Procura de motifs

Motifs (de sequências)

Motifs são **padrões** recorrentes em sequências biológicas que têm, ou se supõe terem, uma função biológica.

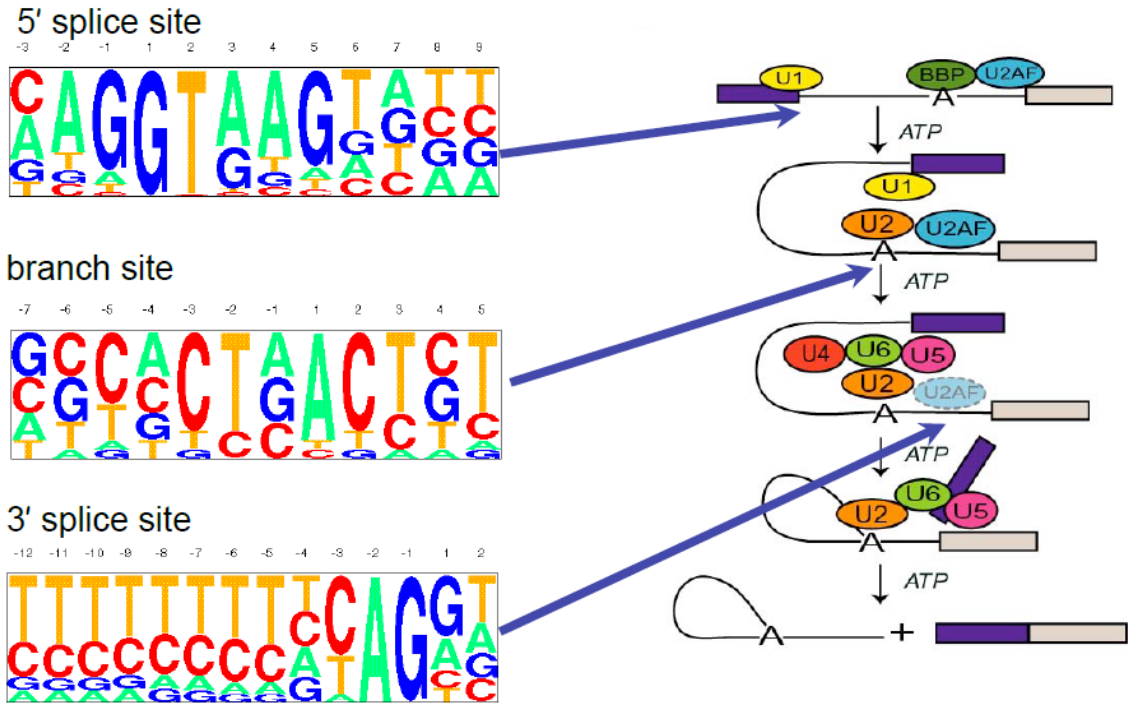
Alguns exemplos de motifs:

- em sequências de **DNA** incluem, por exemplo, locais de ligação de proteínas regulatórias ao DNA, controlando a transcrição;

E. Coli consensus motifs -10 TATAAT e -35 TTGACA.

Motifs (de sequências)

- em sequências de **proteínas** incluem, entre outros, a representação de domínios conservados de proteínas com funções biológicas determinadas (e.g. locais de ligação de enzimas a substratos ou outras moléculas);
- no processamento de **RNA** (splicing, edição, poliadenilação) e terminação de transcrição.



Splicing motifs (Humanos)

Dificuldades

- Não conhecemos a sequência do motif;
- Não sabemos onde está localizado (e.g., Regiões regulatórias variam entre 100 e 1000 bp);
- Os motifs podem variar ligeiramente de um gene para o outro (normalmente 1 ou 2 mutações do consensus).
- Como discernir um motif de outro totalmente aleatório?

Vários tipos de padrões:

- **Blocos** (blocks) ou **motifs** – sem espaçamentos, determinísticos (sub-sequências constantes) (**Consensus**)

- **Patterns** – com espaçamentos, determinísticos (representados por expressões regulares)

- **Perfis** (profiles) – estocásticos (definem probabilidades por posição)

Perfis (probabilísticos)

- Um **perfil** pode ser representado por uma matriz (*position weighted matrix* – **PWM**):
  - Colunas representam as posições do motif;
  - Linhas representam os possíveis carateres do alfabeto;
  - Posições da matriz indicam probabilidade de aparecer o carácter nessa posição.
  - O PWM é obtido a partir da contagem das ocorrências de cada símbolo do alfabeto em cada posição.

Binding sites

CCCATTGTTCTC
TTTCTGGTTCTC
TCAATTGTTTAG
CTCATTGTTGTC
TCCATTGTTCTC
CCTATTGTTCTC
TCCATTGTTCGT
CCAATTGTTTTG

Matriz de ocorrências

A	0	0	2	7	0	0	0	0	0	1	0
C	4	6	4	1	0	0	0	0	5	0	5
G	0	0	0	0	1	8	0	0	1	1	2
T	4	2	2	0	8	7	0	8	8	2	6

PWM

A	0	0	$\frac{1}{4}$	$\frac{7}{8}$	0	0	0	0	0	$\frac{1}{8}$	0
C	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	0	0	0	0	$\frac{5}{8}$	0	$\frac{5}{8}$
G	0	0	0	0	$\frac{1}{8}$	1	0	0	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$
T	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	1	$\frac{7}{8}$	0	1	1	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{1}{8}$

Motif logo

Perfis (probabilísticos)

- Diferentes organismos podem ter diferentes GC contents;
- Podemos usar **log-odds** para refletir a frequência esperada de cada nucleótido;
- Estas probabilidades podem ser convertidas em scores usando a mesma estratégia da geração de matrizes de substituição (log odds) – neste caso ficamos com uma *Position Specific Scoring Matrix* (PSSM):

$S_i = \log \frac{p_i}{q_i}$ onde p_i é a frequência observada e q_i a frequência esperada.

Motif logo

Probabilidades de geração de sequências

Dado um perfil probabilístico:

P =

A	1/2	7/8	3/8	0	1/8	0
C	1/8	0	1/2	5/8	3/8	0
T	1/8	1/8	0	0	1/4	7/8
G	1/4	0	1/8	3/8	1/4	1/8

A probabilidade do consenso:

$$Prob(\mathbf{aaacct}|\mathbf{P}) = 1/2 \times 7/8 \times 3/8 \times 5/8 \times 3/8 \times 7/8 = .033646$$

Probabilidade de outra string, atacag :

$$Prob(\mathbf{atacag}|\mathbf{P}) = 1/2 \times 1/8 \times 3/8 \times 5/8 \times 1/8 \times 1/8 = .000229$$

Nota: O produto das probabilidades supõe a independência das posições, ou seja, a ocorrência de um símbolo numa posição não influência a ocorrência de outro símbolo noutra posição.

Sequência mais provável de P

Dada uma sequência S e o perfil **P** podemos calcular a subsequência de S de tamanho *L* com a maior probabilidade de ter sido "gerada" por **P**.

P =

A	1/2	7/8	3/8	0	1/8	0
C	1/8	0	1/2	5/8	3/8	0
T	1/8	1/8	0	0	1/4	7/8
G	1/4	0	1/8	3/8	1/4	1/8

S = ctataaaccttacatc

Sequência mais provável de P

P =

A	1/2	7/8	3/8	0	1/8	0
C	1/8	0	1/2	5/8	3/8	0
T	1/8	1/8	0	0	1/4	7/8
G	1/4	0	1/8	3/8	1/4	1/8

Calcular *prob*(**a**|**P**) para cada segmento de tamanho *L* (=5)

Tent. 1: **c t a t a a a c c t t a c a t c**

Tent. 2: **c t a t a a a c c t t a c a t c**

Tent. 3: **c t a t a a a c c t t a c a t c**

… continuar o processo com todas as possibilidades.

Sequência mais provável de P

Calcular *prob*(**a**|**P**) para cada segmento de tamanho *L*

String (a vermelho)	Calculo	*prob*(**a**|**P**)
ctataaaccttacat	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 7/8 x 0 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/8 x 7/8 x 3/8 x 0 x 3/8 x 0	0
ctataaaccttacat	1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8	**.0336**
ctataaaccttacat	1/2 x 7/8 x 1/2 x 5/8 x 1/4 x 7/8	.0299
ctataaaccttacat	1/2 x 0 x 1/2 x 0 1/4 x 0	0
ctataaaccttacat	1/8 x 0 x 0 x 0 x 0 x 1/8 x 0	0
ctataaaccttacat	1/8 x 1/8 x 0 x 0 x 3/8 x 0	0
ctataaaccttacat	1/8 x 1/8 x 3/8 x 5/8 x 1/8 x 7/8	.0004

aaacct é a sequência mais provável!

