

- p_a é a probabilidade prévia para a
- m é o número de instâncias virtuais
- com $m = 8$ e prévias uniformes:

$$P(s) = \frac{9 + 0.25 \times 8}{30 + 8} == \frac{11}{38}$$

Estimação de Parâmetros de Primeira Ordem

- Para estimar um parâmetro de primeira ordem, como $P(c|g)$, contamos o número de vezes que g segue a história de c nas nossas sequências
- Usando estimadores de Laplace:

$$P(a|g) = \frac{0+1}{12+4} \quad P(a|c) = \frac{0+1}{7+4}$$

$$P(a|g) = \frac{7+1}{12+4} \quad \ldots$$

$$P(a|g) = \frac{3+1}{12+4}$$

$$P(a|g) = \frac{2+1}{12+4}$$

Cadeias de Markov para Discriminação

- queremos distinguir ilhas CpG de outras regiões
- dadas sequências de ilhas CpG, e sequências de outras regiões, podemos construir:
  - um modelo para representar ilhas CpG
  - um *modelo nulo* para representar as outras regiões
- Podemos avaliar uma sequência de teste por:

$$score(x) = log \frac{P(x|\text{modelo CpG})}{P(x|\text{modelo nulo})}$$

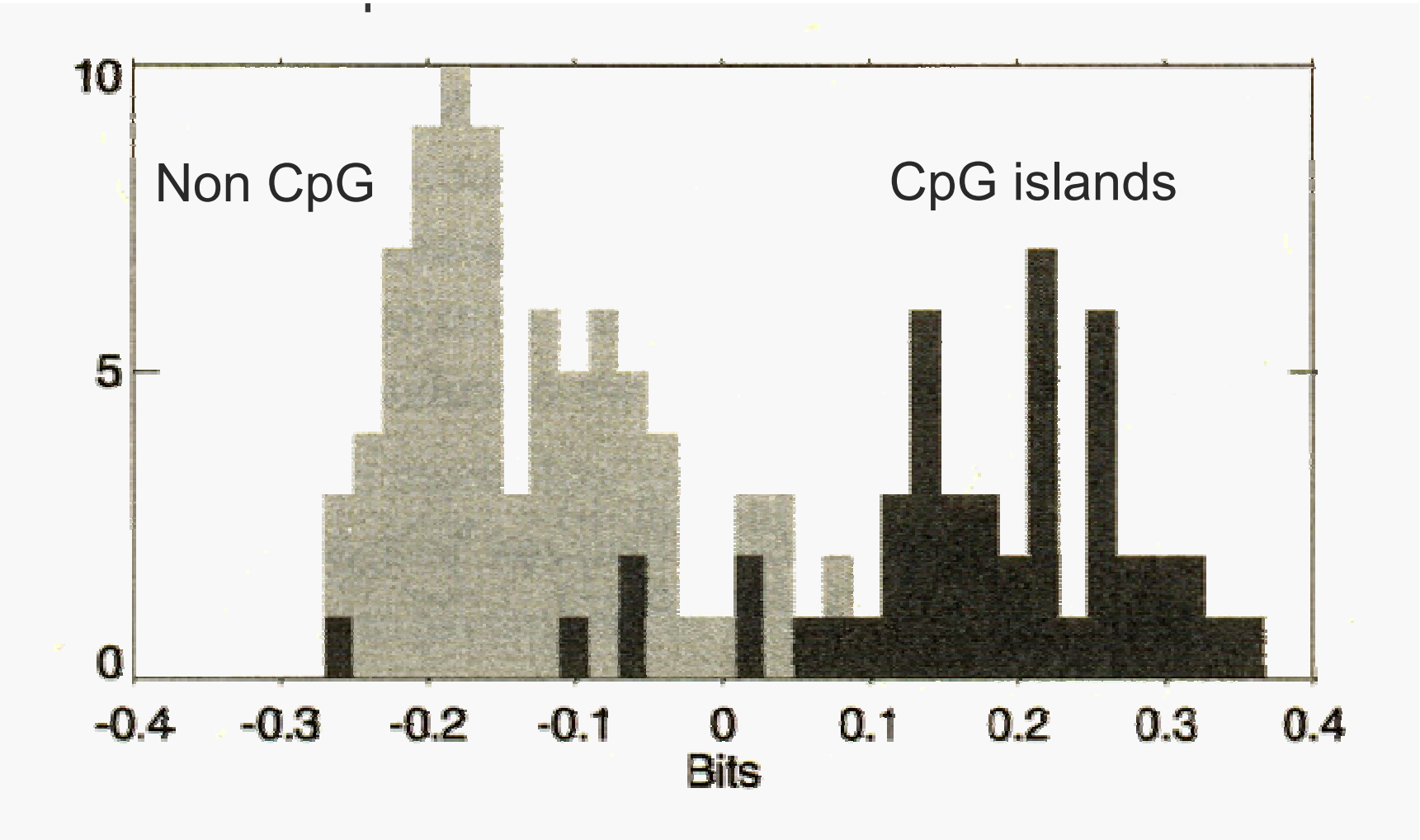
Cadeias de Markov para Discriminação

- Parâmetros estimados para CpG e para modelos nulos:
  - genoma humano contém 48 ilhas CpGs
  - 60000 nucleotídos

$+$	A	C	G	T
A	.18	.27	.43	.12
C	.17	.37	.27	.19
G	.16	.34	.38	.12
T	.08	.36	.38	.18

$-$	A	C	G	T
A	.30	.21	.28	.21
C	.32	.30	.08	.30
G	.25	.24	.30	.21
T	.18	.24	.29	.29

Cadeias de Markov para Discriminação



Normalizado por tamanho de sequências.

Cadeias de Markov para Discriminação

- Porque usar

$$score(x) = log \frac{P(x|CpG)}{P(x|null)}$$

- A regra de Bayes diz-nos que:

$$P(CpG|x) = \frac{P(x|CpG)P(CpG)}{P(x)}$$

$$P(null|x) = \frac{P(x|null)P(null)}{P(x)}$$

- se não estivermos a considerar probabilidades prévias das duas classes ($P(CpG)$) e ($P(null)$) então basta comparar $P(x|CpG)$ e $P(x|null)$.

Cadeias de Markov de Mais Alta Ordem

- A propriedade de Markov especifica que a probabilidade de um estado depende apenas da probabilidade do estado prévio
- mas podemos construir mais memória nos nossos estados usando um modelo de mais alta ordem
- num modelo de Markov de ordem n:

$$P(x_i|x_{i-1}, x_{i-2}, \ldots, 1) = P(P(x_i|x_{i-1}, \ldots, x_{i-n})$$

Selecção da Ordem de um Modelo

- Modelos de ordem mais alta lembram-se de mais *história*
- mais história pode ser útil para previsões
- Por exemplo
  - Preveja a próxima palavra neste fragmento de frase:
    - . . . quer ____
  - Agora com mais história
    - . . . quem desdenha quer ____

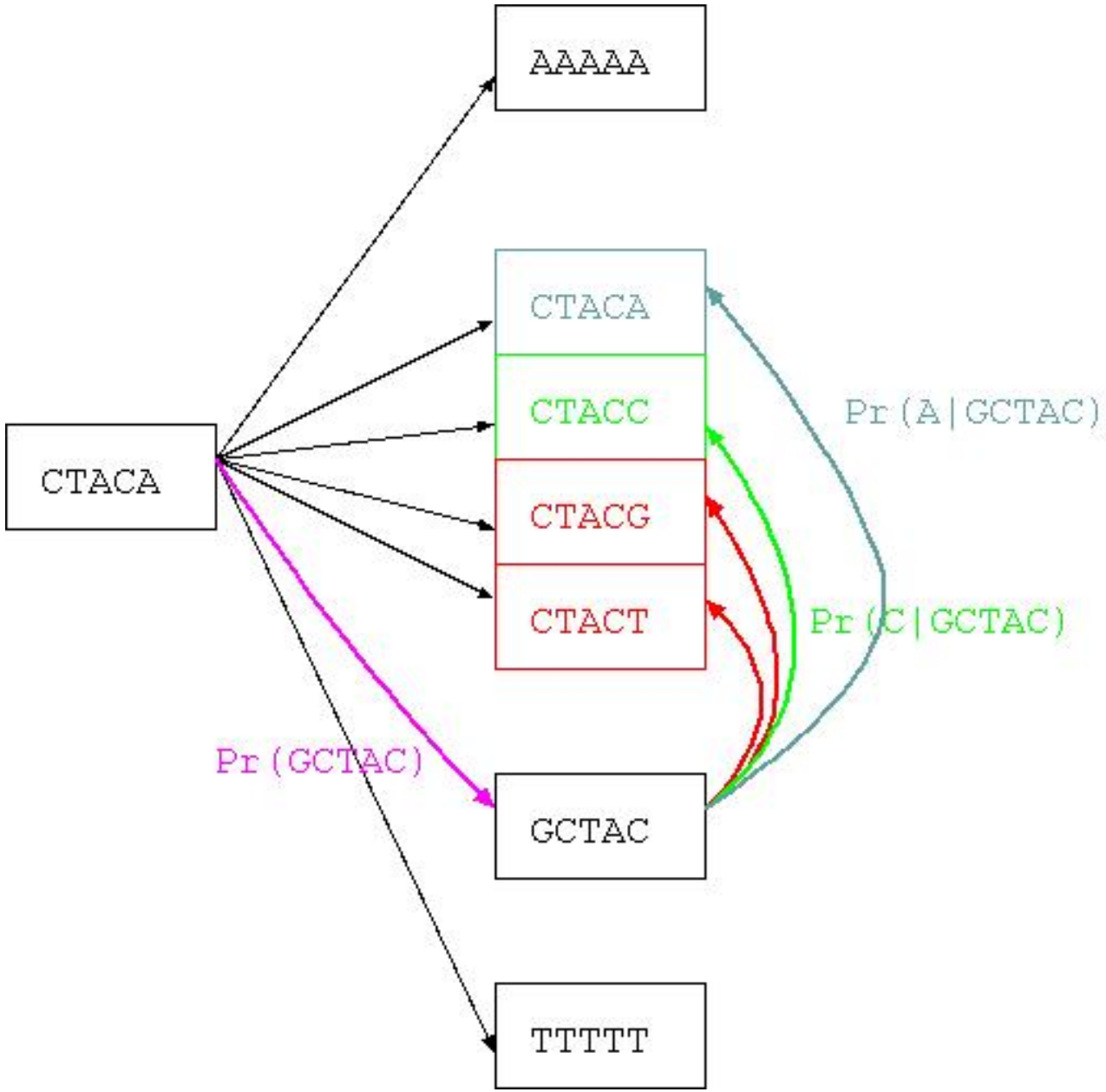
Estimando a ordem de um Modelo de Markov

- O problema é que o número de parâmetros que precisamos de estimar cresce exponencialmente com a ordem
  - Para modelar DNA precisamos de $O(4^{n+1})$ para um modelo de ordem n
- quanto maior a ordem, quanto menos confiáveis serão as nossas estimas de parâmetros:
  - para estimar os parâmetros de uma cadeia de segunda ordem de Markov do genoma completo de E. coli, precisamos de ver cada palavra > 72000 em média
  - para estimar os parâmetros de uma cadeia de oitava ordem, precisamos de ver cada palavra 5 vezes em média

Cadeias de Markov de Mais Alta Ordem

- Uma cadeia de Markov de ordem n sobre um alfabeto A é equivalente a uma cadeia de primeira ordem sobre o alfabeto dos tuplos-n A^n
- Exemplo, uma cadeia de segunda ordem pode ser tratada como uma cadeia de primeira ordem sobre os pares:
  - AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT

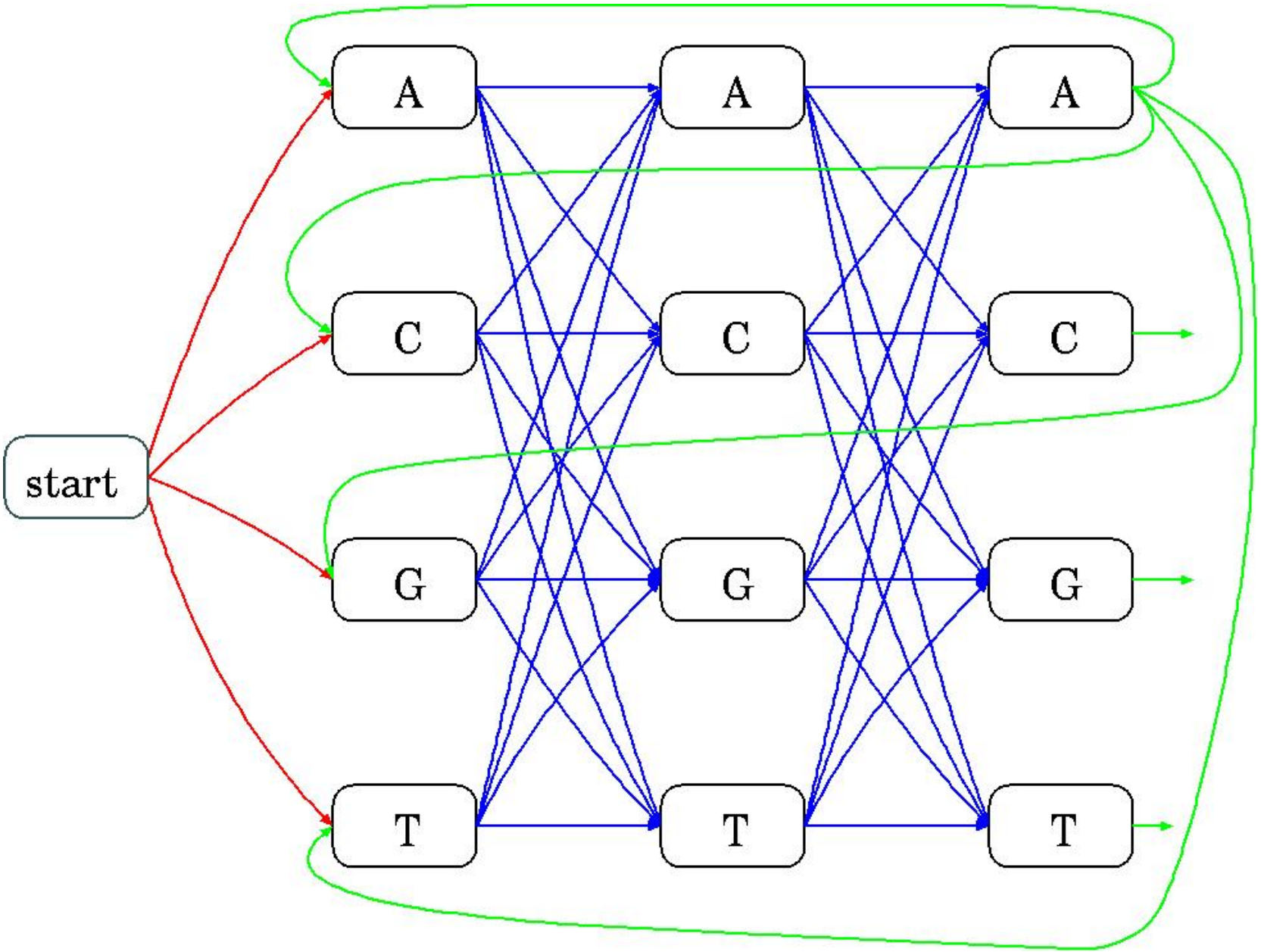
Uma Cadeia de Markov de Quinta Ordem



Cadeias Não Homogéneas de Markov

- Nos modelos de Markov que temos considerado até agora, as probabilidades não dependem de onde estamos numa dada sequência
- Num modelo não homogéneo de Markov, podemos ter distribuições diferentes em posições diferentes da sequência
- Considere a modelação de codões em regiões de codificação de proteínas

Uma Cadeia Não Homogénea de Markov



Aplicação de Cadeia Não Homogénea

- construir modelos de Markov para regiões que codificam ou não
- aplicar modelos para ORFs (quadros de leitura abertos) ou para janelas de tamanho fixo da sequências
- Exemplo: GeneMark de Borodovsky et al:
  - sistema popular para identificar genes em genomas de bactérias
  - usa cadeias de quinta ordem não homogéneas de Markov
  - `http://opal.biology.gatech.edu/GeneMark/`

ORF: Quadro de Leitura Aberta

- Um quadro de leitura aberta é uma sequência que pode potencialmente codificar uma proteína:
  - começa com um potencial codon de inicio
  - acaba com um potencial codon de fim
  - satisfaz um tamanho mínimo
  - em procariotes, o codon de inicio e de estão na mesma janela

ATG CCG AAA TCG CAT GCA TTT GTG ... TAG

Técnicas para Encontrar Genes

Procurar por:

- *semelhança de sequência*: encontrar genes procurando sequências semelhantes a sequências que se sabem ser relacionadas a genes
- *sinal*: encontrar genes identificando os sinais de sequenciamento envolvidos em expressão de genes
- *conteúdo*: usar propriedades estatísticas que distinguem DNA que codifica para proteínas do resto do DNA
- *combinado*: métodos mais avançados combinam estas estratégias.

