

# Analisi di algoritmi per il Motif Finding

Tommaso Papini

[tommaso.papini1@stud.unifi.it](mailto:tommaso.papini1@stud.unifi.it)

Gabriele Bani

[gabriele.bani@stud.unifi.it](mailto:gabriele.bani@stud.unifi.it)



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

11 Dicembre 2015

# Un po' di background

DNA:

- sequenza di **nucleotidi**
- 4 tipi di nucleotide: A, T, C, G
- ***l-mer***: sottosequenza di DNA di lunghezza *l*

## Motifs

In biologia può essere necessario ricavare certe sequenze di DNA “nascoste”

- ✓ pattern di nucleotidi ripetuti (*l-mer*)
- ✓ utili a capire determinati comportamenti biologici
  - sequenze di attivazione di geni specifici

# Il problema del Motif Finding

Il problema del **Motif Finding** consiste nel ricavare un set di  $t$  **l-mer** da un insieme di  $t$  sequenze di DNA.

## Input

- *DNA*: matrice di nucleotidi  $t \times n$ 
  - $t$  sequenze di DNA
  - ognuna di lunghezza  $n$
- $l$ : lunghezza del motif cercato

## Output

- ✓  $s = (s_1, s_2, \dots, s_t)$ : lista di  $t$  posizioni iniziali di l-mer il più simili tra loro

## Un primo esempio

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTGATA  
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA  
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC  
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG  
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC  
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC  
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC

## Un primo esempio

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTTCGATA  
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA  
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC  
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG  
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC  
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC  
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC

# Mutazioni random

CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTTTTCGATA  
TTTGAGGGTGCCCAATAAaggGCAACTCCAAAGCGGACAAA  
GGATGgAtCTGATGCCGTTTTGACGACCTAAATCAACGGCC  
AAGGAaGCAACcCCAGGAGCGCCTTTGCTGGTTCTACCTG  
AATTTTCTAAAAAGATTATAATGTCGGTCCtTGgAACTTC  
CTGCTGTACAACCTGAGATCATGCTGCATGCcAtTTTCAAC  
TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC

Come trovare l'l-mer più simile tra tutti?

# Allineamento

CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTTTCGATA  
TTTGAGGGTGCCCAATAAaggGCAACTCCAAAGCGGACAAA  
GGATGgAtcTGATGCCGTTTGACGACCTAAATCAACGGCC  
AAGGAaGCAACcCCAGGAGCGCCTTTGCTGGTTCTACCTG  
AATTTTCTAAAAAGATTATAATGTCGGTCCtTGgAACTTC  
CTGCTGTACAACTGAGATCATGCTGCATGCcAtTTTCAAC  
TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC

# Profilo e Consenso

Allineamento		A	T	C	C	A	G	C	T
		G	G	G	C	A	A	C	T
		A	T	G	G	A	T	C	T
		A	A	G	C	A	A	C	C
		T	T	G	G	A	A	C	T
		A	T	G	C	C	A	T	T
		A	T	G	G	C	A	C	T
		<hr/>							
Profilo	<b>A</b>	5	1	0	0	5	5	0	0
	<b>T</b>	1	5	0	0	0	1	1	6
	<b>G</b>	1	1	6	3	0	1	0	0
	<b>C</b>	0	0	1	4	2	0	6	1
Consenso		<hr/>							
		A	T	G	C	A	A	C	T



Come definire la “bontà” di un set di l-mer?

## Funzione score

Si definisce una funzione **score** sul vettore  $s = (s_1, s_2, \dots, s_t)$  di posizioni iniziali:

$$\text{Score}(s, \text{DNA}) = \sum_{j=1}^l M_{P(s)}(j)$$

dove

- ✓  $P(s)$ : matrice profilo su  $s$
- ✓  $M_{P(s)}(j)$ : elemento massimo nella colonna  $j$ -esima di  $P(s)$

Si cerca il set di posizioni iniziali  $s$  che **massimizzi**  $\text{Score}(s, \text{DNA})$ !