Inferência de motifs a partir de conjuntos de sequências

Em sequência de DNA os motifs têm tamanho constante e são frequentemente repetidos e conservados, mas, ao mesmo tempo, são de tamanho reduzido, cerca de 6 a 12 pb, e as regiões intergênicas são muito longas e altamente variáveis tornando a descoberta de motifs uma tarefa difícil.

Descoberta dos melhores motifs define **problemas de otimização** que variam conforme o tipo de motif considerado (determinístico, probabilístico) e a função objectivo que define a "qualidade" dos motifs.

Os problemas de otimização são complexos e requerem algoritmos elaborados para a sua resolução.

Definição do problema

Dado um conjunto de ***t*** sequências (***seqs***) de DNA, todas com o mesmo comprimento ***n***, pretende-se identificar o motif de comprimento ***L*** e o vetor de posições iniciais ***S***=(s_1, s_2,… s_t), onde s_i é posição inicial do motif na sequência *i = 1..t.*

```
cctgatagacgctatctggctatccacgtacgtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatacgtacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtacgtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaacgtacgtc
```

O problema da descoberta de motifs

Sequências não alinhadas

```
agggcactagcccatgtgagagggcaaggaccagcggaag
taattcagggccaggatgtatctttctcttaaaaataaca
tatcctacagatgatgaatgcaaatcagcgtcacgagctt
tggcgggcaaggtgcttaaaagataatatcgaccctagcg
attcgggtaccgttcataaaagtacgggaatttcgggtag
gttatgttaggcgagggcaaaagtcatatacttttaggtc
aagagggcaatgcctcctctgccgattcggcgagtgatcg
gatggggaaaatatgagaccaggggagggccacactgcag
ctgccgggctaacagacacacgtctagggctgtgaaatct
gtaggcgccgaggccaacgctgagtgtcgatgttgagaac
attagtccggttccaagagggcaactttgtatgcaccgcc
gcggcccagtgcgcaacgcacagggcaaggtttactgcgg
ccacatgcgagggcaacctccctgtgttgggcggttctga
gcaattgtaaaacgacggcaatgttcggtcgcctaccctg
gataaagaggggggtaggaggtcaactcttccgtattaat
aggagtagagtagtgggtaaactacgaatgcttataacat
gcgagggcaatcgggatctgaaccttctttatgcgaagac
tccaggaggaggtcaacgactctgcatgtctgacaacttg
gtcatagaattccatccgccacgcggggtaatttggacgt
gtgccaacttgtgccgggggggctagcagcttcccgtcaaa
cgcgtttggagtgcaaacatacacagcccgggaatataga
aagatacgagttcgatttcaagagttcaaaacgtgacggg
gacgaaacgagggcgatcaatgcccgataggactaataag
tagtacaaacccgctcacccgaaaggagggcaaatacctt
atatacagccaggggagacctataactcagcaaggttcag
cgtatgtactaattgtggagagcaaatcattgtccacgtg
...
```

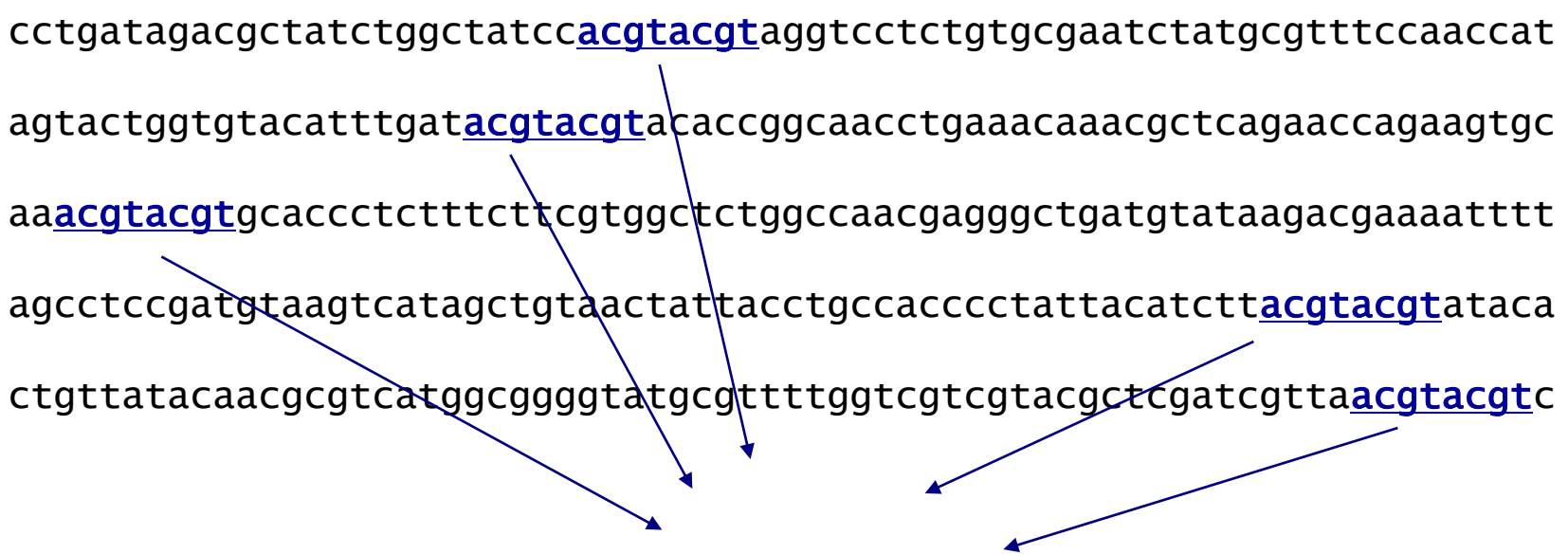
Sequências alinhadas

```
gcggaagagggcactagcccatgtgagagggcaaggacca
atctttctcttaaaaataacataattcagggccaggatgt
gtcacgagctttatcctacagatgatgaatgcaaatcagc
taaaagataatatcgaccctagcgtggcgggcaaggtgct
gtagattcgggtaccgttcataaaagtacgggaatttcgg
tatacttttaggtcgttatgttaggcgagggcaaaagtca
ctctgccgattcggcgagtgatcgaagagggcaatgcctc
aggatggggaaaatatgagaccaggggagggccacactgc
acacgtctagggctgtgaaatctctgccgggctaacagac
gtgtcgatgttgagaacgtaggcgccgaggccaacgctga
atgcaccgccattagtccggttccaagagggcaactttgt
ctgcgggcggcccagtgcgcaacgcacagggcaaggttta
tgtgttgggcggttctgaccacatgcgagggcaacctccc
gtcgcctaccctggcaattgtaaaacgacggcaatgttcg
cgtattaatgataaagaggggggtaggaggtcaactcttc
aatgcttataacataggagtagagtagtgggtaaactacg
tctgaaccttctttatgcgaagacgccgagggcaatcggga
tgcatgtctgacaacttgtccaggaggaggtcaacgactc
cgtgtcatagaattccatccgccacgcggggtaatttgga
tcccgtcaaagtgccaacttgtgccgggggggctagcagct
acagcccgggaatatagacgcgtttggagtgcaaacatac
acgggaagatacgagttcgatttcaagagttcaaaacgtg
cccgataggactaataaggacgaaacgagggcgatcaatg
ttagtacaaacccgctcacccgaaaggagggcaaatacct
agcaaggttcagatatacagccagggggagacctataactc
gtccacgtgcgtatgtactaattgtggagagcaaatcatt
...
```

... pode ser encarado como um problema de alinhamento de sequências.