Procura Por Conteúdo

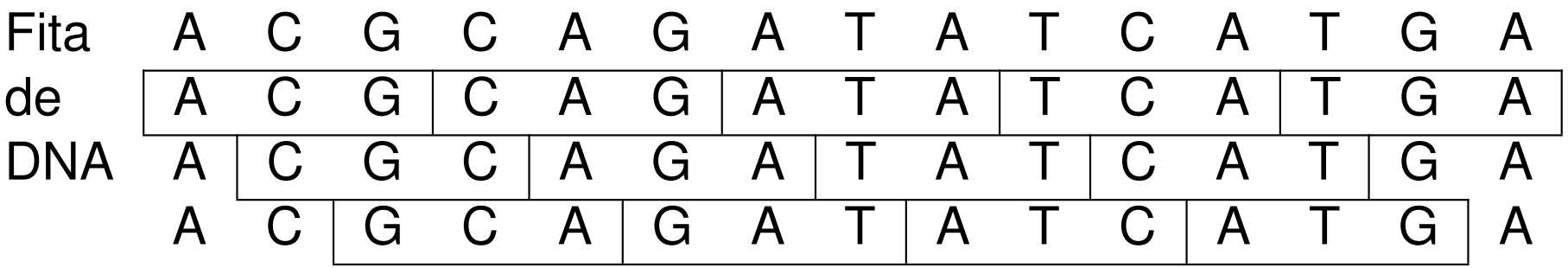
- codificar uma proteína afecta as propriedades estatísticas de uma sequência de DNA:
  - alguns amino-ácidos são ácidos mais frequentemente que outros (Leu é mais popular do que Trp)
  - número diferente de codões para amino-ácidos diferentes (Leu tem 6, Trp tem 1)
  - para um certo amino-ácido, habitualmente um codon é usado mais frequentemente que outros:
    - chamado de *preferência de codon*
    - essas preferências variam com a espécie

Exemplo: E. Coli

AA	codon	freq. por 1000
Gly	GGG	1.89
Gly	GGA	0.44
Gly	GGU	52.99
Gly	GGC	34.55
Glu	GAG	15.68
Glu	GAA	57.20
Asp	GAU	21.63
Asp	GAC	43.26

Quadros de Leitura

- 6 quadros para uma fita:

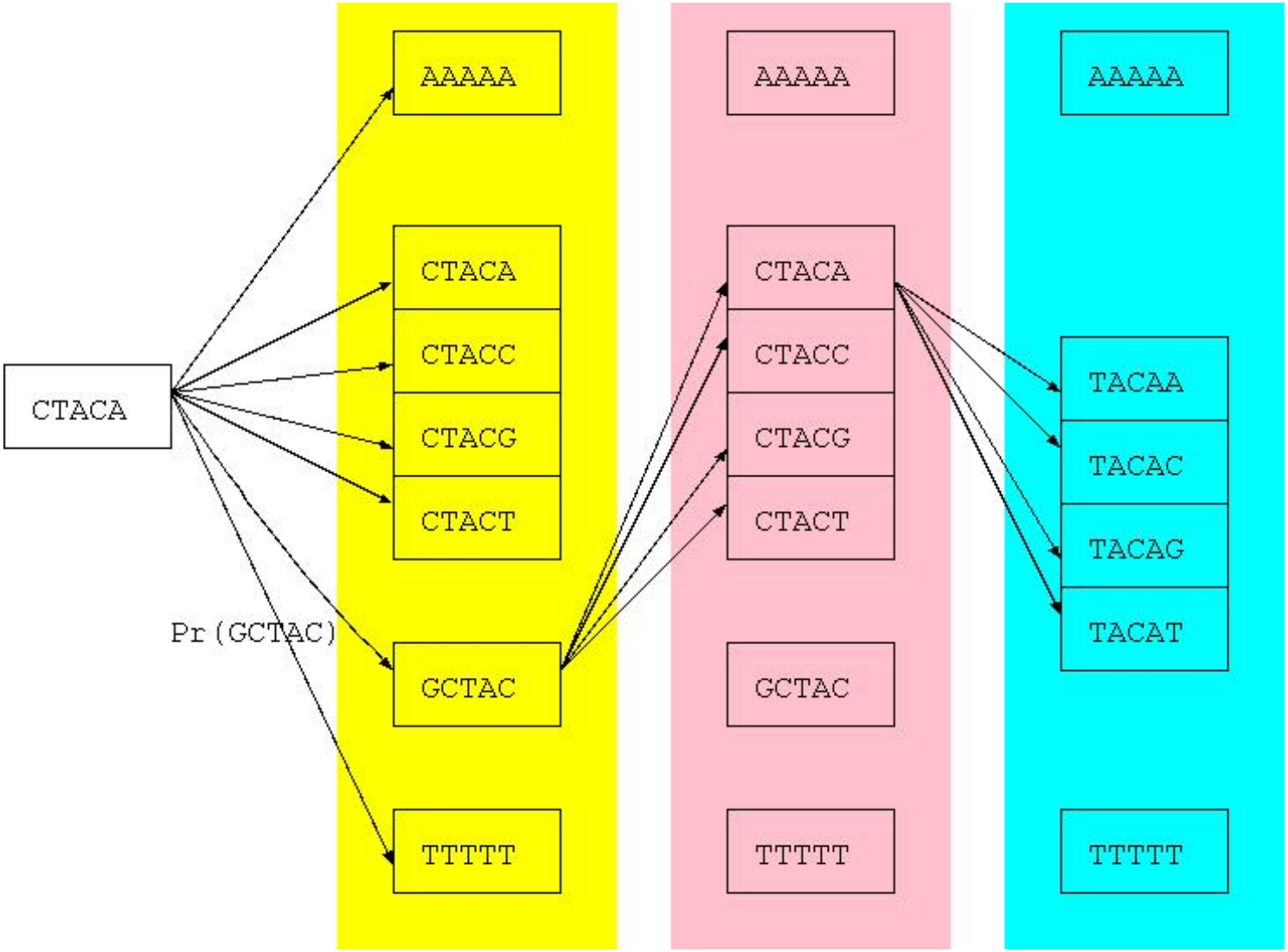


Modelos de Markov e Quadros de Leitura

- Imagine modelar uma sequência de codificação
- para cada palavra que avaliamos, queremos considerar a sua posição em relação ao quadro de leitura que estamos assumindo

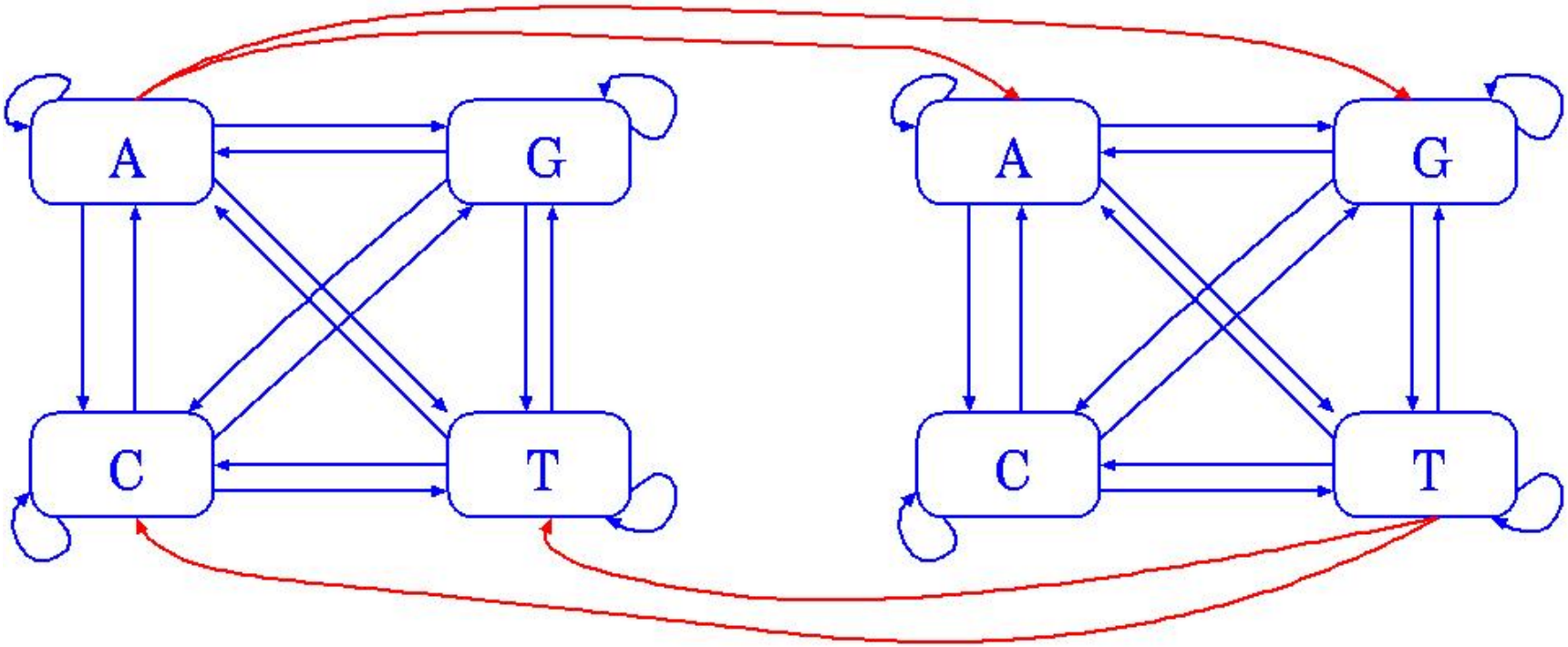
Fita	A	C	G	C	A	G	A	T	A	T	C	A	T	G	A
de	A	C	G	C	A	G	A	T	A	T	C	A	T	G	A
DNA	A	C	G	C	A	G	A	T	A	T	C	A	T	G	A
	A	C	G	C	A	G	A	T	A	T	C	A	T	G	A

Cadeia de Quinta Ordem Não Homogénea



HMMs

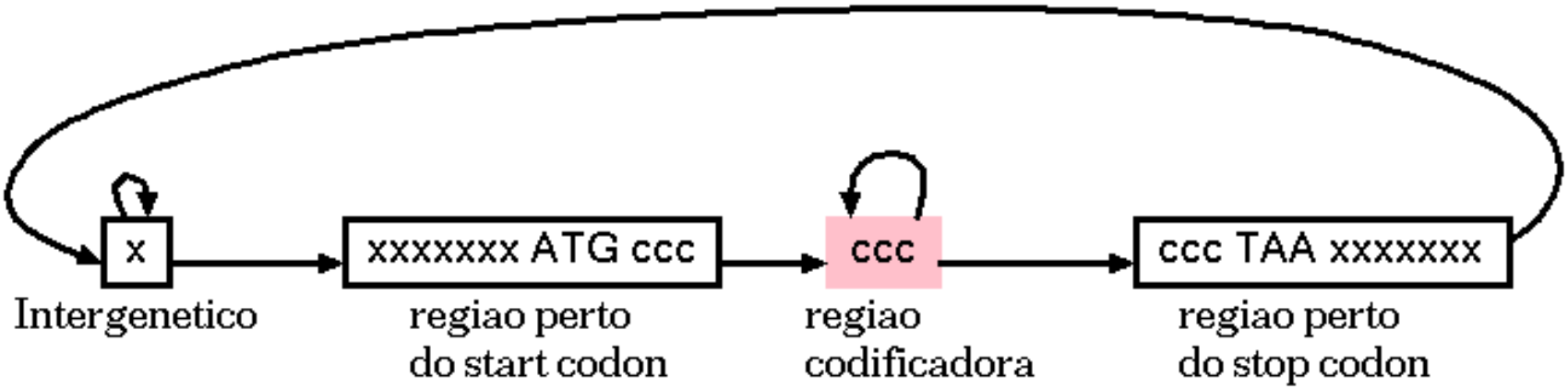
- Dado um T na nossa sequência de leitura, quem o emitiu?

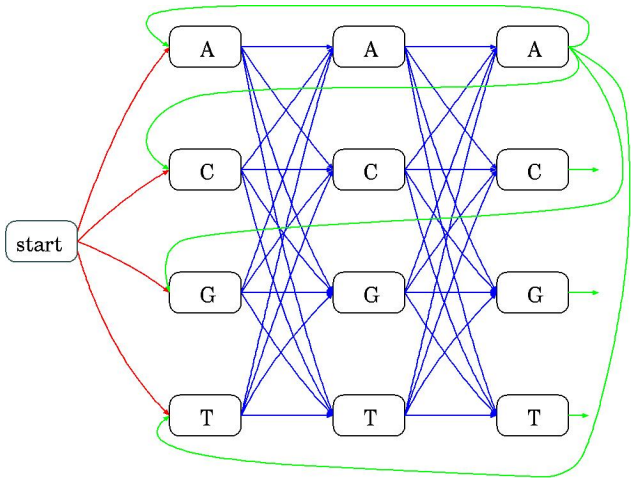


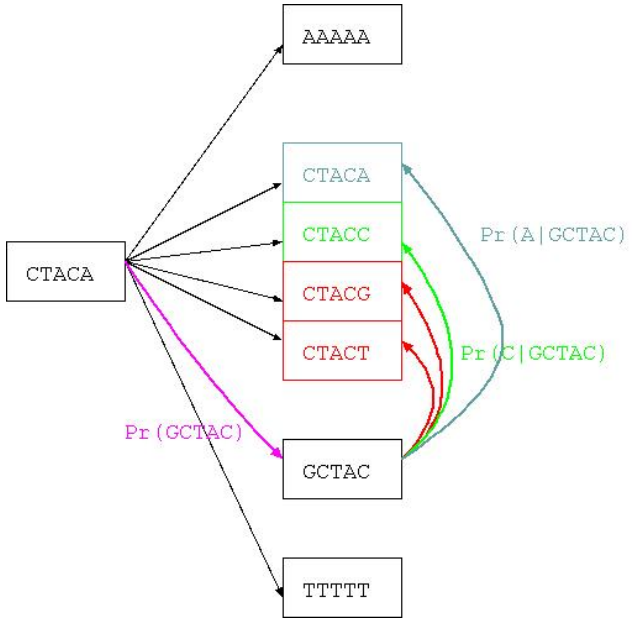
Estado Escondido

- Vamos distinguir entre as partes observadas de um problema e as parte escondidas.
- Nos modelos que consideramos antes, é claro que estado é responsável por que parte da sequência
- no modelo anterior, há múltiplos estados que podem causar cada parte de uma sequência observada:
  - é a parte escondida do problema