## Score: l'esempio di prima

<b>Allineamento</b>		A	T	C	C	A	G	C	T
		G	G	G	C	A	A	C	T
		A	T	G	G	A	T	C	T
		A	A	G	C	A	A	C	C
		T	T	G	G	A	A	C	T
		A	T	G	C	C	A	T	T
		A	T	G	G	C	A	C	T
<b>Profilo</b>	<b>A</b>	5	1	0	0	5	5	0	0
	<b>T</b>	1	5	0	0	0	1	1	6
	<b>G</b>	1	1	6	3	0	1	0	0
	<b>C</b>	0	0	1	4	2	0	6	1
<b>Consenso</b>		A	T	G	C	A	A	C	T

$$\text{Score}(s, \text{DNA}) = 5 + 5 + 6 + 4 + 5 + 5 + 6 + 6 = 42$$

Quanto può valere lo score?

$$\text{Score}(s, \text{DNA}) = \begin{cases} l \cdot t, & \text{nel caso migliore} \\ \frac{l \cdot t}{4}, & \text{nel caso peggiore} \end{cases}$$

- ✓  $lt$  corrisponde al caso in cui tutti gli  $l$ -mer sono identici
- ✓  $\frac{lt}{4}$  corrisponde al caso in cui gli  $l$ -mer siano diversi in tutte le posizioni

## Forza bruta

In informatica il metodo “forza bruta” (o ricerca esaustiva della soluzione) è un algoritmo di risoluzione di un problema dato che consiste nel verificare tutte le soluzioni teoricamente possibili fino a che si trova quella effettivamente corretta.

# Simple motif search

## L'idea

Esamina tutte le possibili combinazioni delle posizioni di partenza  $s$  e prendi quella con maggior *Score*.

- Posizioni iniziali:  $s = (1, \dots, 1)$
- Posizioni finali:  $s = ((n - l + 1), \dots, (n - l + 1))$

Si utilizza il metodo *NextElement* per passare da un elemento al successivo in ordine alfabetico.

# Simple motif search

## Pseudocode

```
1: procedure SimpleMotifSearch(DNA, t, n, l)
2:    $s \leftarrow (1, 1, \dots, 1)$ 
3:   bestScore  $\leftarrow$  Score(s, DNA)
4:   while true do
5:      $s \leftarrow \text{NextElement}(s, t, n - l + 1)$ 
6:     if Score(s, DNA) > bestScore then
7:       bestScore  $\leftarrow$  Score(s, DNA)
8:       bestMotif  $\leftarrow$  s
9:     end if
10:    if  $s = (1, 1, \dots, 1)$  then
11:      return bestMotif
12:    end if
13:  end while
14: end procedure
```

# Simple motif search

## Complessità

Quante **iterazioni** fa l'algoritmo?

- ✓ vengono esaminate tutte le possibili combinazioni
  - $(n - l + 1)$  possibili scelte per ogni posizione iniziale
  - $t$  posizioni iniziali (una per ogni sequenza)
- ✓  $(n - l + 1)^t$  possibili combinazioni di posizioni iniziali

Quanti **passi** per calcolare *Score*?

- ✓  $tl$  per calcolare la matrice Profilo
- ✓  $4l$  per calcolare *Score*

Costo

$$\mathcal{O}(tln^t)$$

## L'idea

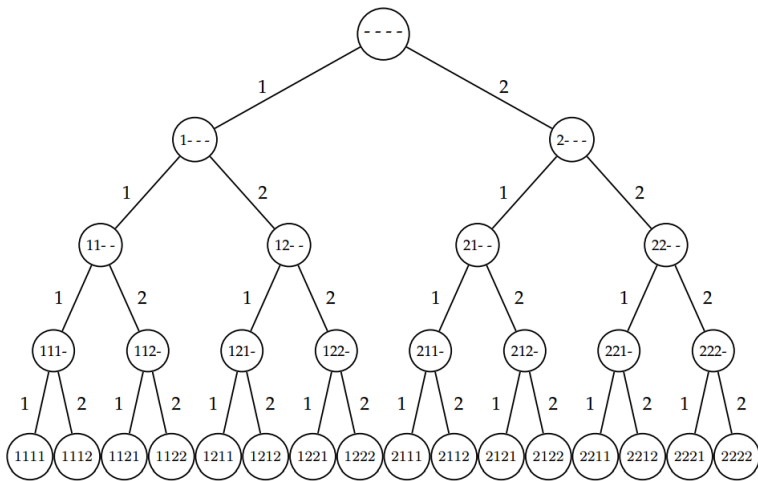
Ottimizza *Simple Motif Search* evitando di analizzare sequenze non ottimali