O problema da descoberta de motifs

Padrões sem mutação:



acgtacgt

Sequência Consenso

O problema da descoberta de motifs

Padrões com duas mutações

cctgatagacgctatctggctatcc**aggtacgt**aggtcctctgtgcgaatctatgcgtttccaaccat

agtactggtgtacatttgat**ccatacgt**acaccggcaacctgaaacaaacgctcagaaccagaagtgc

aa**acgttagt**gcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt

agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatctt**acgtccat**ataca

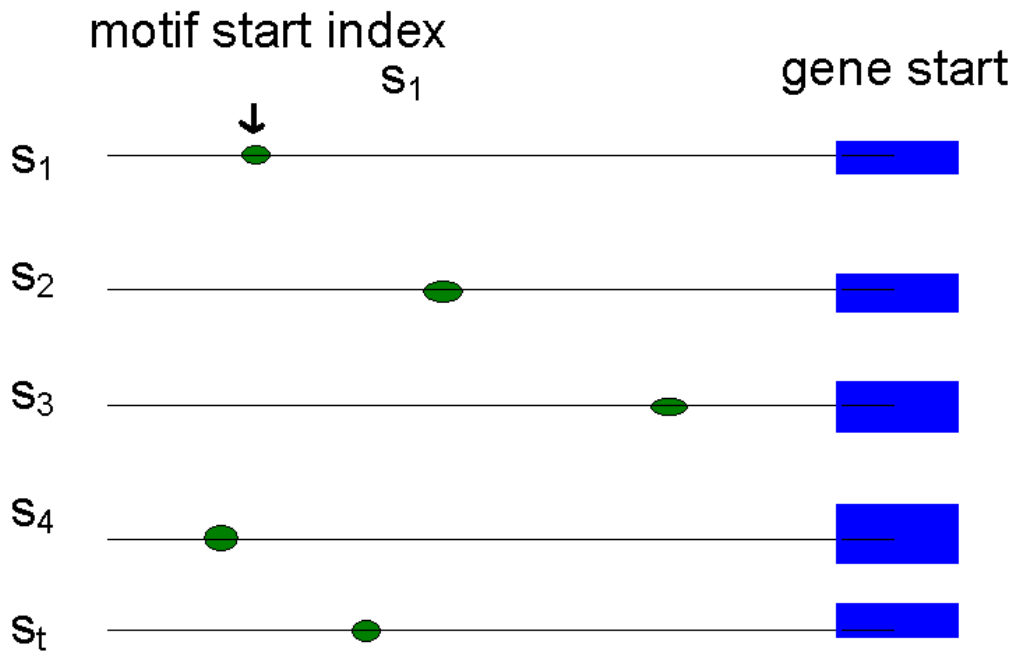
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgtta**ccgtacgg**c

Ainda conseguimos descobrir o padrão ?

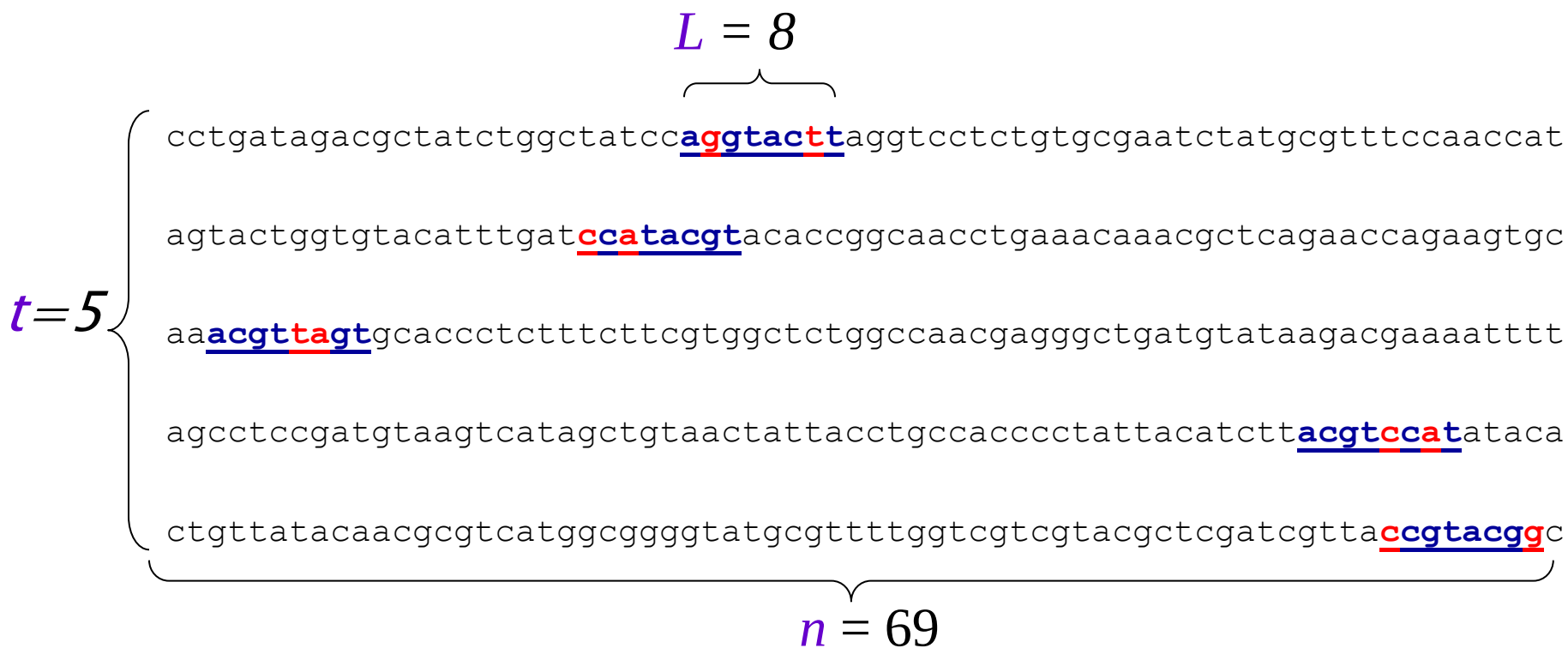
Qual é a sequência de consenso ?

Definição de motifs

Vamos começar por supor que sabemos onde o motif começa em cada sequência, ou seja, é conhecido o vetor $\boldsymbol{s} = (s_1,s_2,s_3,\ldots,s_t)$.



Definição de motifs



$$\mathbf{s}=\{s_1 = 25,\ s_2 = 20,\ s_3 = 2,\ s_4 = 55,\ s_5 = 59\}$$

Motifs: Perfis e Consensos

Alinhamento	a g g t a c t t c c a t a c g t a c g t t a g t a c g t c c a t c c g t a c g g	Alinhar os padrões pelas suas posições iniciais $\mathbf{s} = (s_1, s_2, \ldots, s_t)$
Matriz de ocorrências	A 3 0 1 0 3 1 1 0 C 2 4 0 0 1 4 0 0 G 0 1 4 0 0 0 3 1 T 0 0 0 5 1 0 1 4	Construir perfil (matriz) com frequências de cada nucleótido nas várias posições da sequência (colunas)
Consenso	A C G T A C G T	Nucleótido que em cada posição tem maior score em cada coluna.

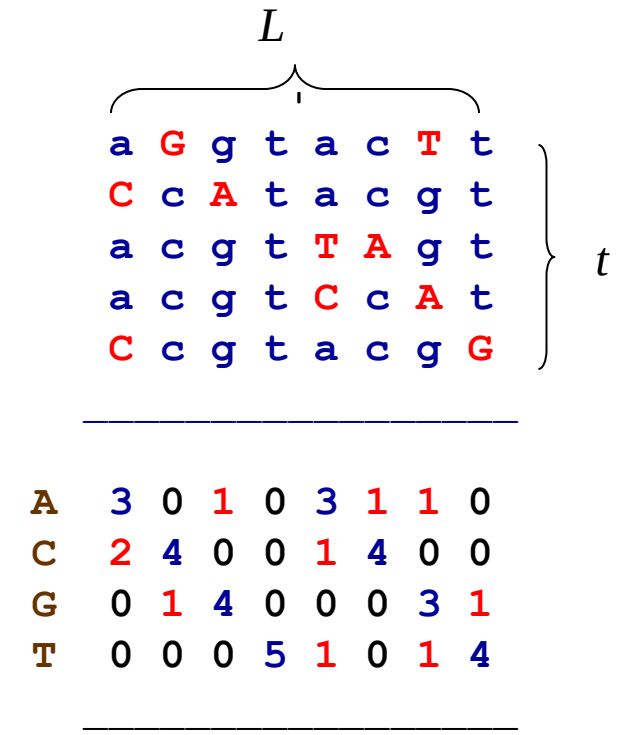
Função de avaliação (scoring)

Dado **s** = $(s_1, \ldots s_t)$:

$$score(s) = \sum_{i=1}^{L} \max_{k \in Symbols} \left(count(k,i) \right)$$

O score mínimo é 0, e o máximo é $t \times L$, obtido quando todas as sequências coincidem com o consenso.

Podem definir-se outras funções, como por exemplo uma semelhante à anterior, mas com características multiplicativas, ou log-odds.