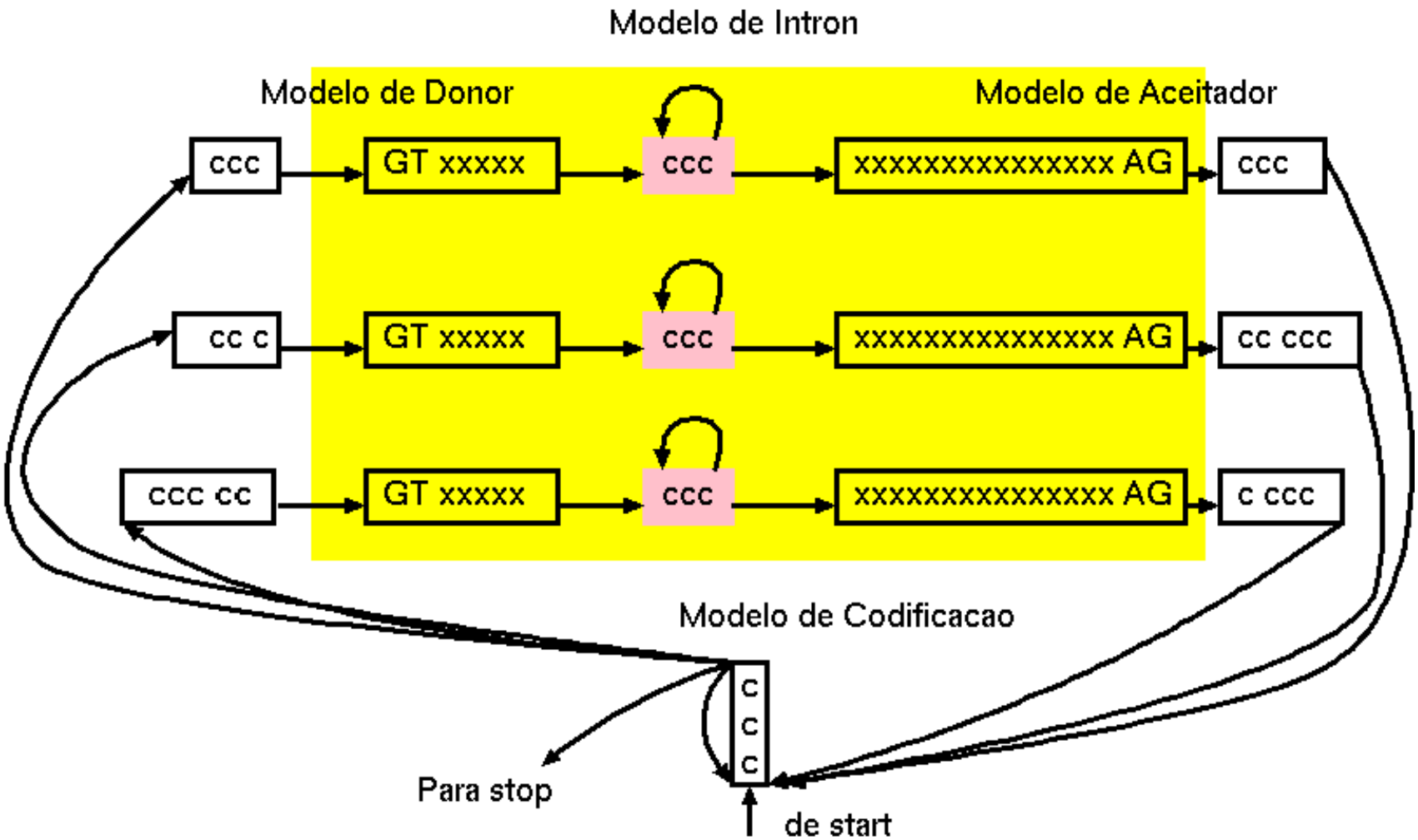
HMM Simples para Encontrar Genes





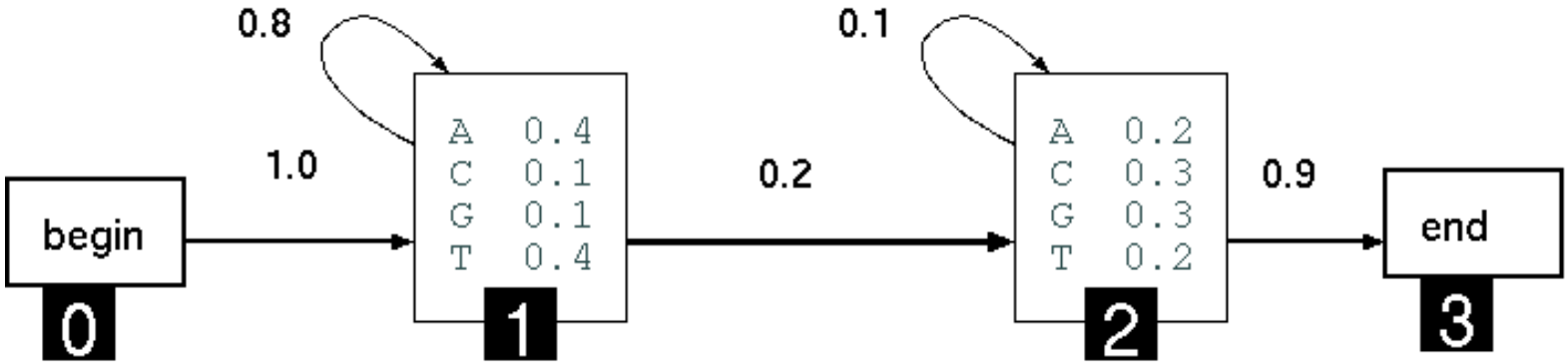


HMM Simples para Eucariotes



Exemplo do Algoritmo de Baum-Welch

- dado
  - um HMM com os parâmetros inicializados como em baixo
  - sequências de treino `TAG`, `ACG`



- vamos seguir uma iteração de Baum-Welch

Exemplo do Algoritmo de Baum-Welch

- determinando os valores de forward para TAG:

$$
\begin{aligned}
f_0(0) = \quad & 1 \\
f_1(1) = \quad & e_1(\mathtt{T}) \times a_{01} \times f_0(0) = 0.4 \times 1 = 0.4 \\
f_1(2) = \quad & e_1(\mathtt{A}) \times a_{11} \times f_1(1) = 0.4 \times 0.8 \times 0.4 = 0.128 \\
f_2(2) = \quad & e_2(\mathtt{A}) \times a_{12} \times f_1(1) = 0.2 \times 0.2 \times 0.4 = 0.016 \\
f_2(3) = \quad & e_2(\mathtt{G}) \times (a_{12} \times f_1(2) + a_{22} \times f_2(2)) = \\
& 0.3 \times (0.0256 + 0.0016) = 0.00816 \\
f_3(3) = \quad & a_{23} \times f_3(2) = 0.9 \times 0.01056 = 0.007344
\end{aligned}
$$

- computamos apenas valores que representam probabilidade não-zero
- da mesma forma podemos computar os valores de forward de ACG

Exemplo do Algoritmo de Baum-Welch

- determinando os valores de backward para TAG:

$$
\begin{aligned}
b_3(3) = & \quad 1 \\
b_2(3) = & \quad a_{23} \times b_3(3) = 0.9 \times 1 = 0.9 \\
b_2(2) = & \quad a_{22} \times e_2(\mathtt{G}) \times b_2(3) = 0.1 \times 0.3 \times 0.9 = 0.027 \\
b_1(2) = & \quad a_{12} \times e_2(\mathtt{G}) \times b_2(3) = 0.2 \times 0.3 \times 0.9 = 0.054 \\
b_1(1) = & \quad a_{11} \times e_1(\mathtt{A}) \times b_1(2) + a_{12} \times e_2(\mathtt{A}) \times b_2(2) = \\
& \quad 0.8 \times 0.4 \times 0.054 + 0.2 \times 0.2 \times 0.027 = 0.01836 \\
b_0(0) = & \quad a_{01} \times e_1(\mathtt{T}) \times b_1(1) = 1.0 \times 0.4 \times 0.01836 = 0.007344
\end{aligned}
$$

- computamos apenas valores que representam probabilidade não-zero
- da mesma forma podemos computar os valores de backward de ACG

Exemplo do Algoritmo de Baum-Welch

- determinando o número esperado de contagens de emissões para o estado 1:

	contribuição de TAG		contribuição de ACG		pseudo contagem
$n_{1,A} =$	$\frac{f_1(2)b_1(2)}{f_3(3)}$	$+$	$\frac{f_1(1)b_1(1)}{f_3(3)}$	$+$	1
$n_{1,C} =$			$\frac{f_1(2)b_1(2)}{f_3(3)}$	$+$	1
$n_{1,G} =$					1
$n_{1,T} =$	$\frac{f_1(1)b_1(1)}{f_3(3)}$			$+$	1

- note que os valores de forward/backward diferem entre as duas sequências.

Exemplo do Algoritmo de Baum-Welch

- determinando o número esperado de transições para o estado 1 (não usamos pseudo-contagens):

$$n_{1\to 1} = \underbrace{\frac{f_1(1)a_{11}e_1(\mathtt{A})b_1(2)}{f_3(3)}}_{\text{contribuição de } \mathtt{TAG}} + \underbrace{\frac{f_1(1)a_{11}e_1(\mathtt{C})b_1(2)}{f_3(3)}}_{\text{contribuição de } \mathtt{ACG}}$$

$$n_{1\to 2} = \frac{f_1(1)a_{12}e_2(\mathtt{A})b_2(2)+f_1(2)a_{12}e_2(\mathtt{G})b_2(3)}{f_3(3)} + \frac{f_1(1)a_{12}e_2(\mathtt{C})b_2(2)+f_1(2)a_{12}}{f_3(3)}$$

- da mesma forma, podemos determinar os números esperados de emissão e de transição para o estado 2.

Exemplo do Algoritmo de Baum-Welch

- Determinando as probabilidades para o estado 1:

$$e_1(\mathtt{A}) = \frac{n_{1,A}}{n_{1,A}+n_{1,C}+n_{1,G}+n_{1,T}}$$

$$e_1(\mathtt{C}) = \frac{n_{1,C}}{n_{1,A}+n_{1,C}+n_{1,G}+n_{1,T}}$$

$$\ldots$$

$$a_{11} = \frac{n_{1\to 1}}{n_{1\to 1}+n_{1\to 2}}$$

$$a_{12} = \frac{n_{1\to 2}}{n_{1\to 1}+n_{1\to 2}}$$

Convergência em Baum-Welch

- Alguns critérios de convergência:
  - verosimilhança da sequência de treino não muda muito
  - número máximo de iterações
- habitualmente converge num número pequeno de iterações
- converge para máximo *local* (na verosimilhança dos dados dado o modelo)

$$logP(\text{sequências}|\theta) = \sum_{x^j} logP(x^j|\theta)$$

Complexidade Computacional dos Algoritmos HMM

- Dado um HMM com S estados e uma sequência de comprimento L, a complexidade dos algoritmos Forward, Backward e de Viterbi é de:

$$O(S^2 L)$$

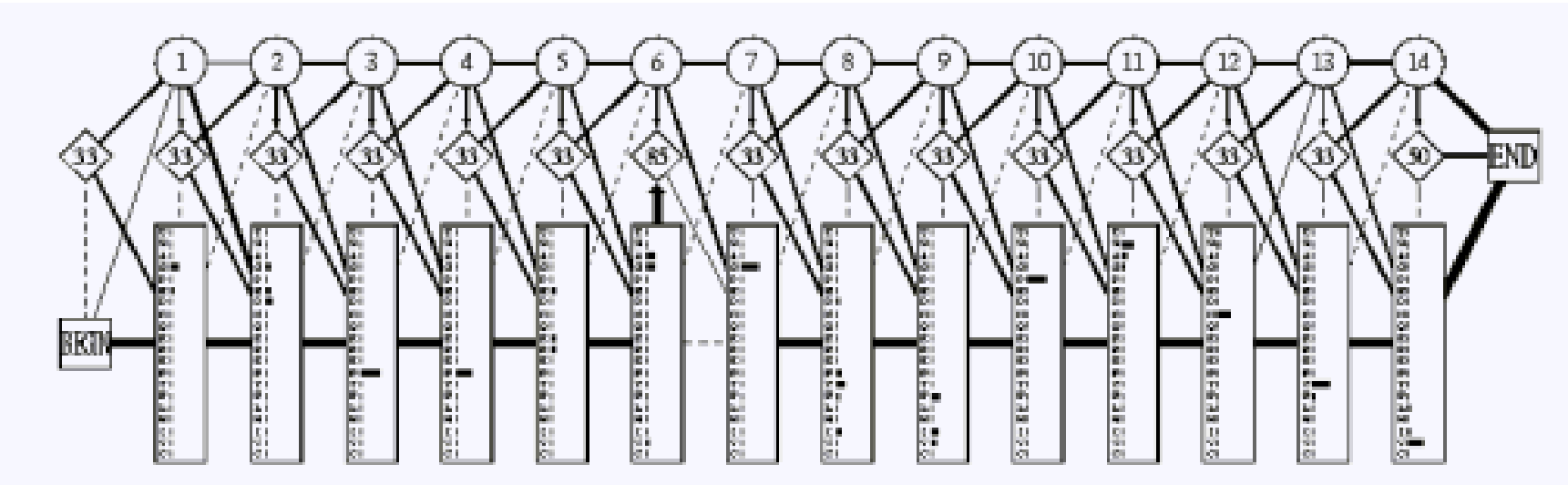
  - Isto assume que os estados são densamente interligados
- Dadas M sequências de comprimento L, a complexidade de Baum-Welch em *cada iteração* é:

$$O(MS^2 L)$$

Uma Aplicação de Aprendizagem com Baum-Welch em HMMs

- Modelando famílias de proteínas usando *profile HMMs*
- *profile HMMs* podem ser usados para:
  - determinar uma sequência múltipla de alinhamento para um conjunto de proteínas
  - detectar novo membros de uma família de proteínas
- Aplicação mais importante de HMMs em BioComputação:
  - HMMer
  - Tutorial em ISMB99
  - Tutorial sobre evolução de proteínas (ISMB2000)

Um Profile HMM treinado para o Domínio SH3



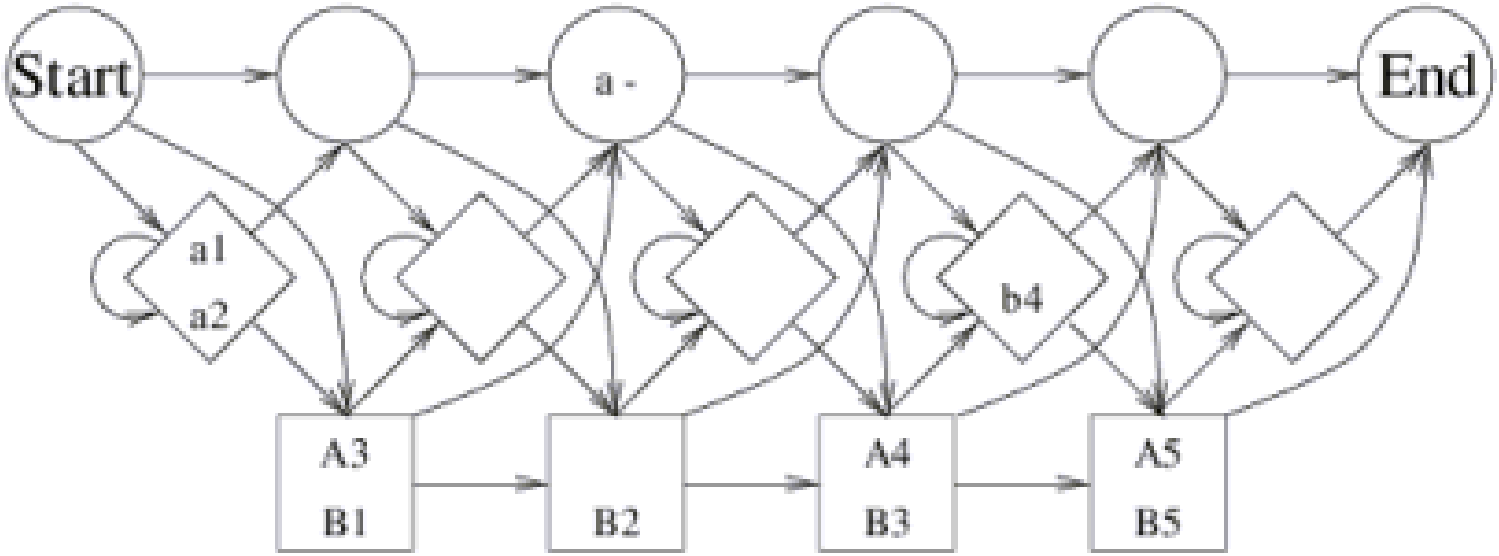
A Estrutura de um Profile HMM

- *Estados de Emparelhamento*: representam posições essencialmente conservadas na família de sequências
- *Estados de Inserção*: representam subsequências que foram inseridas em alguns membros da família
- *Estados de Remoção*: representam subsequências que foram removidas em alguns membros da família

Alinhamento Múltiplo no Domínio SH3

```
G G W W R G d y . g g k k q L W F P S N
I G W L N G y n e t t g e r G D F P G T
P N W W E G q l . . n n r r G I F P S N
D E W W Q A r r . . d e q i G I V P S K
G E W W K A q r . . t g q e G F I P F N
G D W W L A r s . . s g q t G Y I P S N
G D W W D A e l . . k g r r G K V P D N
- D W W E A r s l s s g h r G Y V P S N
G D W W Y A r s l i t n s e G Y I P S T
G E W W K A r s l a t r k e G Y I P S N
G D W W L A r s l v t g r e G Y V P S N
G E W W K A q t . k n g q . G W V P S N
S D W W R V v n l t t r q e G L I P L N
L P W W R A r d . k n g q e G Y I P S N
R D W W E F r s k t v y t p G Y Y E S G
E H W W K V k d . q l g n v G Y I P S N
I H W W R V q d . r n g h e G Y V O S S
K D W W K V e v . . n d r q G F V P A A
V G W M P G l n e r t r q r G D F P G T
P D W W E G e l . . n g q r G V F P A S
```