- ✓ enumerazione delle sequenze tramite **alberi**
- ✓ funzione di *Score ottimistico*
  - *Score parziale* su un nodo interno
  - *Score ideale* per le restanti posizioni



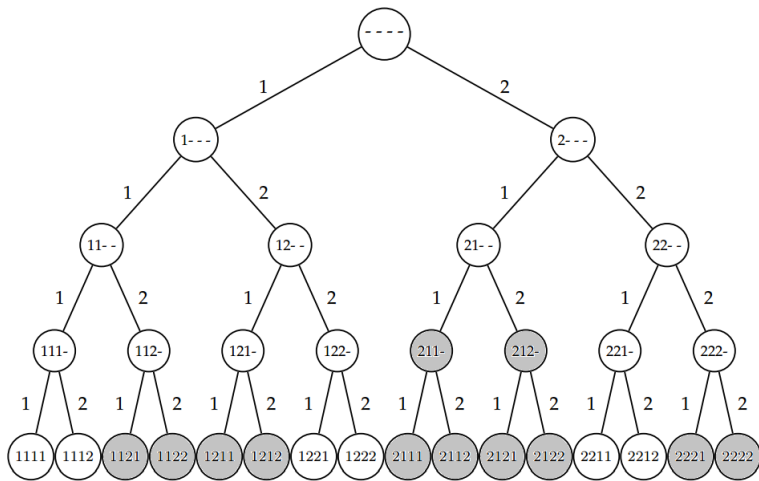
# Branch & bound

Un esempio



# Branch & bound

Un esempio



# Branch & bound

Si esegue una visita in profondità dell'albero

- ✓ calcola lo *Score ottimistico* per ogni nodo
- ✓ scarta i sottoalberi con *Score ottimistico* sub-ottimo

## Score ottimistico

$$\text{Score ottimistico} = \text{Score}(s, i, \text{DNA}) + (t - i) \cdot l$$

- ✓  $\text{Score}(s, i, \text{DNA})$ : *Score parziale* relativo alle prime  $i$  sequenze di DNA
  - ✓  $(t - i) \cdot l$ : *Score parziale* delle restanti posizioni (supponendole identiche)
- 
- ✓ *NextVertex* per passare al prossimo vertice (DFS)
  - ✓ *Skip* per passare al prossimo vertice saltando il sottoalbero attuale

# Branch & bound

## Pseudocode

```
1: procedure BranchAndBoundMotifSearch(DNA, t, n, l)
2:   s  $\leftarrow$  (1, 1, ..., 1)
3:   bestScore  $\leftarrow$  0
4:   i  $\leftarrow$  1
5:   while i > 0 do
6:     if i < t then
7:       optimisticScore  $\leftarrow$  Score(s, i, DNA) + (t - i)l
8:       if optimisticScore < bestScore then
9:         (s, i)  $\leftarrow$  Skip(s, i, (n - l + 1))
10:      else
11:        (s, i)  $\leftarrow$  NextVertex(s, i, t, (n - l + 1))
12:      end if
13:    else
14:      if Score(s, DNA) > bestScore then
15:        bestScore  $\leftarrow$  Score(s, DNA)
16:        bestMotif  $\leftarrow$  s
17:      end if
18:      (s, i)  $\leftarrow$  NextVertex(s, i, t, (n - l + 1))
19:    end if
20:  end while
21:  return bestMotif
22: end procedure
```

# Branch & bound

## Complessità

Quante **iterazioni**?

- ✓ una per ogni nodo interno/foglia (caso pessimo)
  - $N = \frac{(n-l+1)^t - 1}{(n-l+1) - 1}$  nodi interni
  - $L = (n-l+1)^t$  foglie
- ✓  $N + L$  passi totali

Quanto **passi** per calcolare *Score*?

- ✓ come prima!

Costo

$$\mathcal{O}(t \ln^t)$$

Come prima!

- ✓ più costoso nel caso pessimo
- ✓ conveniente se esegue tanti *Skip*

# Algoritmi greedy





# Conclusioni





*Fine.*



*Domande? Grazie!*

*Fine.*



*Domande? Grazie!*

*Fine.*



*Domande? Grazie!*