Consenso: a c g t a c g t

Score = 3+4+4+5+3+4+3+4=30

Problema de descoberta de motifs

Se as posições iniciais **s**=(s_1, s_2,… s_t) são dadas, o problema resume-se a construir o perfil para conhecer o motif (consenso).

Mas… tipicamente as posições iniciais **s** não são conhecidas.

Como podemos alinhar os padrões e calcular o melhor perfil, ou seja, descobrir **s**?

Formulação do problema de optimização

Problema da descoberta de motifs: Dado um conjunto de sequências de DNA, descobrir um conjunto de subsequências de comprimento *L*, uma de cada sequência de DNA, que maximiza uma dada função de avaliação (Score)

- Entradas: ***t*** sequências de DNA (tamanho ***n)***; ***L***, o comprimento das subsequências

- Saída: Um vector de ***t*** posições iniciais (inteiros)
  s = (s_1, s_2, … s_t) que maximiza *Score*(s,*DNA*)

Uma solução de procura exaustiva

Calcular o score de cada possível vetor de posições iniciais **s.**

- O melhor score determina o respetivo perfil e o padrão de consenso.
- O objetivo é maximizar *Score*(**s**,***seqs***) variando as posições iniciais s_i, onde s_i = [0,.., n-L].

São avaliadas todas as combinações de posições:

(0,0,…,0,0) (0,0,…,0,1) … (n-L,…, n-L, n-L).

Algoritmo da procura exaustiva

ProcuraExaustivaMotifs(***seqs***, ***t***, ***n***, ***L***)

melhorScore ← 0

PARA cada **s** desde (0,0 . . . 0)
até (***n***-***L***, ***n***-***L***,. . ., ***n***-***L***)

SE (*score*(**s**) > ***melhorScore***)

melhorScore ← *score*(**s**)

melhorMotif ← s

retorna **melhorMotif**

Uma solução de procura exaustiva

Variando ($n - L + 1$) posições em cada uma das ***t*** sequências, estamos a considerar $(n - L + 1)^t$ possíveis soluções, i.e., vectores de posições iniciais.

Para cada vector de posições iniciais, a função de score realiza L operações, logo a complexidade é $L.(n - L + 1)^t = O(L \cdot n^t)$.

Mesmo para problemas de reduzida dimensão, como por exemplo, para 10 sequências de tamanho 20 e um motif de comprimento 6, serão necessárias aproximadamente $6 \times 20^{10} = 61.440.000.000.000$ operações.

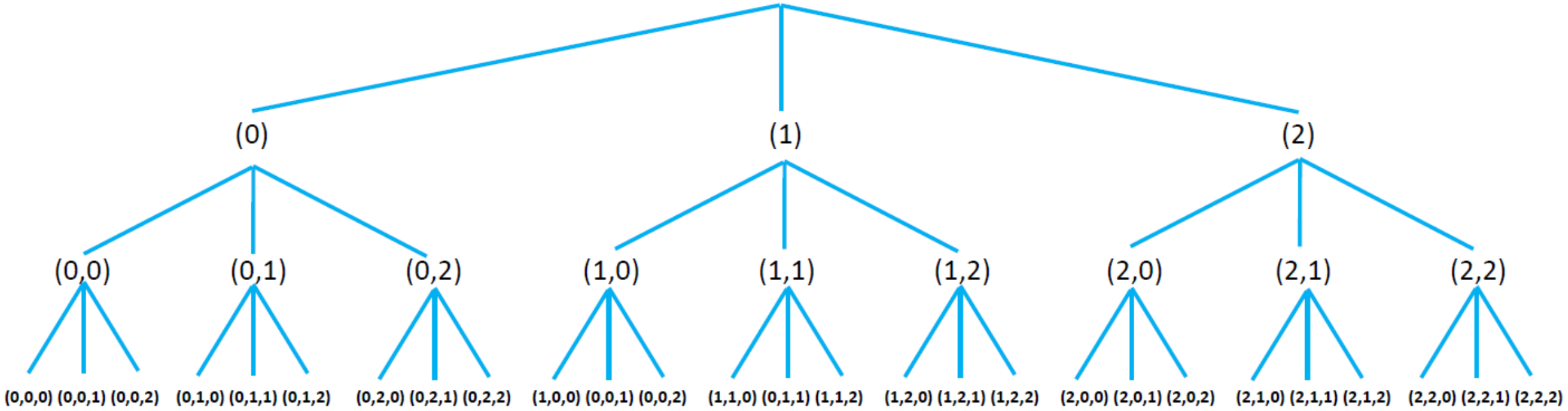
Re-estruturando a procura

- Como é que podemos re-implementar o processo de procura de modo mais eficiente?

- Necessitamos de um método para estruturar eficientemente a navegação entre os possíveis motifs!

Árvores de procura (Branch & Bound)

- Podemos agrupar as soluções $s=(s_1, s_2, \ldots, s_t)$ pelos seus prefixos numa árvore de procura.
- A título de exemplo, considere-se 3 sequências de comprimento 5 e um motif de comprimento 3, ou seja, **t = 3, n = 5 e *L* =3**.



Árvores de procura

- Os caminhos do nó inicial (raiz) para qualquer nó são muito curtos (coincide com o número t de sequências);
- Características das árvores de procura:
  - As soluções estão nas folhas
  - O pai de um nó é o prefixo da sua solução

- Como é que navegamos na árvore ?

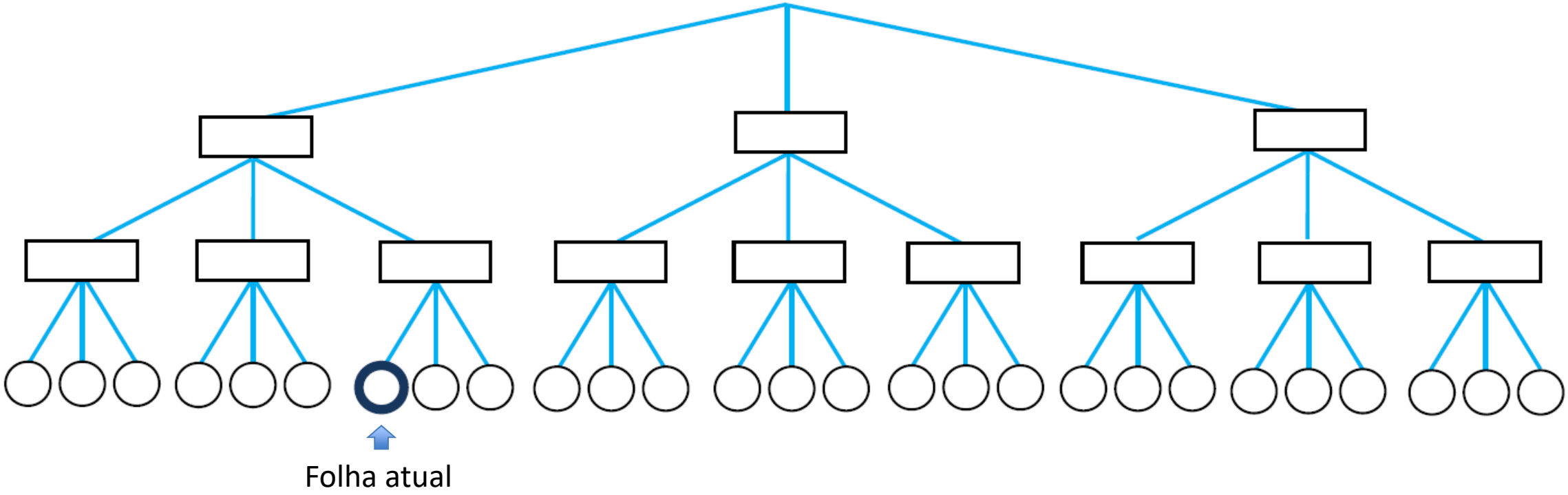
Navegação na árvore de procura

Quando a árvore está criada, necessitámos desenhar algoritmos para percorrer a árvore.