Como é que os Profile HMMs Representam Famílias de Proteínas

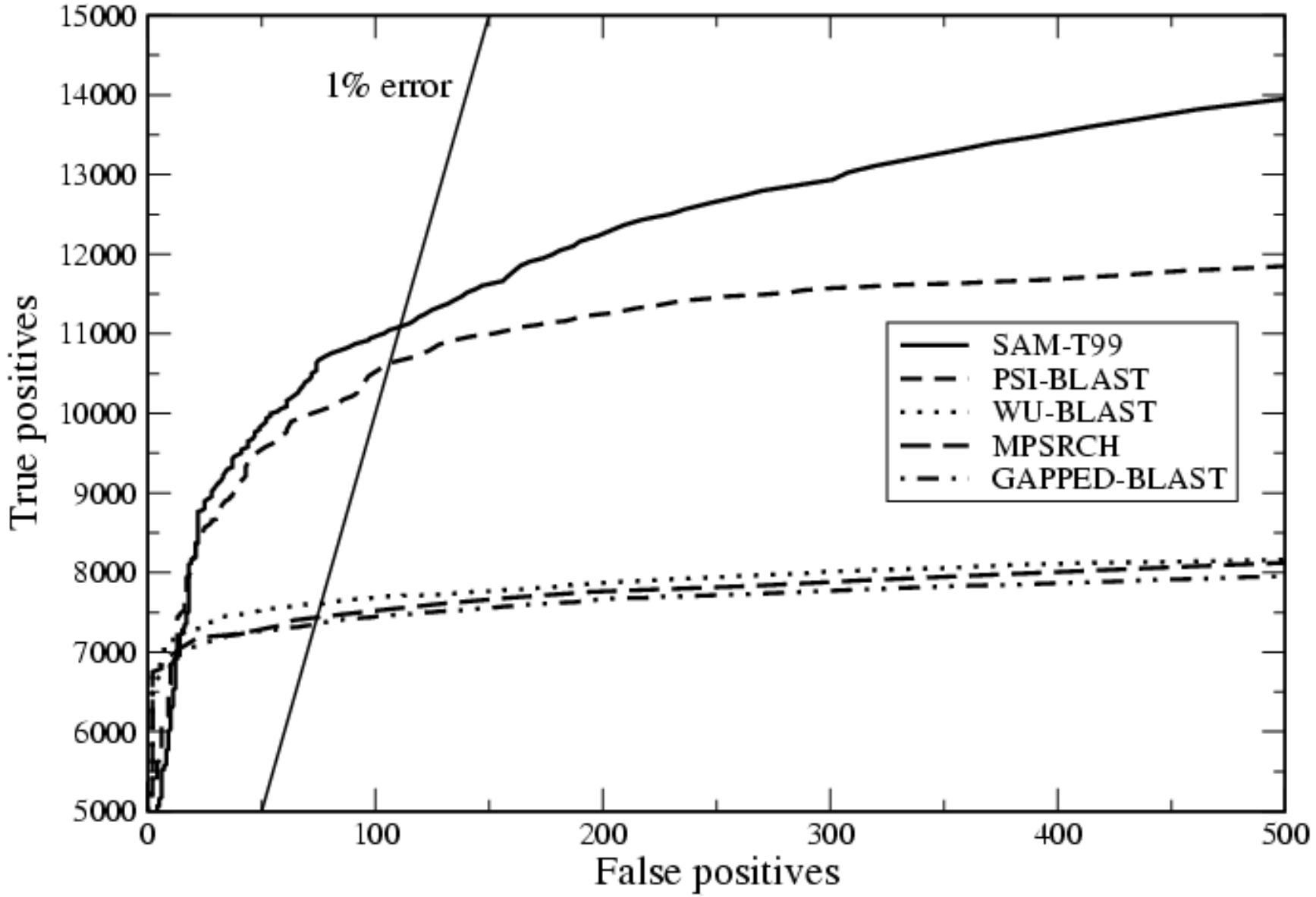



```
a1  a2  A3      -      A4  .  A5
 .   .  B1      B2     B3 b4  B5
```

Classificando Sequências: Três Métodos

- escolha um limite sobre $P(x)$ que permita boa discriminação entre casos positivos e casos negativos:
  - depende do comprimento de x
- construa um *modelo nulo*; rode uma sequência x pelos dois para ver quem obtém um $P(x)$ maior
- construa um conjunto de modelos para famílias disjuntos; rode a sequência de interrogação x por todos os modelos para ver quem obtém o maior $P(x)$

Acurácia de Profile HMM

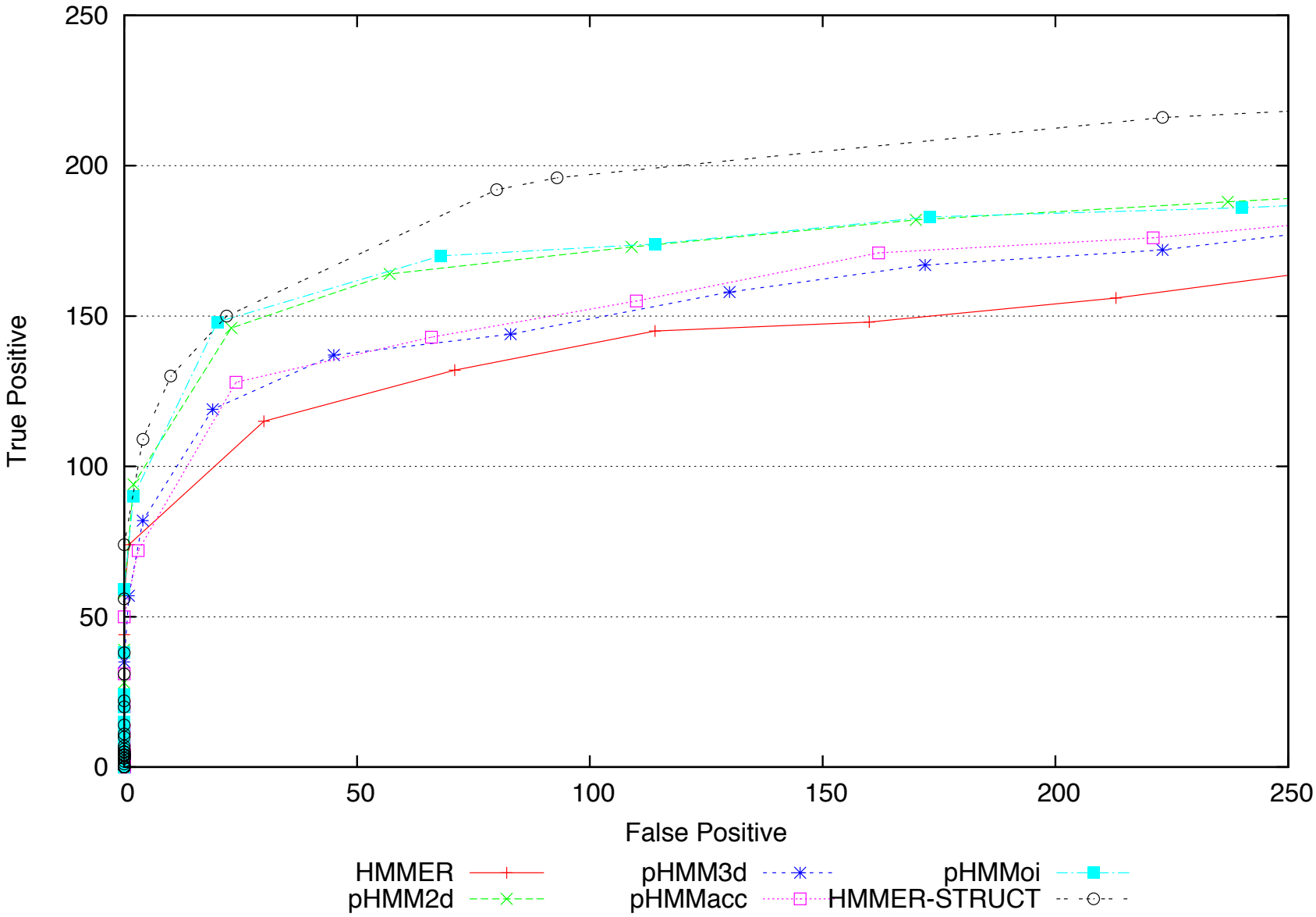


- classificando 2447 proteínas em 33 famílias

Selecção de Modelo para Profile HMMs

- assumimos que recebemos um modelo sem comprimento especificado; como determinar o comprimento?
- heurísticas:
  - escolha um comprimento inicial; aprenda parâmetros
  - se mais do que $x - del\%$ dos caminhos de Viterbi, vão por posições de remoção na posição k, remova essa posição do modelo
  - se mais do que que $x - ins\%$ vão por inserções na posição k, adicione novas posição ao modelo.
  - itere

Acurácia de Profile HMM na Twilight Zone



Comentários sobre Modelos de Markov

- Há muitas aplicações bem-sucedidas em biologia computacional
  - reconhecimento de genes e tarefas associadas
  - modelagem de famílias de proteínas
  - modelagem de motivos
  - e mais
- Existem muitas variantes dos modelos que consideramos aqui:
  - modelos de motivos de tamanho fixo
  - modelos de semi-markov
  - gramáticas estocásticas de contexto livre
  - amostragem de Gibbs para aprender parâmetros

Redes Bayesianas: Mais Tutoriais

- Tutorial Web de Kevin Murphy, autor de BNT
- Tutoriais de Nir Friedman
- Tutoriais de Andrew Moore sobre Statistical Data Mining
- PRMs e POMDPs
- Estatística Bayesiana

Métodos Probabílisticos para Árvores

- *Label*, ou *estado* é um vector de m caracteres associado com cada espécie, ou nó.
- *Reconstrução* é uma etiquetagem dos nós internos
- t_{uv} mede a distância entre as espécies nos nós u e v.

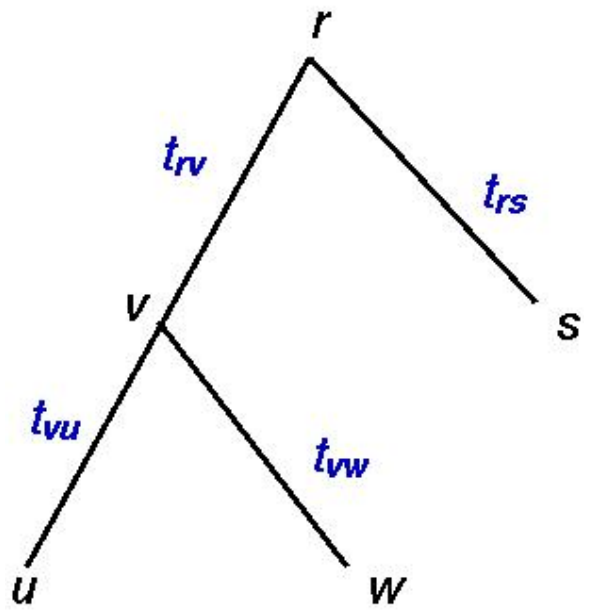
Assumimos:

- Probabilidade de um caracter x é sempre fixa: $P(x)$
- Temos uma função de distância que computa $Px \rightarrow y(t_{vu})$, a probabilidade de passarmos de x para y em t_{vu}.
- Ramificação é um processo de Markov; ie, $P(x)$ depende apenas do estados do pai e da distância entre ele e o pai.

Problema:

- Dados:
  1. matriz M descrevendo m caracteres para n espécies
  2. árvore T com n folhas, e com comprimentos t_{vu} conhecidos
- Queremos:
  1. Maximizar $P(M|T)$ encontrando a reconstrução óptima T, nós internos e comprimentos.
- Assumimos;
  1. Caracteres em m são independentes
  2. Modelo de Markov

Problema:



$$L = P(M|T) = \sum_r \sum_v P((r)P_{r\rightarrow s}(t_{rs})P_{r\rightarrow v}(t_{rv})P_{v\rightarrow u}(t_{vu})P_{v\rightarrow w}(t_{vw})$$

- $\sum_r$ e $\sum_v$ variam sobre todas as possibilidades para r e v

Estimando a probabilidade:

- Caracteres são independentes: produtório
- Em geral $L = P(M|T) = \prod_{1 \leq j \leq m} P(M_j|T)$
- logo $L = \prod_{1 \leq j \leq m} \{\sum_{\text{reconstrução} R} P(M_j, R|T)\}$
- e $L = \prod_{1 \leq j \leq m} \{\sum_{\text{reconstrução} R} \prod_{u \rightarrow v} [P(\text{raíz}) P(M_{u \rightarrow v})\}$
- Ineficiente?

Programação Dinâmica

- Proposto por Felsenstein
- Aproveita propriedades de Markov
- $C_j(x, v)$: probabilidade de sub-árvore enraízada em v, dado que v tem etiqueta x
- Folhas: $C_j(x, v) = \begin{cases} 1 & \text{se } V_j = x \\ 0 & \text{nos outros casos} \end{cases}$
- Nós internos, para cada estado x e assumindo filhos u e w:

$$C_j(x, v) = [\sum_y C_j(y, u) P_{x \to y}(t_{uv})][\sum_y C_j(y, w) P_{x \to y}(t_{vw})]$$

- Final: $\prod_{j=1}^{m} [\sum_x C(x, \text{raíz}) P(x)]$

Tamanho dos Ramos Óptimo?

- Problema: não conhecemos t_{rv}!
- Assumindo que apenas não conhecemos um comprimento para a raíz:

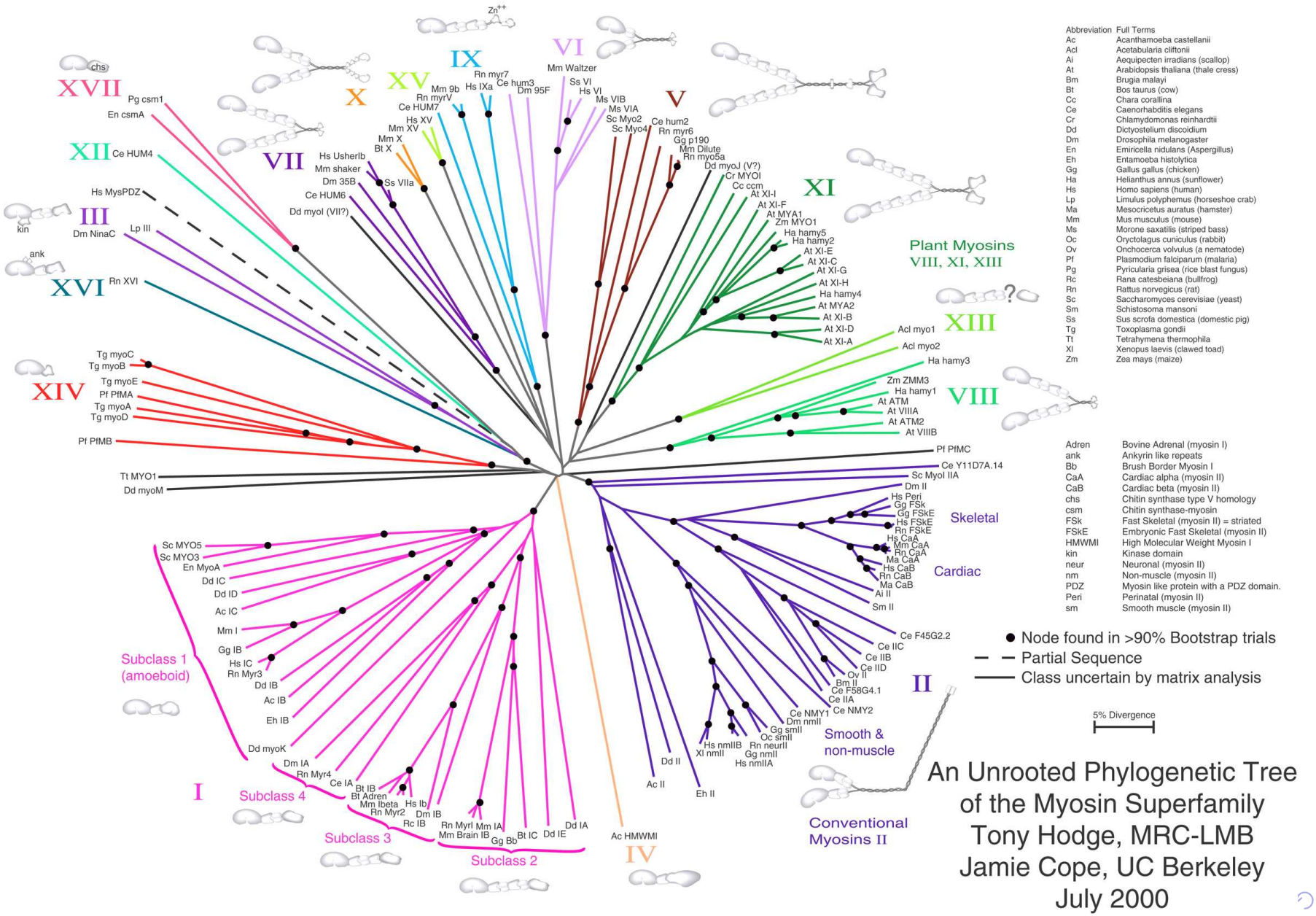
$$logL = \sum_{j=1} mlog[\sum_{x,y} P(x) C_j^v(x,r) P_{x \to y}(t_{rv}) C_j^r(y,v)$$

- Podemos tentar minimizar esta função variando os x, y
- Depois experimentar sobre cada aresta
- Problema:
  - São dependentes
  - Na prática, optimizar cada uma tende a não afectar os outros.