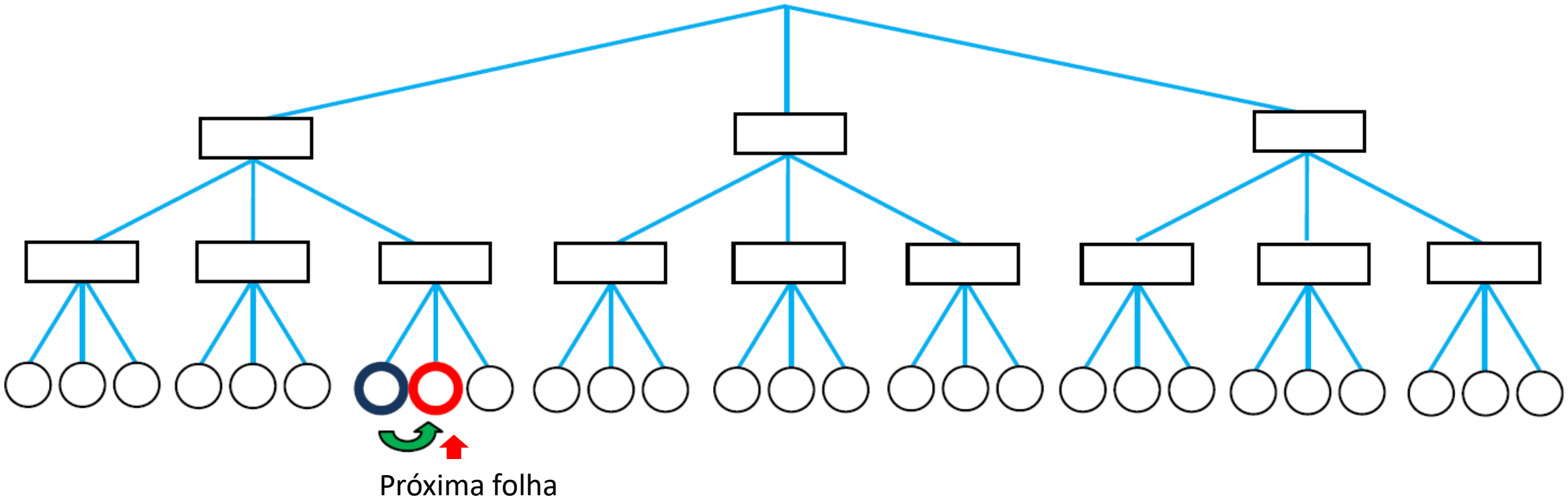
Quatro movimentos típicos:

- Ir para a próxima folha;
- Visitar todas as folhas;
- Visitar o próximo nó;
- Saltar à frente dos ramos de um dado nó (bypass) – visitar o próximo nó num dado nível.

Próxima folha: exemplo

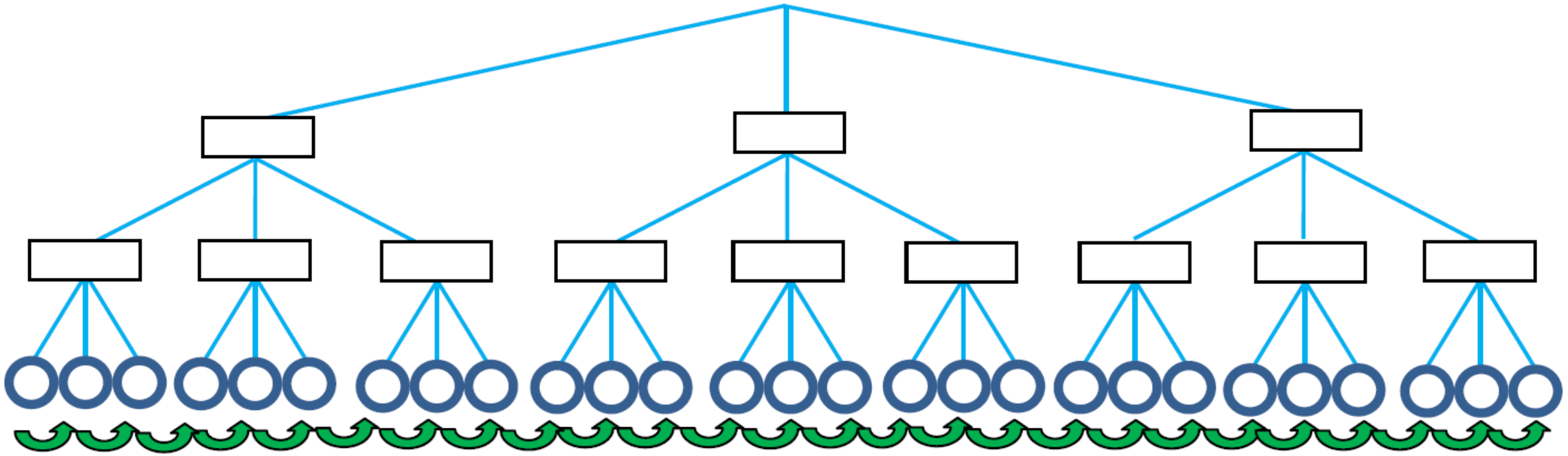


Próxima folha: exemplo



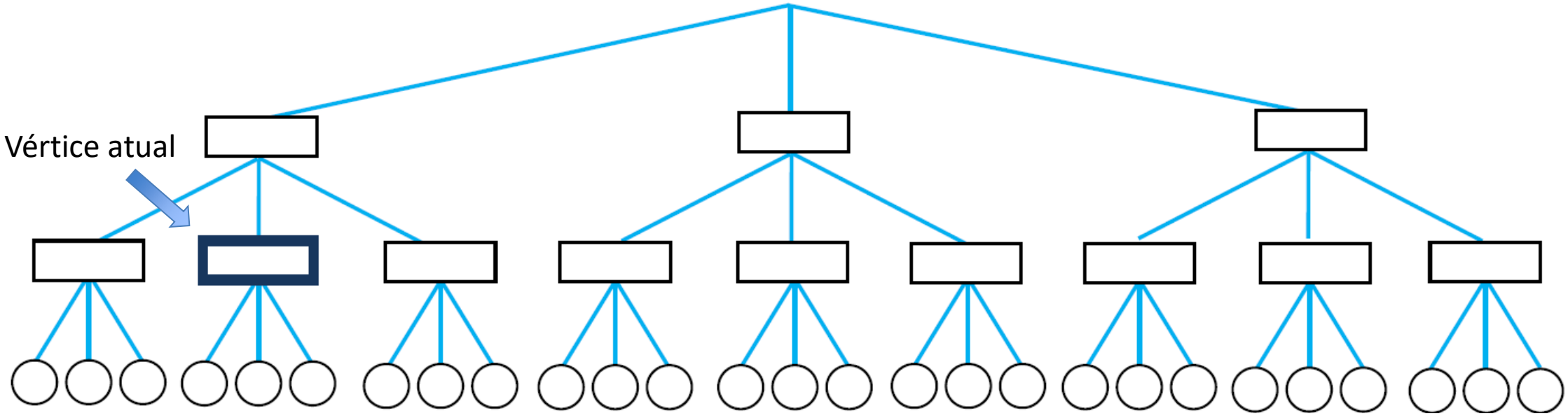
Semelhante à próxima solução na procura exaustiva.

Visitar todas as folhas: exemplo

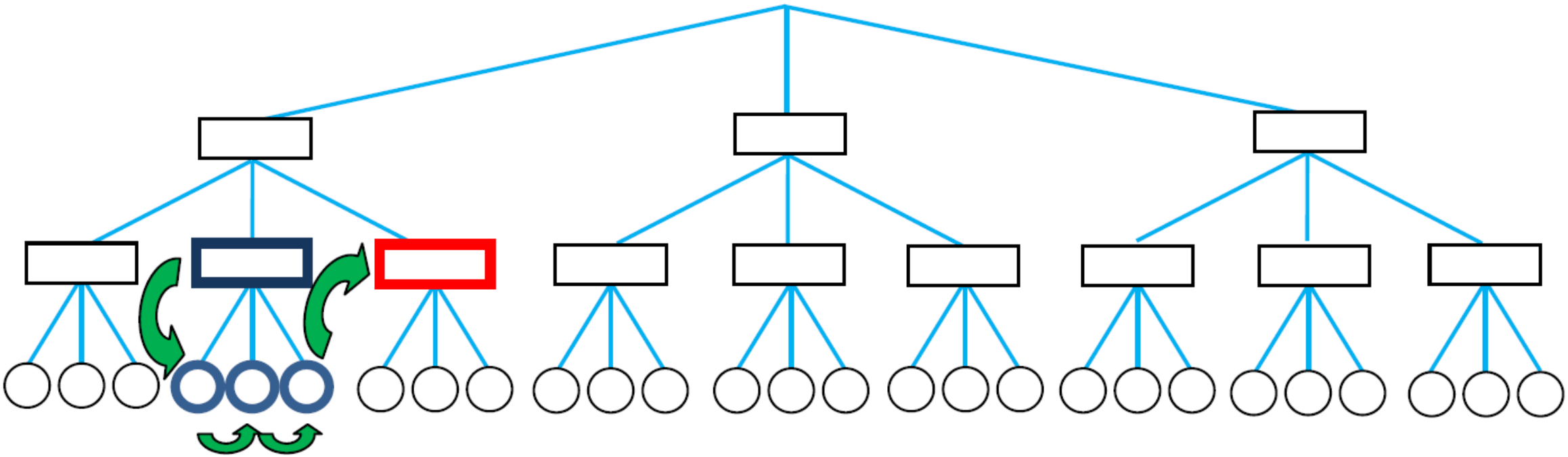


Percorrer todas as folhas é equivalente a realizar uma procura exaustiva.

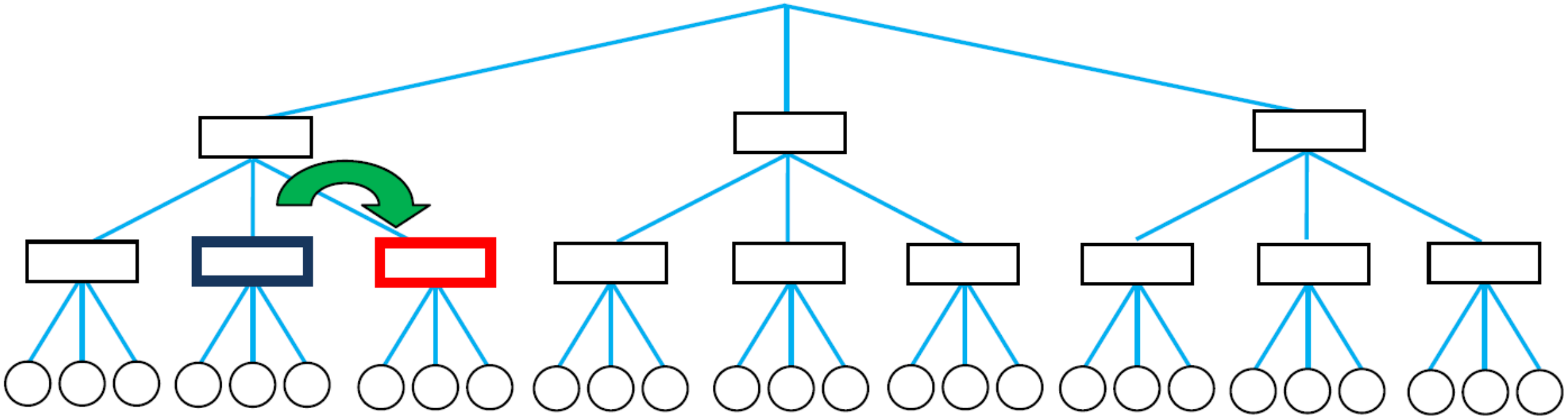
Próximo vértice: exemplo



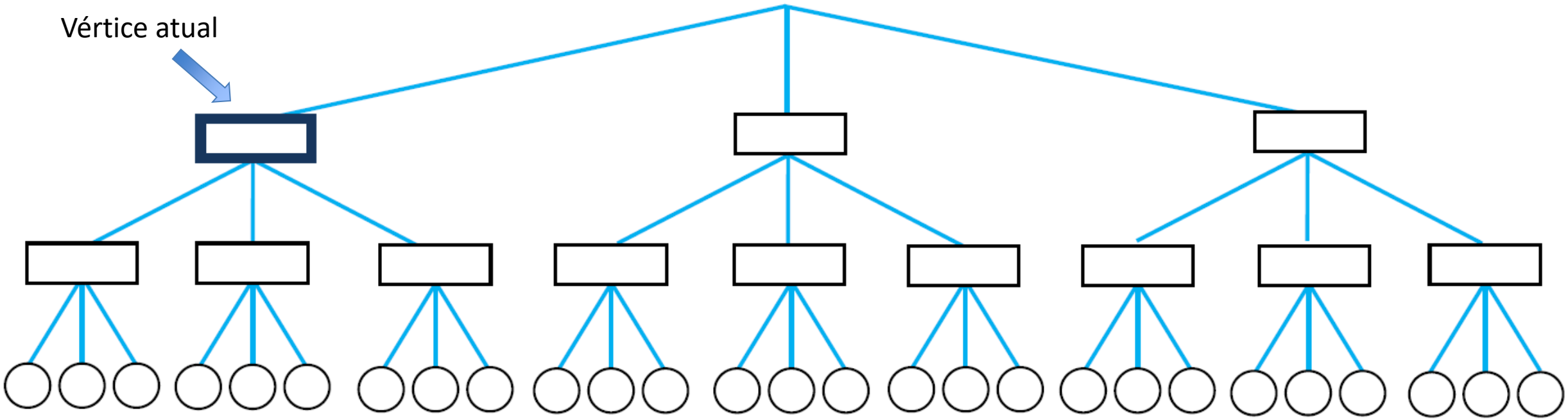
Próximo vértice: exemplo



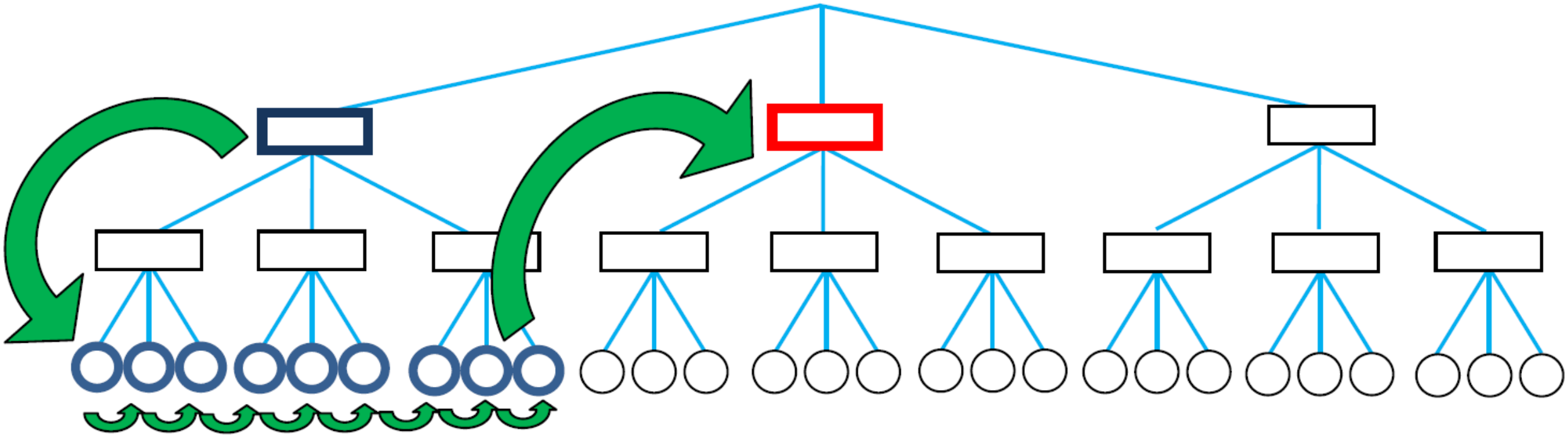
Bypass - exemplo



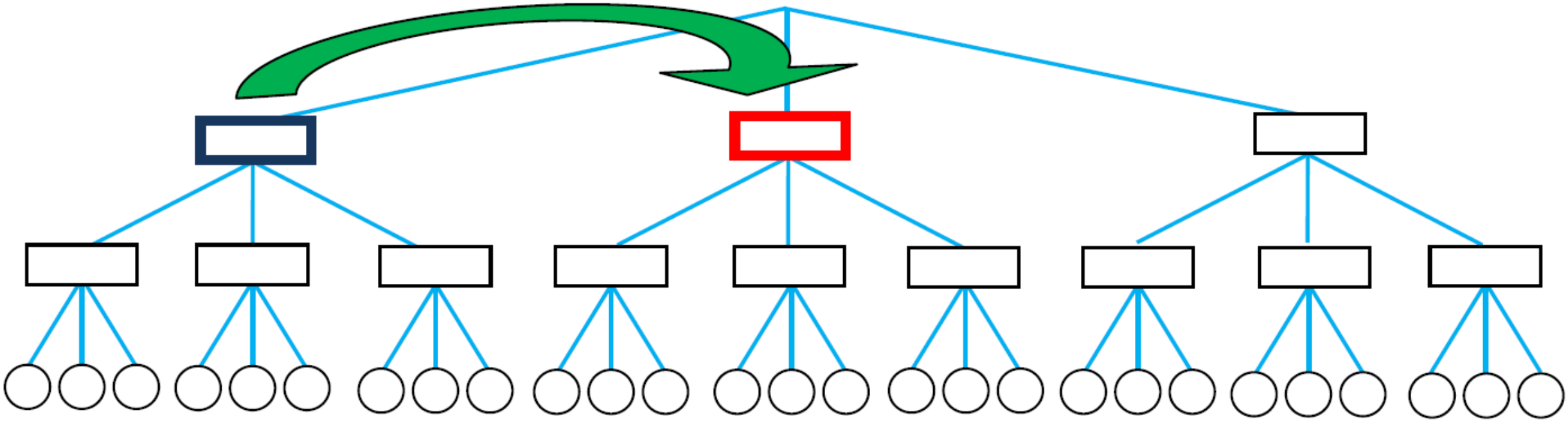
Próximo vértice: exemplo



Próximo vértice: exemplo



Bypass - exemplo



Pretendemos, quando as folhas de um prefixo não contêm a melhor solução, poder saltar para outro prefixo no mesmo nível.