Árvores Filogenéticas

- *Árvore Filogenética*: diagrama mostrando a linha evolucionaria de espécies ou de genes
- Porquê usar árvores:
  - para entender a ascendência de várias espécies
  - para compreender como várias funções evoluíram
  - para informar sobre alinhamentos múltiplos

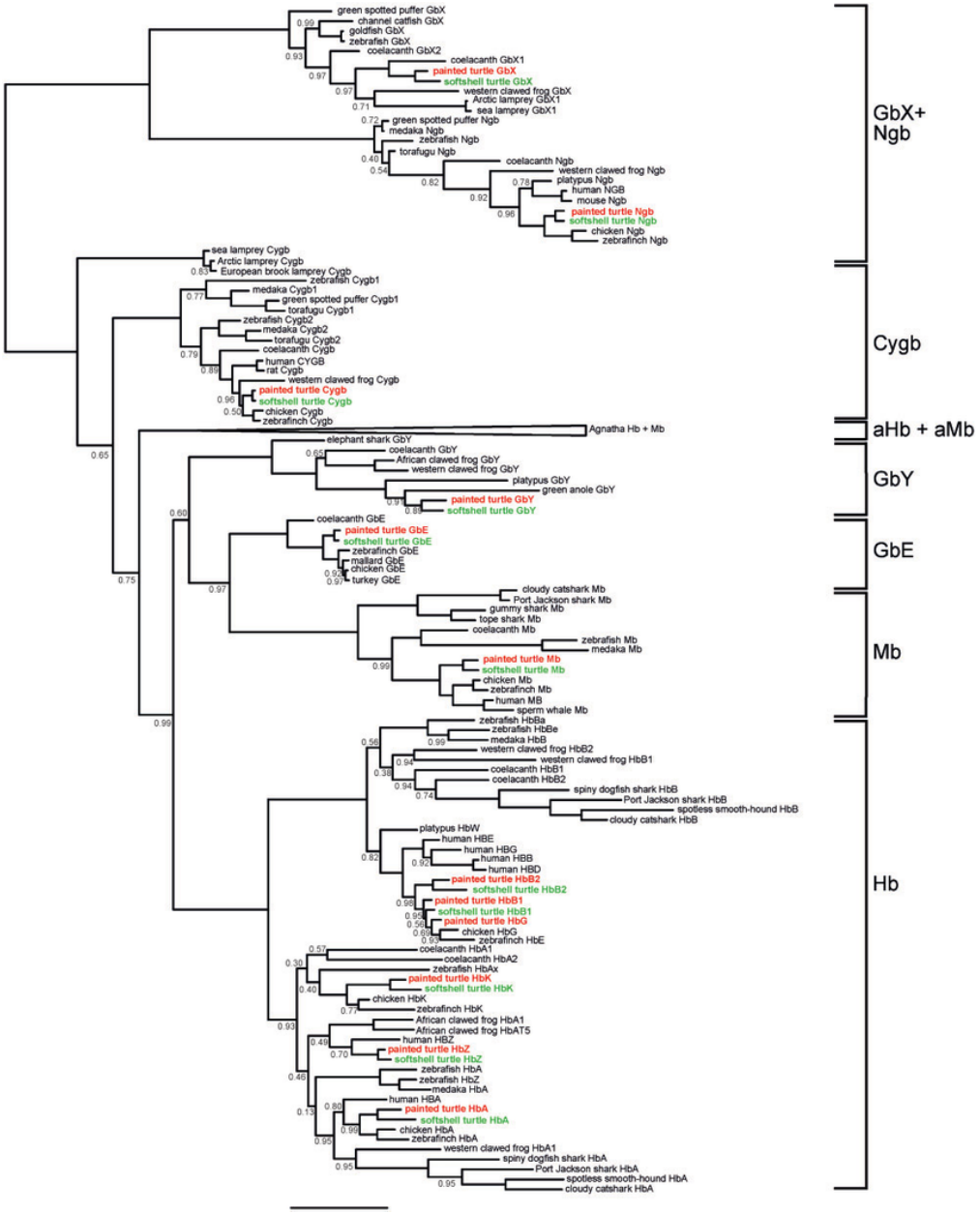
Exemplo de Filogenia: A Árvore da Vida do EMBL



Abbreviation	Full Terms
Ac	Acanthamoeba castellanii
Acl	Acetabularia cliftonii
Ai	Aequipecten irradians (scallop)
At	Arabidopsis thaliana (thale cress)
Bm	Brugia malayi
Bt	Bos taurus (cow)
Cc	Chara corallina
Ce	Caenorhabditis elegans
Cr	Chlamydomonas reinhardtii
Dd	Dictyostelium discoidium
Dm	Drosophila melanogaster
En	Emiricella nidulans (Aspergillus)
Eh	Entamoeba histolytica
Gg	Gallus gallus (chicken)
Ha	Helianthus annus (sunflower)
Hs	Homo sapiens (human)
Lp	Limulus polyphemus (horseshoe crab)
Ma	Mesocricetus auratus (hamster)
Mm	Mus musculus (mouse)
Ms	Morone saxatilis (striped bass)
Oc	Oryctolagus cuniculus (rabbit)
Ov	Onchocerca volvulus (a nematode)
Pf	Plasmodium falciparum (malaria)
Pg	Pyricularia grisea (rice blast fungus)
Rc	Rana catesbeiana (bullfrog)
Rn	Rattus norvegicus (rat)
Sc	Saccharomyces cerevisiae (yeast)
Sm	Schistosoma mansoni
Ss	Sus scrofa domestica (domestic pig)
Tg	Toxoplasma gondii
Tt	Tetrahymena thermophila
Xl	Xenopus laevis (clawed toad)
Zm	Zea mays (maize)

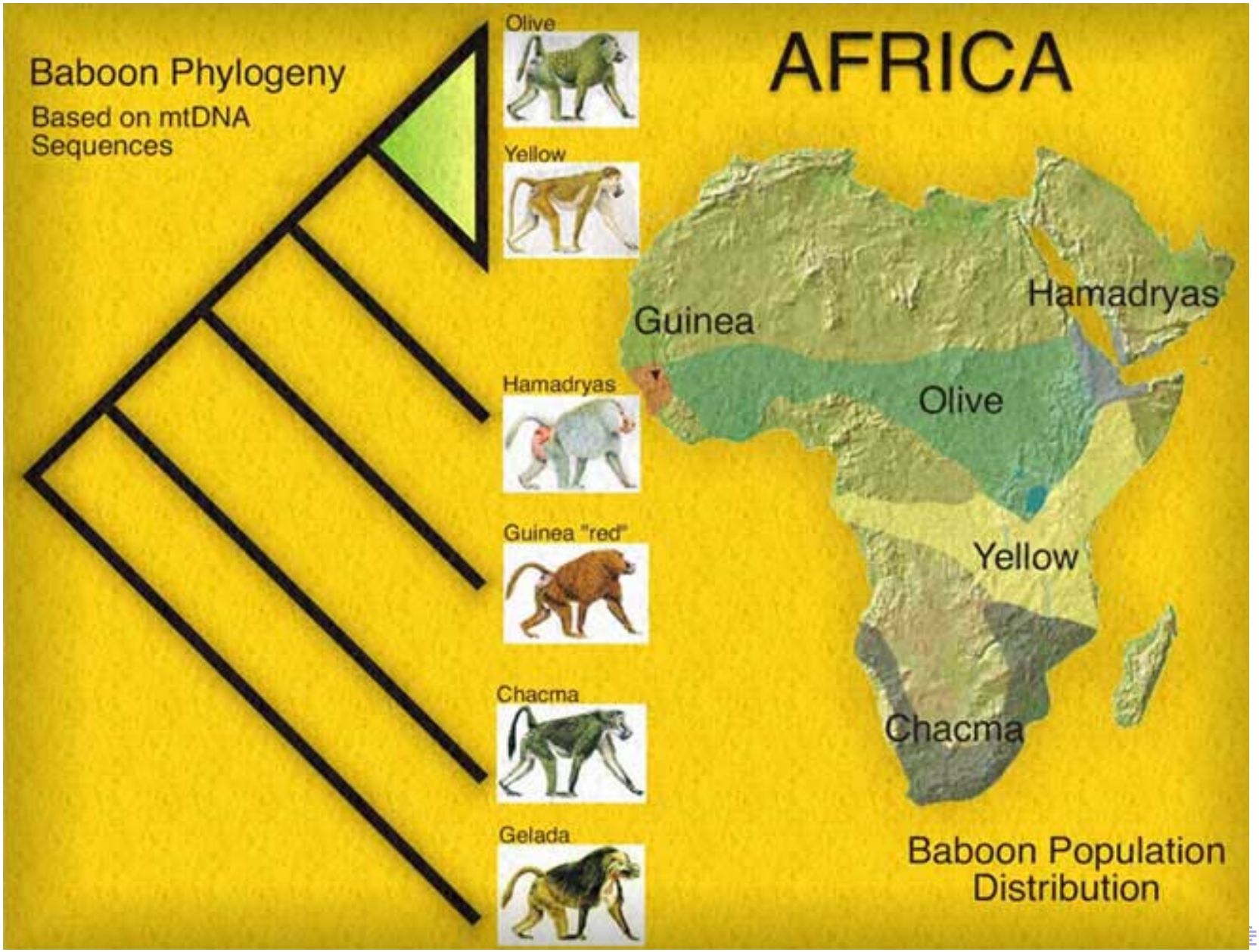
Adren	Bovine Adrenal (myosin I)
ank	Ankyrin like repeats
Bb	Brush Border Myosin I
CaA	Cardiac alpha (myosin II)
CaB	Cardiac beta (myosin II)
chs	Chitin synthase type V homology
csm	Chitin synthase-myosin
FSk	Fast Skeletal (myosin II) = striated
FSkE	Embryonic Fast Skeletal (myosin II)
HMWMI	High Molecular Weight Myosin I
kin	Kinase domain
neur	Neuronal (myosin II)
nm	Non-muscle (myosin II)
PDZ	Myosin like protein with a PDZ domain.
Peri	Perinatal (myosin II)
sm	Smooth muscle (myosin II)