Podemos melhorar a procura exaustiva?

Vamos supor que estamos a realizar a travessia da árvore estando no prefixo: (s_1, s_2, …,s_i), ou seja, no nível i considerando "fixas" as posições iniciais em i sequências.

Numa visão optimista, se todas as posições subsequentes fazem crescer o score de forma máxima – contributo será de $(t - i) * L$ para somar ao score atual - **score(s)**.

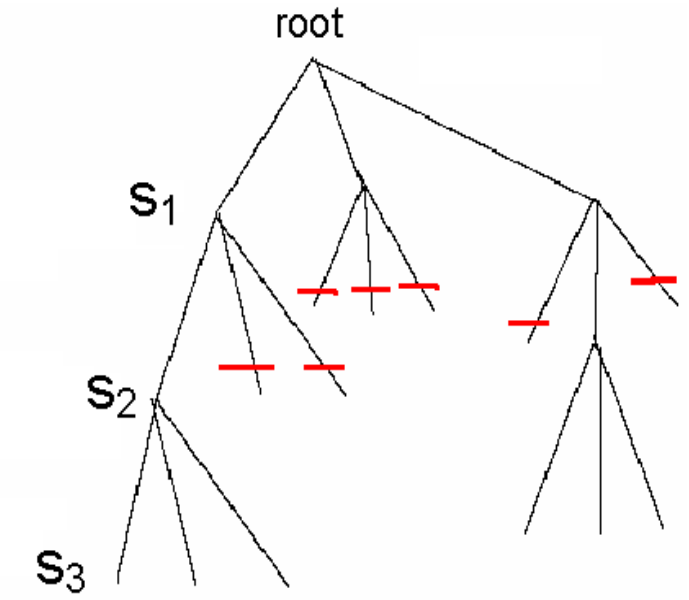
Se ainda assim **score(s)** + $(t - i) * L$ < **MelhorScore** (melhor score obtido até ao momento) não faz sentido procurar nos descendentes da sua sub-árvore.

Usar **bypass** e continuar a procura a partir do próximo vértice no mesmo nível **(**este é o passo de **Bound)**.

Algoritmo Branch and Bound

- Dado que cada nível da árvore vai mais em profundidade na procura, descartar um prefixo equivale a descartar todos os ramos abaixo.

- "Poupámos" a pesquisa de $(\boldsymbol{N} - L + 1)^{t-i}$ árvores.



Nota:

Na implementação em Python, usar as funções **ProximoVertice** e **ByPass**

Algoritmos exatos

Os algoritmos de procura exaustiva e branch-and-bound são ambos **algoritmos exatos**.

O que significa que descobrem sempre a solução ótima, embora possam tornar-se demasiado ineficientes em problemas práticos, sobretudo quando o espaço de procura é muito grande.

Muitos algoritmos atuais sacrificam a solução ótima para ganhar em eficiência. Às vezes o ótimo é inimigo do bom.

Algoritmo heurístico (tipo Consensus)

1. Considerando apenas as duas primeiras sequências, escolher as posições iniciais s_1 e s_2 que dão um melhor score (melhor contribuição parcial para o score, i.e. considerando que existem apenas estas duas sequências).

2. Para cada uma das sequências seguintes (i=3, …,t) , de forma iterativa, escolher a melhor posição inicial na sequência i, de forma a maximizar o score, considerando as posições anteriores (s_1, s_2, …,s_{i-1}) fixas.

Algoritmo heurístico (tipo Consensus)

- Algoritmo é fortemente dependente das posições iniciais;

- Resultados podem ser melhorados se se considerarem várias ordens de apresentação das sequências distintas (por ex. aleatórias);

- CONSENSUS – mantém um grande nº de soluções parciais em cada iteração (na ordem dos milhares) selecionando o melhor resultado no final.

Algoritmos estocásticos

- Métodos heurísticos usados em problemas complexos, onde a solução óptima não se pode atingir em tempo útil.
- Em algumas etapas tomam decisões com uma base **aleatória**.

Métodos estocásticos de extração de motifs

- Algoritmos heurísticos simples;
- **Gibbs sampling**
- Algoritmo **E-M** (Expectativa- Maximização)
  - Programas MEME/ MAST (UCSD)
- **Algoritmos evolucionários**

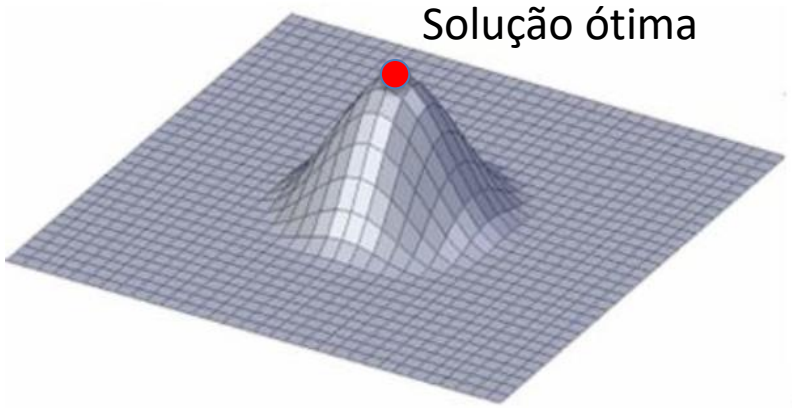
Um algoritmo heurístico estocástico

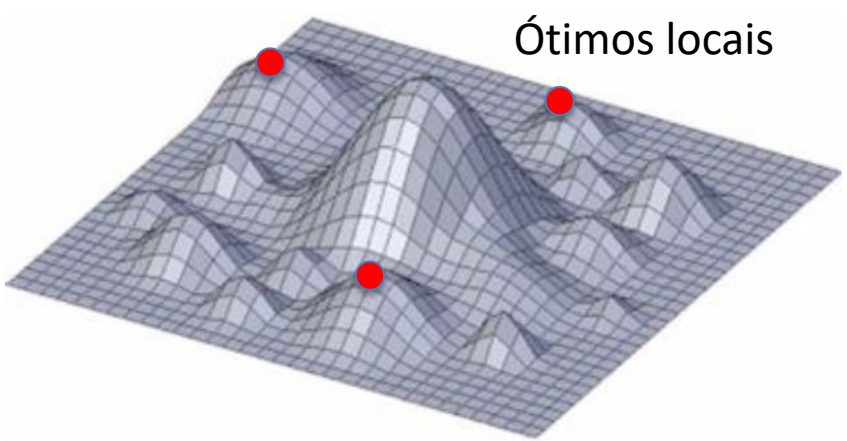
Usar segmentos mais prováveis para ajustar as posições iniciais até atingir o melhor perfil – este será o motif.

1) Selecionar posições iniciais em cada sequência de forma aleatória
2) Criar um perfil ***P*** a partir das posições geradas em 1.
3) Descobrir o segmento ***α*** mais provável em cada sequência usando ***P***. Mudar a posição inicial nessa sequência para a posição inicial do segmento **a**.
4) Calcular um novo perfil ***P*** baseado nas posições calculadas em 3. Repetir os passos 3 e 4 enquanto for possível aumentar o score.

Análise do algoritmo heurístico estocástico

- Dado que as posições iniciais são geradas aleatoriamente, as probabilidades de estarem próximas do motif ótimo são bastante reduzidas em problemas de dimensão real.
- Dado o “mau” ponto de partida é pouco provável que se atinja uma solução ótima.
- Na prática, este algoritmo tem que ser corrido muitas vezes com diferentes pontos iniciais para melhorar as hipóteses de obter um bom resultado.