Exemplo de Filogenia: Globinas

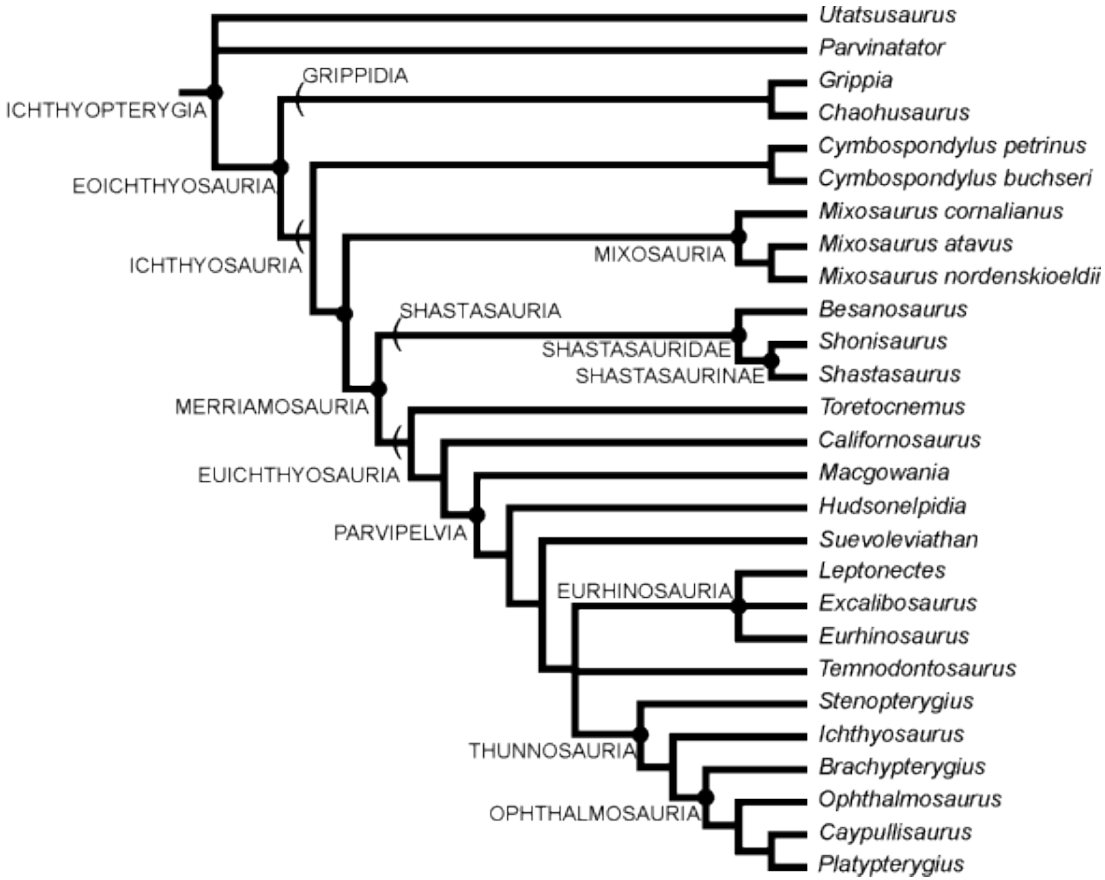


Evolução das globinas nos

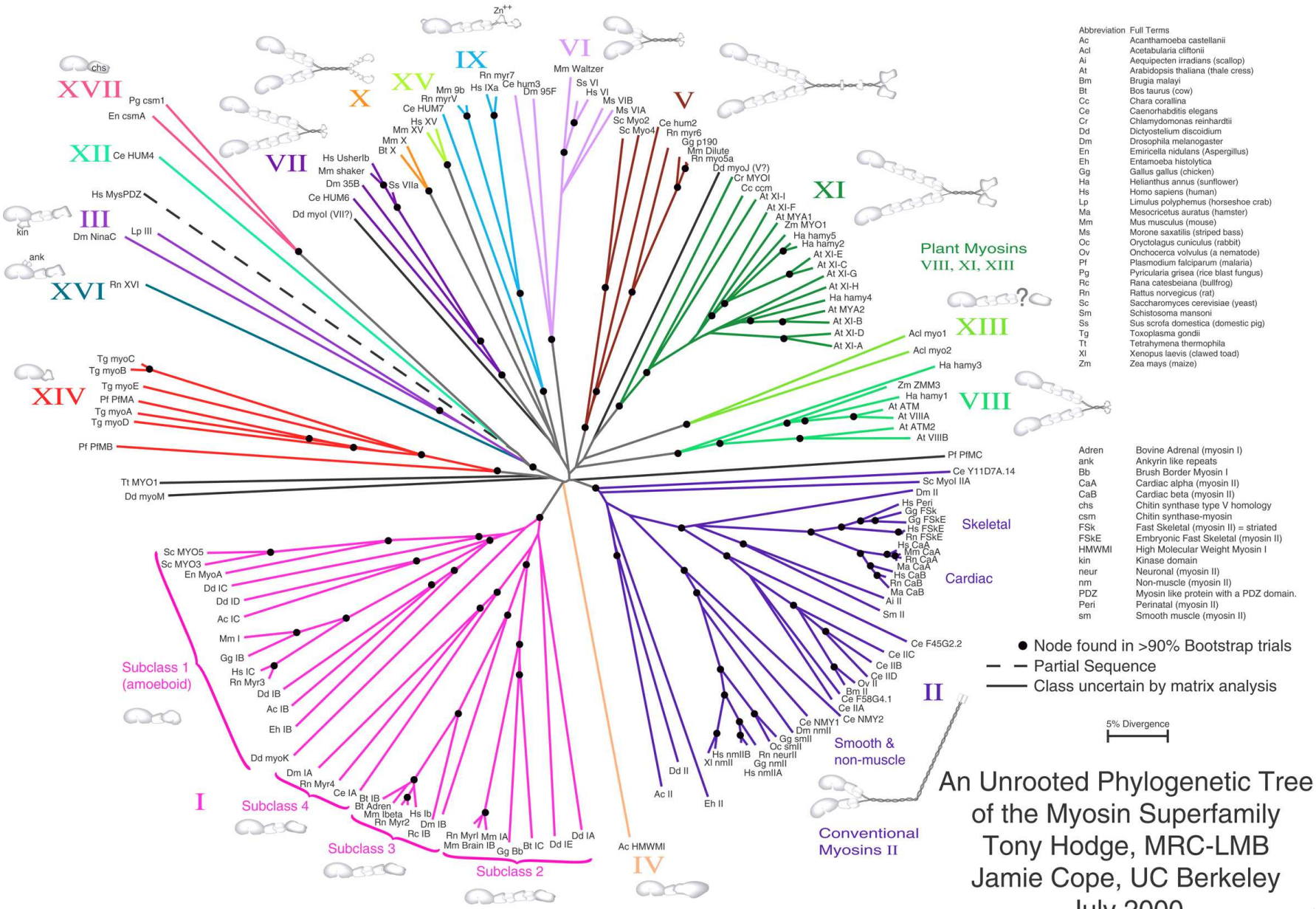
Exemplo de Filogenia: Babuínos



Exemplo de Filogenia: Ichtyosaurus



Exemplo de Filogenia: Myosin



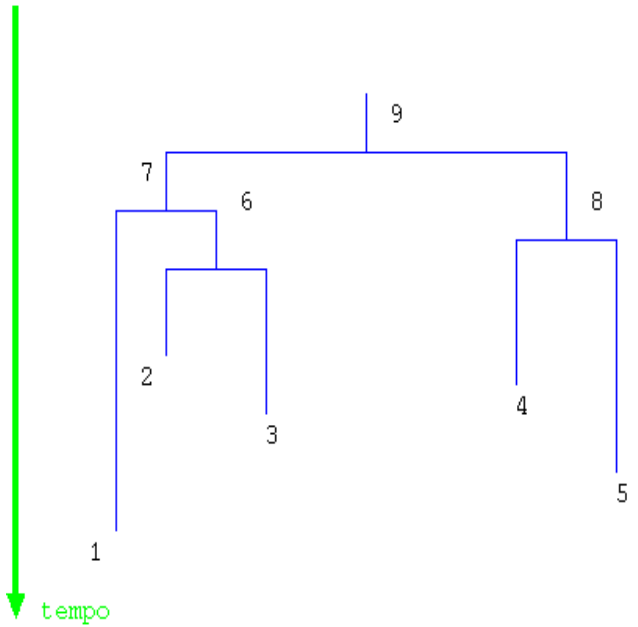
Árvores Filogenéticas: Ideias Básicas

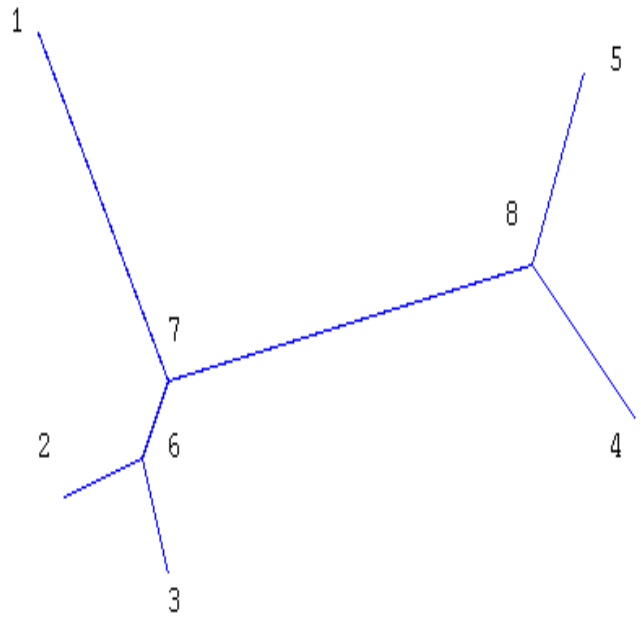
- Folhas representam coisas (genes, indivíduos/famílias, espécies) sendo comparadas
  - o termo *taxão* é usado para referir a esses elementos quando representam espécies e classificações mais amplas de organismos
  - vamos chamá-las de sequências
- nós internos são hipotéticos antepassados
- numa árvore enraizada, um caminho desde a raíz até a um nó representa um caminho evolucionário
- uma árvore não-enraizada representa relações entre coisas, mas não caminhos evolucionários

Dados para Construir Árvores

- Árvores podem ser construídas de vários tipos de dados:
  - *baseados em distâncias*: medidas de distâncias entre espécies/genes
  - *baseados em caracteres*: traços morfológicos (eg, pernas), sequências de DNA/proteínas
  - *ordem de genes*: ordem linear de genes ortológicos encontrados em genomas dados

Árvores Enraizadas e Não-Enraizadas





Número de Árvores Possíveis

- dadas n sequências, existem $\prod_{i=3}^{n}(2i-5)$ árvores não-enraizadas possíveis
- e $(2n-3)\prod_{i=3}^{n}(2i-5)$ árvores enraizadas

Número de Árvores Possíveis

# sequências (n)	# árvores não-enraizadas	# árvores enraizadas
4	3	15
5	15	105
6	105	945
8	10,395	135,135
10	2,027,025	34,459,425

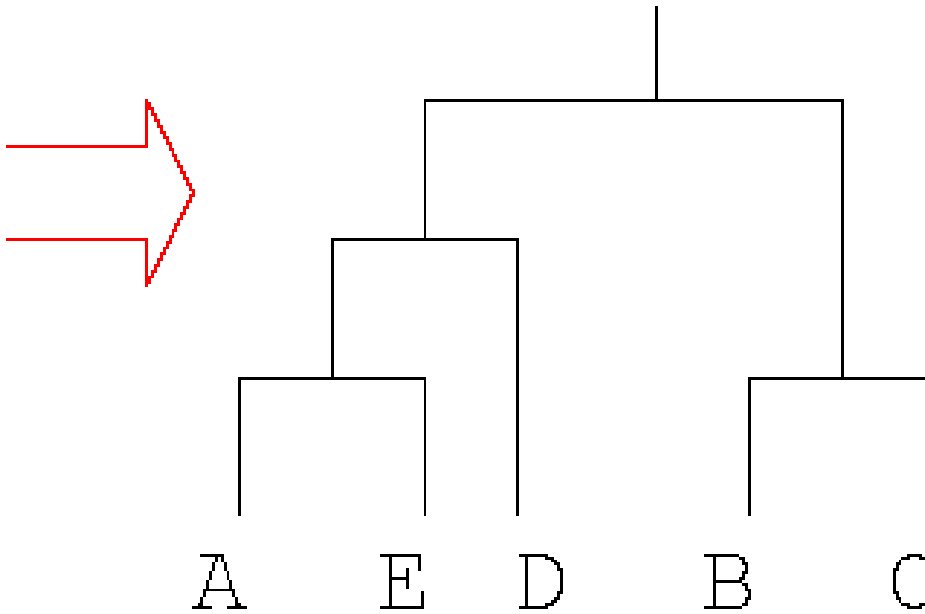
Construção de Árvores Filogenéticas

- Três tipos de métodos gerais:
  - *distância*: encontrar uma árvore que explique as distâncias evolucionárias estimadas
  - *parcimónia*: encontrar a árvore que requer o número mínimo de alterações para explicar os dados
  - *máxima verosimilhança*: encontrar uma árvore que maximize a verosimilhança dos dados

Métodos Baseados em Distância

- **Dados:** uma matriz $n \times n$ M onde M_{ij} é a distância entre os objectos i e j
- **faça:** construa uma árvore pesada nas arestas tal que a distância entre as folhas i e j corresponda a M_ij

	A	B	C	D	E
A	0	8	8	5	3
B		0	3	8	8
C			0	8	8
D				0	5
E					0



O Método UPGMA

- Unweighted Pair Group Method using Arithmetic Averages
- Ideia básica:
  - Iterativamente tirar duas sequências/clusters e agregá-los
  - criar novo nó na árvore para o cluster agregado
- a distância d_{ij} entre os clusters C_i e C_j de sequências é definida como:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

ou distância média entre pares de sequências de cada cluster

Algoritmo UPGMA

- Dar a cada sequência o seu próprio cluster
- definir uma folha para cada sequência e colocar na altura 0
- enquanto há mais de 2 clusters:
  - determinar dois clusters i, j com o menor d_{ij}
  - defina um novo cluster $C_k = C_i \cup C_j$
  - defina um nó k com filhos i e j, coloque-o na altura $d_{ij}/2$
  - substitua os clusters i e j com k
- junte os últimos dois clusters, i e j, pela raíz na altura $d_{ij}/2$

UPGMA

- dado um novo cluster C_k formado pela agregação de C_i e de C_j
- podemos calcular a distância entre C_k e qualquer outro cluster C_i como segue:

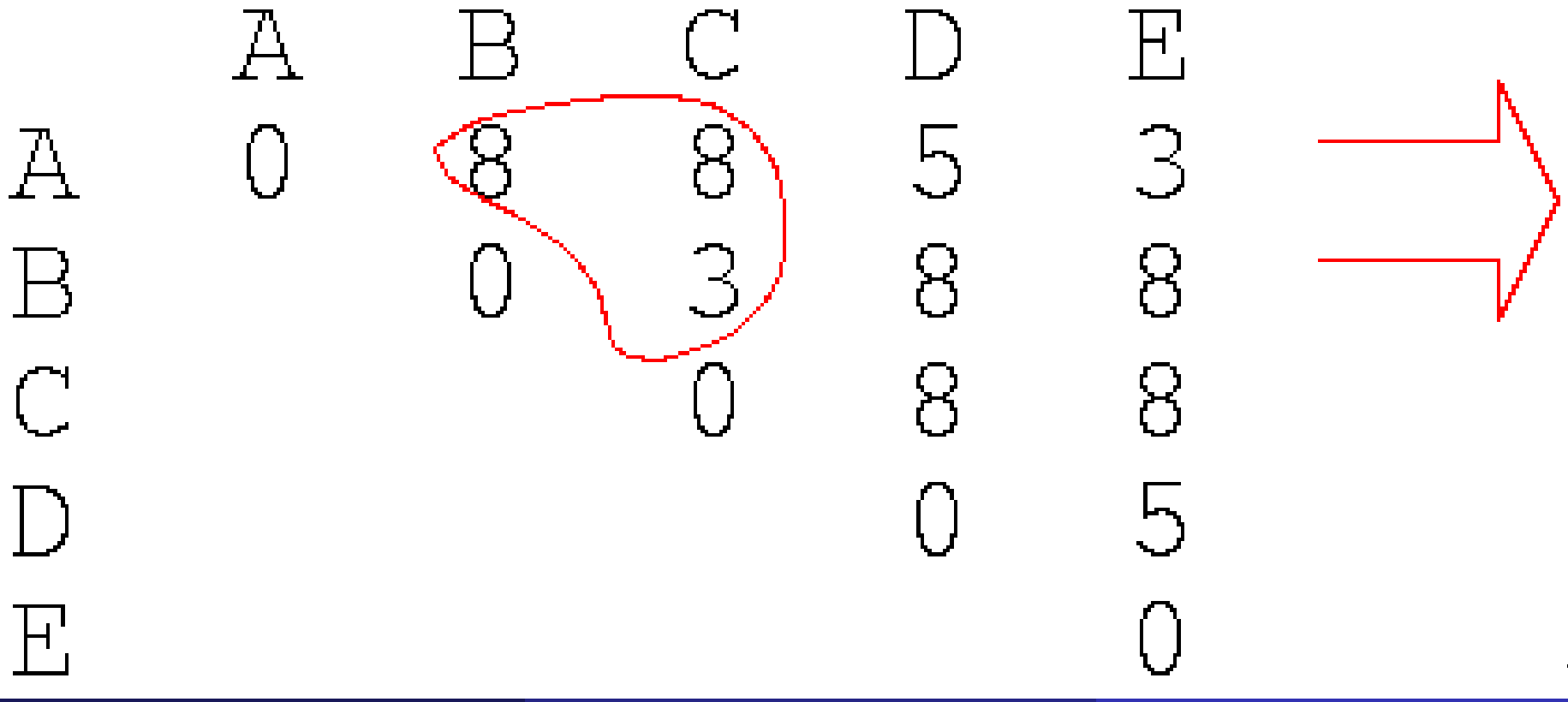
$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$$

A Premissa do Relógio Molecular e Dados Ultramétricos

- A *premissa do relógio molecular*: divergência das sequências é assumida ocorrer à mesma velocidade em todos os pontos da árvore
- esta premissa não é verdade em geral: pressões evolucionárias variam de acordo com o tempo, organismos, genes num organismo e regiões num gene
- se podemos assumir esta premissa, os dados são chamados de *ultramétricos*

Dados Ultramétricos: Condição Necessária e Suficiente

- Dados Ultramétricos: para qualquer tripla de sequências i, j, k as distâncias ou são todas iguais, ou duas são iguais e a restante é menor.



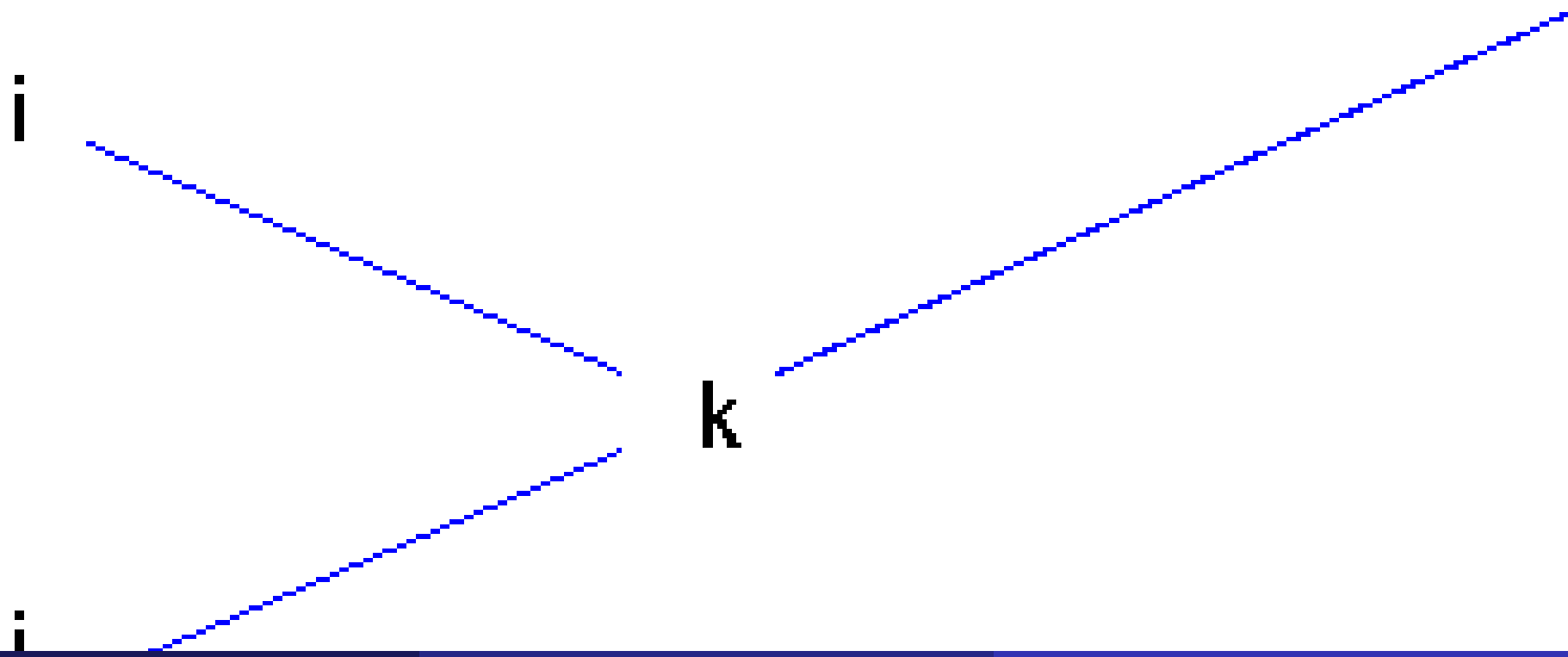
Junção de Vizinhos

- com em UPGMA, construímos uma árvore juntando iterativamente sub-árvores
- diferente de UPGMA:
  - não assumimos o relógio molecular
  - produz árvore não enraizada
- assuma *aditividade*: a distância entre dois pares de folhas é a soma dos comprimentos dos vértices que fazem a ligação.