Gibbs Sampling

O algoritmo anterior pode ser melhorado introduzindo o método de Gibbs Sampling, um processo iterativo que vai substituindo um segmento em cada iteração.

O método de Gibbs Sampling é mais lento, escolhendo novos segmentos de forma aleatória.

Mas, desta forma, aumenta as possibilidades de se convergir para uma solução correta.

Gibbs Sampling: passos

1) Escolher posições iniciais de forma aleatória $\boldsymbol{s} = (s_1,...,s_t)$ e formar os segmentos respectivos.

2) Escolher aleatoriamente uma sequência *i*

3) Criar perfil *P* das outras sequências a partir de *s*

4) Para cada posição *p* na sequência *i*, calcular a probabilidade do segmento iniciado em p com tamanho *L*, ser gerado por *P*.

5) Escolher *p* de modo de forma estocástica, de acordo com as probabilidades calculadas em 4).

6) Repetir passos 2) a 5) enquanto for possível melhorar

Exemplo: Gibbs Sampling

Input:

t = 5 sequências com L = 8

1. GTAAACAATATTTATAGC
2. AAAATTTACCTCGCAAGG
3. CCGTACTGTCAAGCGTGG
4. TGAGTAAACGACGTCCCA
5. TACTTAACACCCTGTCAA

Exemplo: Gibbs Sampling

1) Escolher posições iniciais aleatórias

S_1=7 GTAAACAATATTTATAGC

S_2=11 AAAATTTACCTTAGAAGG

S_3=9 CCGTACTGTCAAGCGTGG

S_4=4 TGAGTAAACGACGTCCCA

S_5=1 TACTTAACACCCTGTCAA

2) Escolha aleatória de uma das sequências:

S_2: AAAATTTACCTTAGAAGG

Exemplo: Gibbs Sampling

3) Criar perfil:

Sequências

1	A	A	T	A	T	T	T	A
3	T	C	A	A	G	C	G	T
4	G	T	A	A	A	C	G	A
5	T	A	C	T	T	A	A	C

PWM

A	1/4	2/4	2/4	3/4	1/4	1/4	1/4	2/4
C	0	1/4	1/4	0	0	2/4	0	1/4
T	2/4	1/4	1/4	1/4	2/4	1/4	1/4	1/4
G	1/4	0	0	0	1/4	0	3/4	0

Consenso **T A A A T C G A**

Exemplo: Gibbs Sampling

4) Calcular *prob*(**a**|**P**):

S	Segmento a	Prob(a|P)
1	AAAATTTACCTTAGAAGG	.000732
2	AAAATTTACCTTAGAAGG	.000122
3	AAAATTTACCTTAGAAGG	0
4	AAAATTTACCTTAGAAGG	0
5	AAAATTTACCTTAGAAGG	0
6	AAAATTTACCTTAGAAGG	0
7	AAAATTTACCTTAGAAGG	0
8	AAAATTTACCTTAGAAGG	.000183
9	AAAATTTACCTTAGAAGG	0
10	AAAATTTACCTTAGAAGG	0
11	AAAATTTACCTTAGAAGG	0

Exemplo: Gibbs Sampling

5) Calcular probabilidades:

P(pos = 1) = 0.00732 / (0.000732+0.000122+0.000183) = 0.706

P(pos = 2) = 0.000122/ (0.000732+0.000122+0.000183) = 0.118

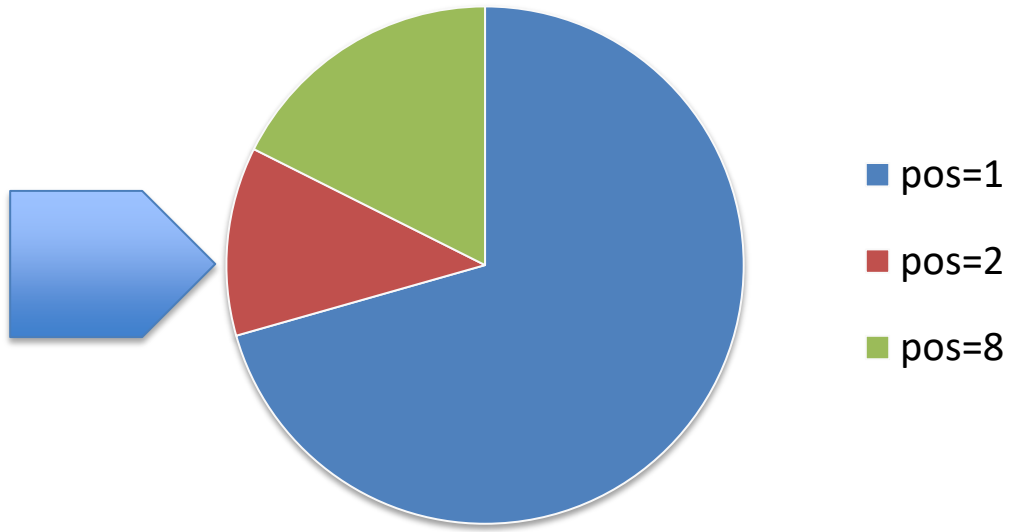
P(pos = 8) = 0.000183 / (0.000732+0.000122+0.000183) = 0.176

P(pos = *) = 0 (restantes posições iniciais)

Nota : a soma destas probabilidades = 1

Aplicar uma Roulette wheel:

A probabilidade de escolher uma determinada posição é proporcional ao seu score.



Exemplo: Gibbs Sampling

Escolhemos uma substring segundo estas probabilidades. Vamos supor que é a posição 1.

S_1=7	GTAAACAATATTTATAGC
S_2=1	AAAATTTACCTCGCAAGG
S_3=9	CCGTACTGTCAAGCGTGG
S_4=5	TGAGTAATCGACGTCCCA
S_5=1	TACTTCACACCCTGTCAA

6) O processo é iterado até atingir um critério de terminação (e.g. número fixo de iterações, nº de iterações sem melhorias)

Pseudo-contagens

Dadas as características dos algoritmos estocásticos devem ser usadas pseudo-contagens para evitar probabilidades nulas.

Neste caso, ao gerar-se a matriz de ocorrências deve somar-se sempre uma unidade (ou um valor mais pequeno) a todas as contagens.

Matriz de ocorrências

A	4	0	1	0	3	1	1	0
C	2	5	0	0	2	5	0	0
G	0	1	5	0	0	0	4	2
T	0	0	0	6	1	0	1	4

Pseudo-contagem

5	1	2	1	4	2	2	1
3	6	1	1	3	6	1	1
1	2	6	1	1	1	5	3
1	1	1	7	2	1	2	5

PWM

0.5	0.1	0.2	0.1	0.4	0.2	0.2	0.1
0.3	0.6	0.1	0.1	0.3	0.6	0.1	0.1
0.1	0.2	0.6	0.1	0.1	0.1	0.5	0.3
0.1	0.1	0.1	0.7	0.2	0.1	0.2	0.5

Problemas do Gibbs Sampling

- Gibbs Sampling converge frequentemente para soluções óptimas locais e não globais.

- Uma das soluções para o problema anterior é correr o processo um nº elevado de vezes.

- Gibbs Sampling pode funcionar mal quando há regiões com distribuições distintas de nucleótidos.

Implementação em python

- Classe **MyMotifs**: implementa motifs biológicos, PWMs e processos de procura de motifs conhecidos em sequências

- Implementada nas aulas de Algoritmos para análise de sequências biológicas no 1º semestre

Implementação em python

- Classe **MotifFinding**: implementa algoritmos de procura de motifs conservados, dados conjuntos de sequências, i.e. algoritmos de otimização para descoberta/inferência de motifs
  - São dados métodos para calcular scores aditivos e multiplicativos
  - São dados métodos que permitem a navegação na árvore de procura (próxima solução, próximo vértice e bypass)
  - É dada a implementação do algoritmo de procura exaustiva e branch & bound

Implementação em python

Exercícios (completar classe anterior)

- Implementar algoritmo heurístico tipo consensus
- Implementar algoritmo heurístico estocástico
- Implementar algoritmo de Gibbs sampling (função que implementa roulette wheel é dada)
- Adaptar os dois anteriores para funcionar com pseudo-contagens. (**A entregar na próxima semana**)

Sugestões de leitura

P. D'Haeseleer, "What are DNA sequence motifs?“, *Nature Biotechnology*, vol. 24, n.º 4, pp. 423-425, 2006.

P. D’haeseleer, “How does DNA sequence motif discovery work?”, *Nature Biotechnology*, vol. 24, n.º 8, pp. 959–961, 2006.