Distâncias em Junção de Vizinhos

- dado um novo nó interno k, a distância para outro nó m é dada por:

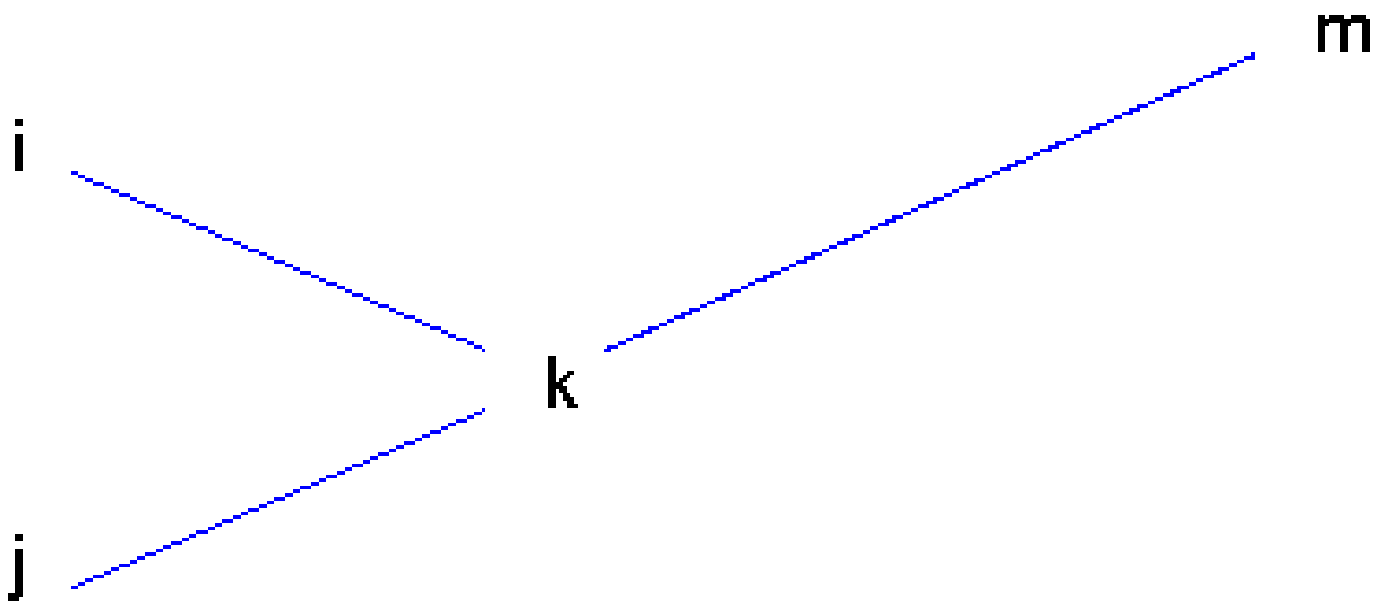
$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$$



Distâncias em Junção de Vizinhos

- Podemos calcular a distância de uma folha para o nó pai na seguinte forma:

$$d_{ik} = \frac{1}{2}(d_{ij} + d_{im} - d_{jm})$$



$$d_{jk} = d_{ij} - d_{ik}$$

Distâncias em Junção de Vizinhos

- Podemos generalizar esta regra de forma a tomar em conta a distância para todas as outras folhas:

$$d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$$

onde

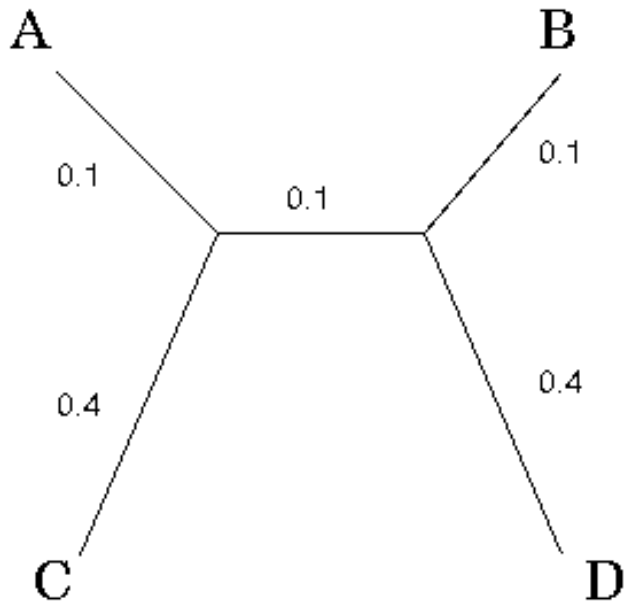
$$r_i = \frac{1}{|L| - 2} \sum_{m \in L} d_{im}$$

e L é o conjunto das folhas

- isto é mais robusto se os dados não forem estritamente aditivos

Juntar que Nós?

- Em cada passo escolhemos um par de nós para juntar. Devemos escolher os nós com o menor d_{ij}?
- Suponhamos que a árvore verdadeira parece como isto e que estamos a escolher os primeiros nós para juntar:



$$d_{AB} = 0.3$$
$$d_{AC} = 0.5$$

- Decisão errada em juntar A e B: precisamos de considerar distância do par até outras folhas.

Juntar que Nós?

- Para evitar o problema escolha o par de nós baseado nas distâncias baseado em D_{ij}:

$$D_{ij} = d_{ij} - (r_i + r_j)$$

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}$$

Algoritmo de Junção de Vizinhos

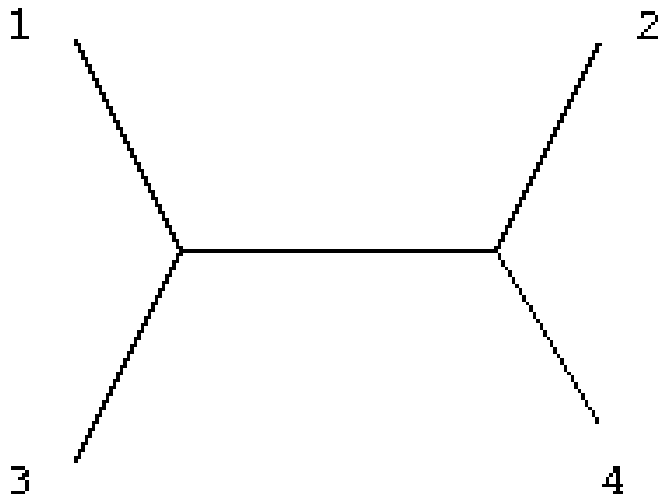
- defina a árvore T como o conjunto de nós folhas
- $L = T$
- enquanto há mais que duas sub-árvores em T:
  - escolha o par i, j em L com D_{ij} mínimo
  - adicione a T um novo nó agregando i e j
  - determine novas distâncias:

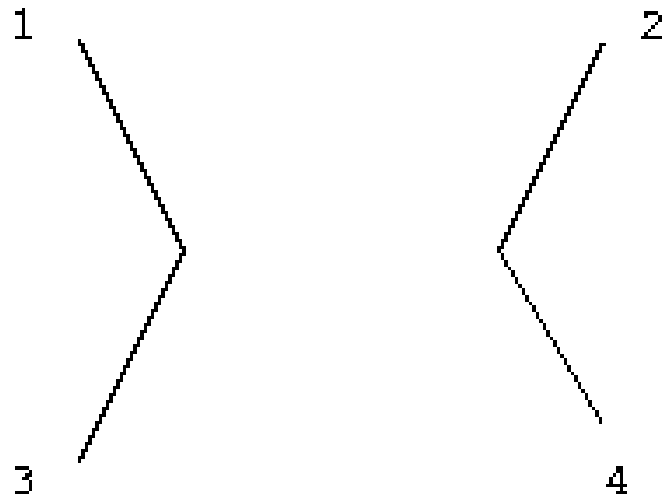
$$d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$$
$$d_{jk} = d_{ij} - dik$$
$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij}) \text{para todos os outros} m \in L$$

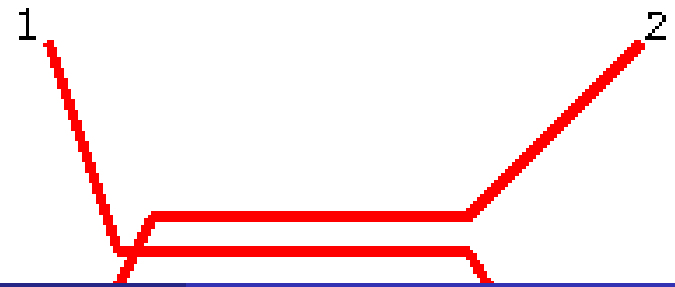
  - remova i e j de L e insira k (processe-o como se uma folha)
- junte as duas árvores restantes, i e j com um vértice de comprimento d_{ij}

Testando Aditividade

- Para qualquer conjunto de qualquer folhas i, j, k, l duas das distâncias $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$ e $d_{il} + d_{jk}$ devem ser iguais e maiores que a terceira distância







Escolhendo Raízes

- Escolher uma raíz para árvores não-enraizadas é muitas vezes feita usando um “outgroup”
- *Outgroup* é uma espécie que se sabe ser mais diferentes das outras espécies do que elas são entre elas.
- o ponto onde o outgroup se junta ao resto da árvore é o melhor candidato para a raíz.

Comentários Sobre Métodos Baseados em Distância

- Se os dados de distância são ultramétricos (e as distâncias são distâncias genuínas), então UPGMA encontra a árvore certa
- Se os dados são aditivos (e as distâncias são distâncias genuínas), então junção de vizinhos identifica a árvore correcta
- senão, os métodos podem não recuperar a árvore correcta, mas são boas heurísticas

Construção de Árvores Filogenéticas

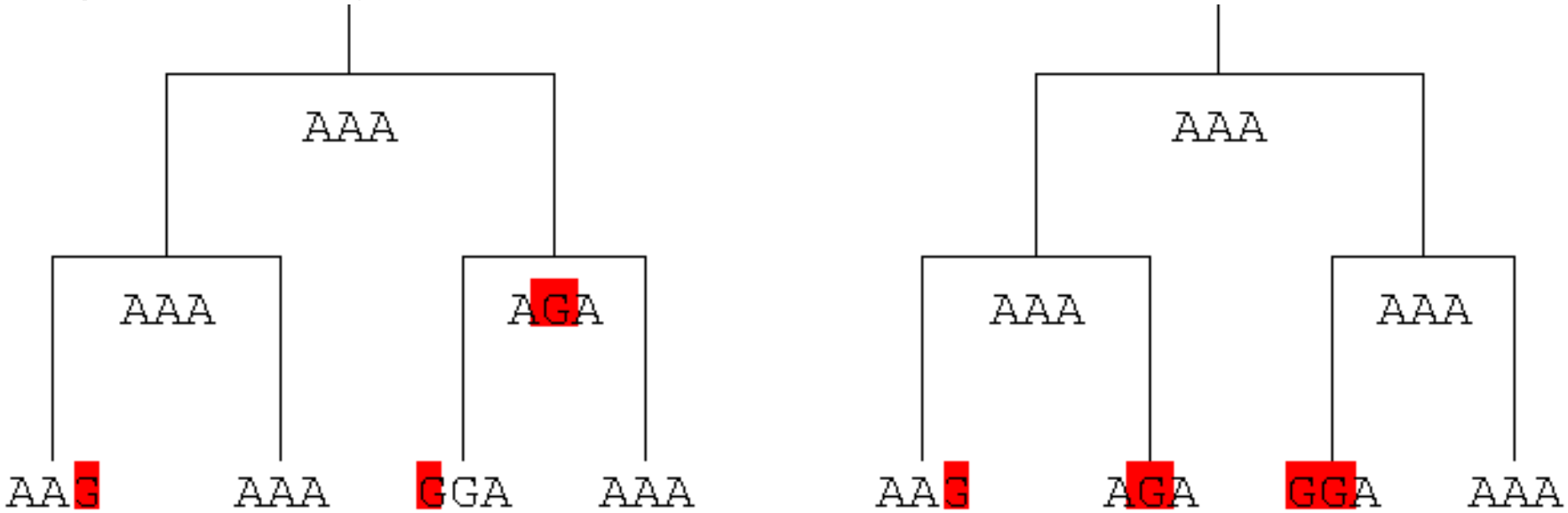
- Três tipos de métodos gerais:
  - *distância*: encontrar uma árvore que explique as distâncias evolucionárias estimadas
  - *parcimónia*: encontrar a árvore que requer o número mínimo de alterações para explicar os dados
  - *maximum likelihood*: encontrar uma árvore que maximize a verosimilhança dos dados

Métodos Baseados em Parcinómia

- *dado*: dados baseados em caracteres
- *faça*: encontrar árvore que explique os dados com o número mínimo de alterações.

Exemplo de Parcinómia

- existem muitas árvores que podem explicar a filogenia das sequências seguintes: AAG, AAA GGA, AGA.



- parcimónia prefere a primeira árvore porque requer menor número de substituições

Métodos Baseados em Parcimónia

- habitualmente estes métodos envolvem dois componentes:
  - uma procura pelo espaço das árvores
  - um processamento para explicar o menor número de mudanças necessárias para explicar os dados (para uma dada topologia).

Encontrar Menor Número de Mudanças Numa Árvore

- Algoritmo de Fitch [1971]:
  - assume qualquer estado (nucleotído, amino-ácido) e pode converter para qualquer outro estado
  - assume que as posições são independentes

Algoritmo de Fitch

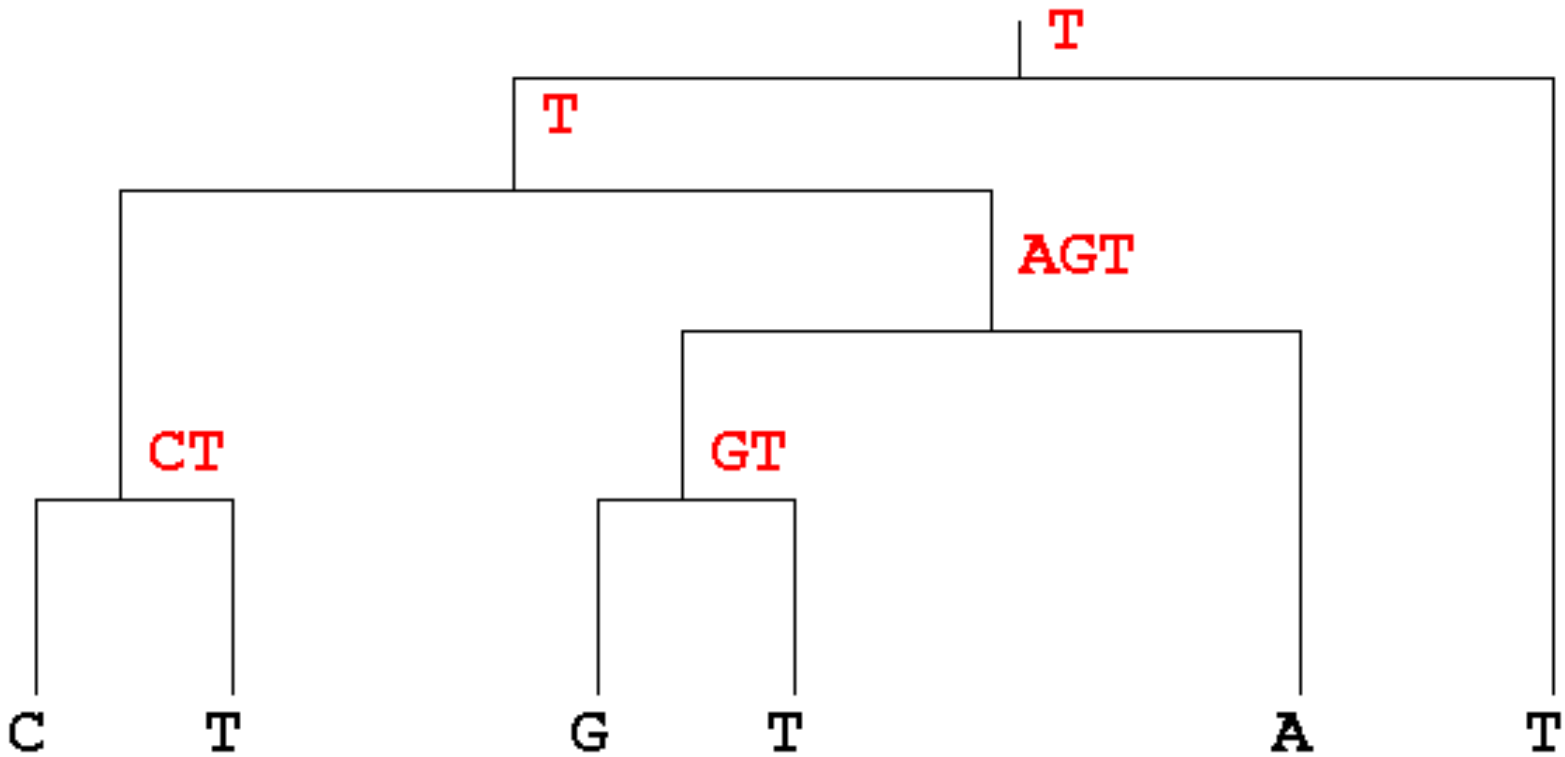
- atravessa a árvore desde as folhas até à raíz determinando o número possível de *estados* (eg, nucleotídos) que podem ser tomados por cada nó interno.
- atravessa a árvore desde a raíz até às folhas estabelecendo os estados para os nós internos.

Passo 1: Estado Possível para os Nós Internos

- atravesse a árvore em pós-ordem (desde as folhas até à raíz)
- determinar os estados possíveis R_i do nó interno i com filhos j e k:

$$R_i = \begin{cases} R_j \cup R_k, & \text{se} R_j \cap R_k = \emptyset \\ R_j \cap R_k, & \text{senão} \end{cases}$$

O Algoritmo de Fitch: Passo 1



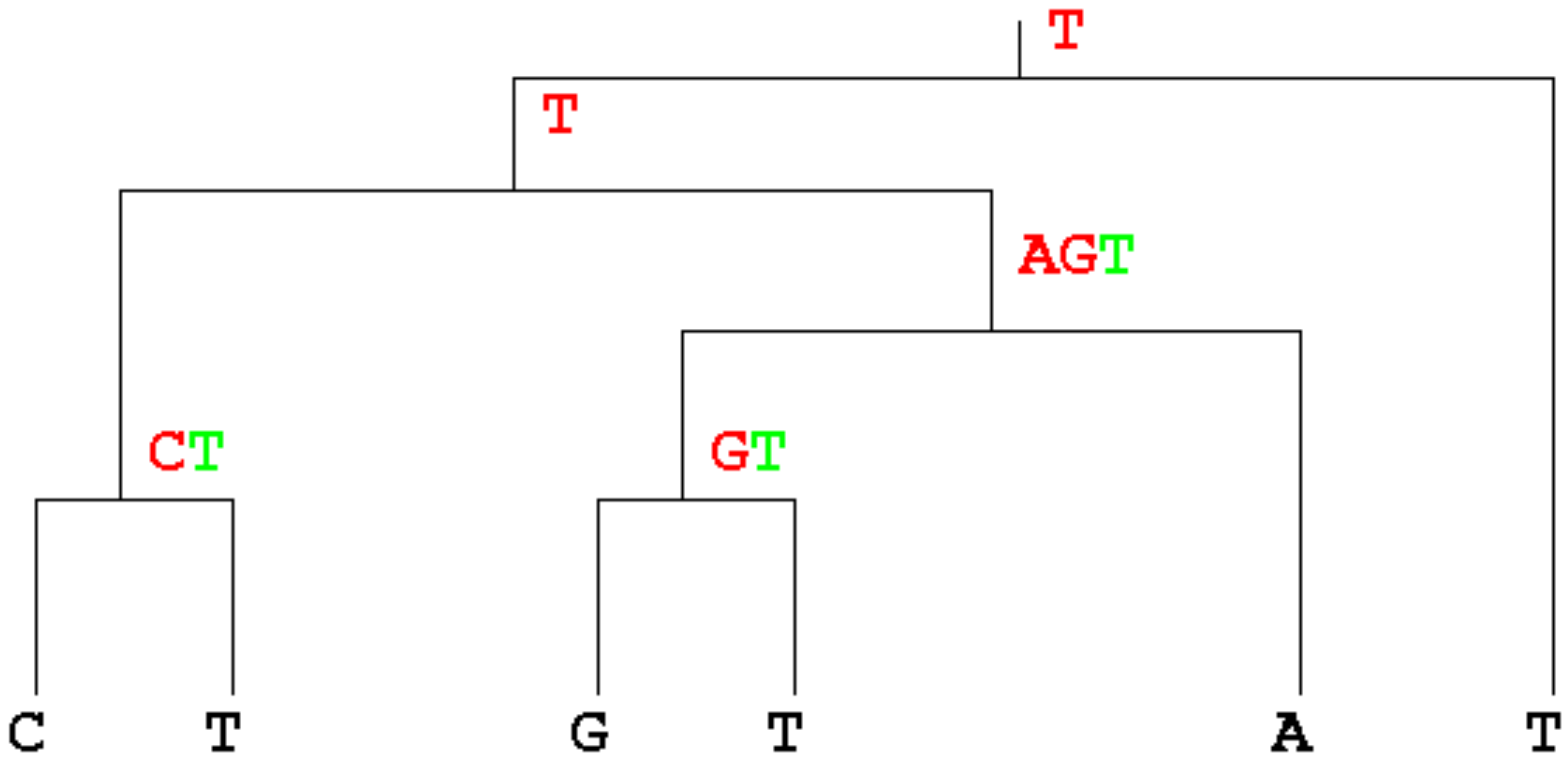
- # de mudanças = # de uniões

O Algoritmo de Fitch: Passo 2

- atravesse a árvore em pré-ordem (desde a raíz até às folhas)
- seleccionar um estado r_j do nó interno j com pai i:

$$r_j = \begin{cases} r_i, & \text{se} r_i \in R_j \\ \text{estado arbitrário} \in R_j, & \text{senão} \end{cases}$$

O Algoritmo de Fitch: Passo 2



O Algoritmo de Sankoff

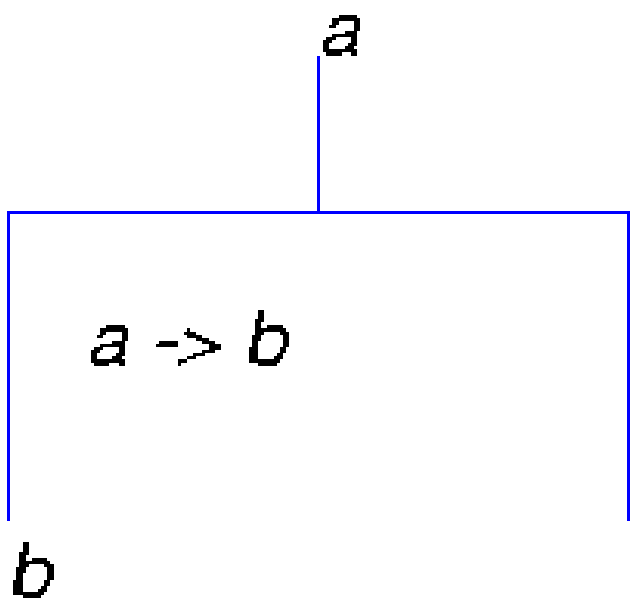
- Sankoff & Cedergren [1983]
- Em vez de assumir que todos as mudanças de estado são igualmente prováveis, use custos diferentes $S(a, b)$ para mudanças diferentes
- primeiro passo do algoritmo é propagar custos subindo na árvore:

$$a \rightarrow b$$

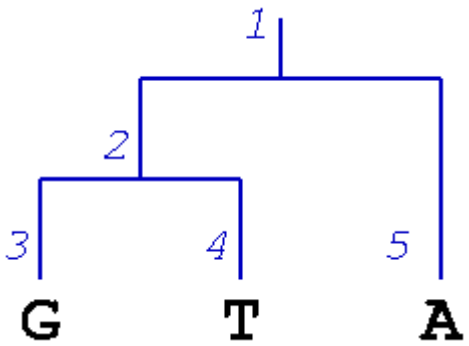
O Algoritmo de Sankoff

- para um nó interno i com filhos j e k

$$R_i(a) = min_b(R_j(b) + S(a,b)) + min_b(R_k(b) + S(a,b))$$



O Algoritmo de Sankoff



- $R_3[\mathtt{A}] = \infty, R_3[\mathtt{C}] = \infty, R_3[\mathtt{G}] = 0, R_3[\mathtt{T}] = \infty$
- $R_4[\mathtt{A}] = \infty, R_4[\mathtt{C}] = \infty, R_4[\mathtt{G}] = \infty, R_4[\mathtt{T}] = 0$
- $\begin{cases} R_2[\mathtt{A}] = R_3[\mathtt{G}] + S(\mathtt{A},\mathtt{G}) + R_4[\mathtt{T}] + S(\mathtt{A},\mathtt{T}) \\ \ldots \\ R_2[\mathtt{T}] = R_3[\mathtt{G}] + S(\mathtt{A},\mathtt{T}) + R_4[\mathtt{T}] + S(\mathtt{T},\mathtt{T}) \end{cases}$
- $R_5[\mathtt{A}] = 0, R_5[\mathtt{C}] = \infty, R_5[\mathtt{G}] = \infty, R_5[\mathtt{T}] = \infty$
- $\begin{cases} R_1[\mathtt{A}] = min(R_2[\mathtt{A}] + S(\mathtt{A},\mathtt{A}), \ldots, R_2[\mathtt{T}] + S(\mathtt{A},\mathtt{T})) + R_5[\mathtt{A}] + S(\mathtt{A},\mathtt{A}) \\ \ldots \\ R_1[\mathtt{T}] = min(R_2[\mathtt{A}] + S(\mathtt{T},\mathtt{A}), \ldots, R_2[\mathtt{T}] + S(\mathtt{T},\mathtt{T})) + R_5[\mathtt{A}] + S(\mathtt{A},\mathtt{T}) \end{cases}$

O Algoritmo de Sankoff: Passo 2

- faça uma travessia em pré-ordem da árvore (desde a raíz para as folhas)
- seleccione o caracter de menor custo para cada nó

Explorando o Espaço das Árvores

- Nós consideramos como encontrar o menor número de mudanças para cada topologia
- precisa de um método para procurar no espaço das árvores

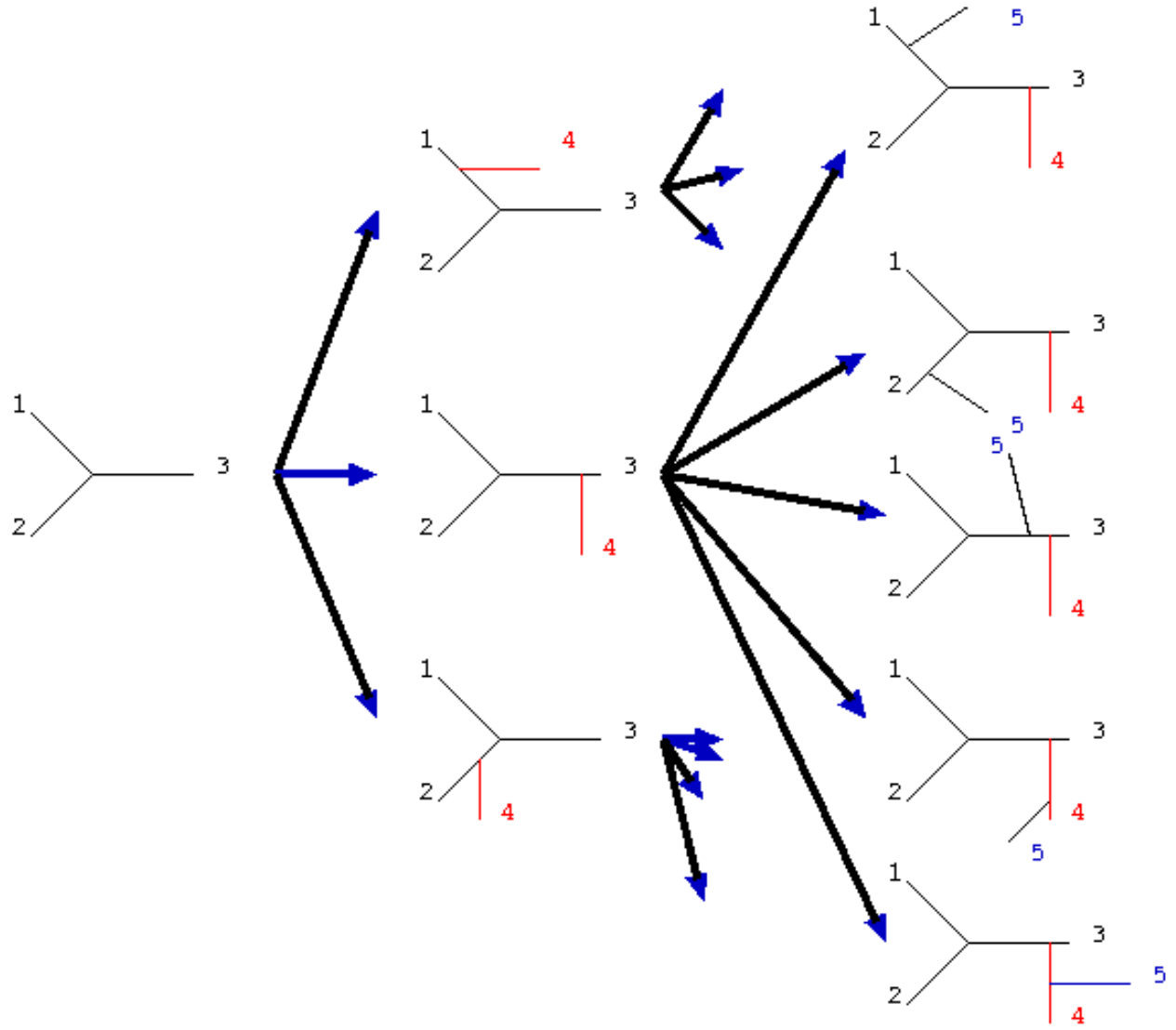
Métodos de Procura

- exaustiva
- branch & bound:
  - encontre um árvore inicial (eg, por UPGMA ou por junção de vizinhos) e determine o custo
  - use procura para encontrar outras árvores:
    - abandone árvores parciais cujo custo excede a árvore de menor custo até agora
- métodos gulosos: eg, troca de ramos

Procura Por Branch & Bound

- procure pelo espaço das árvores sem raíz:
  - adicione folhas à árvore incrementalmente
  - mantenha a árvore de custo menor completa até agora T'
  - corte uma árvore T e os seus descendentes se $custo(T) > custo(T')$
- Propriedade Chave: adicionar folhas só pode aumentar o custo da árvore

Procura Por Branch & Bound



Algoritmo

- Para n sequências mantenha um vector de contadores:

$$[i_3][i_5][i_7]\ldots[i_{2n-5}]$$

  onde i_k toma os valores $0 \ldots k$
- uma árvore completa é representada por uma atribuição de todos os i_k a valores não-zero.
- i_k indica, com uma árvore parcial com k vértices, onde adicionar um ramo para a sequência seguinte
- $i_k = 0$ indica uma árvore parcial

Algoritmo

- Para procurar o espaço, rode contadores através dos seus valores possíveis (como se fossem odómetros):
  - contadores mais à direita mudam mais depressa
  - quando um contador é zero, os contadores à direita devem ser 0 também
  - teste o custo (parcial) da árvore em cada tick
  - faça com que o odómetro salte quando há um corte

Algoritmo

- É um método completo
  - garantido encontrar solução óptima
- frequentemente muito mais eficiente que procura exaustiva
- no pior caso, não é melhor
- a eficiência depende da qualidade da árvore inicial

Comentários sobre Inferência de Árvores

- o espaço de procura pode ser grande, mas pode encontrar a árvore óptima eficientemente em alguns casos
- em alguns casos métodos heurísticos podem ser aplicados
- difícil avaliar filogenias inferidas: a verdade-alvo não é habitualmente sabida:
  - podemos olhar para a concordância entre diferentes fontes de evidência
  - quando a procura não é completa, podemos procurar repetibilidade em sub-amostras dos dados
- alguns métodos novos usam dados baseados na ordem linear dos genes ortológicos no cromossoma
- filogenia de bactérias e vírus não é trivial devido a transferências laterais de material genético: *filogenias locais* podem ser mais apropriadas

Comentários sobre Inferência de Árvores

Um visão diferente:

- Aula de Felsenstein
- Aula de Shamir
- Phylip
- PAUP*
- Molphy
- PAML
- MrBayes