

---

# Universidad del Caribe

## Ingeniería en Datos e Inteligencia Organizacional

---



Tarea: Búsqueda de motivos I / Técnicas Algorítmicas

Dr. David Israel Flores Granados

PDF realizado en:



Victoria Andrea Garza Romero  
Alexis Iván Prudencio García  
Joan de Jesús Méndez Pool  
19/10/2018

## 0.1. Descripción de la tarea:

Realice un algoritmo y su respectivo programa de fuerza bruta que resuelva el problema de Búsqueda de Motivos para las Secuencias de ADN y patrón de la página 96 de capítulo 4 del libro "Introduction to Bioinformatics".

```

                CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTT...
      TTTGAGGGTGCCCAATAAaggGCAACTCCAAAGCGGACAAA
                GGATGgAtCTGATGCCGTTTGACGACCTA...
                AAGGAaGCAACcCCAGGAGCGCCTTTGCTGG...
AATTTTCTAAAAAGATTATAATGTCGGTCcTGgAACTTC
CTGCTGTACAACCTGAGATCATGCTGCATGcCAtTTTCAAC
      TACATGATCTTTTGATGgCACTTGGATGAGGGAATGATGC
  
```

(a) Superposition of the seven highlighted 8-mers from figure 4.2 (d).

Alignment		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
Profile	A	5	1	0	0	5	5	0	0
	T	1	5	0	0	0	1	1	6
	G	1	1	6	3	0	1	0	0
	C	0	0	1	4	2	0	6	1
Consensus		A	T	G	C	A	A	C	T

(b) The alignment matrix, profile matrix and consensus string formed from the 8-mers starting at positions  $s = (8, 19, 3, 5, 31, 27, 15)$  in figure 4.2 (d).

**Figure 4.3** From DNA sample, to alignment matrix, to profile, and, finally, to consensus string. If  $s = (8, 19, 3, 5, 31, 27, 15)$  is an array of starting positions for 8-mers in figure 4.2 (d), then  $Score(s) = 5 + 5 + 6 + 4 + 5 + 5 + 6 + 6 = 42$ .

# Antecedentes "The Motif Finding Problem"

Encontrar los motivos en varias muestras para generar secuencias, esto forma parte como uno de los problemas importantes en la computación aplicado en la biología, especialmente en la identificación de motivos reguladores en secuencias de ADN de determinado número de muestras. Esta proviene de un experimento que realizaron unos biólogos con unas moscas de fruta, las cuales infectaron con bacterias y luego las licuaron para tener este array de secuencias de ADN que analizar y detectar patrones clave.

Estas cadenas de secuencia que se intentan buscar en este problema se llaman sitios de unión NF-kB, el cual es el factor de activación de los genes de inmunidad que los biólogos intentan determinar. La mayoría de los algoritmos anteriores diseñados para encontrar motivos utilizan un conjunto de secuencias de genes corregidos para identificar motivos estadísticamente representados en exceso. Todo basado en un enfoque combinatorio (en strings de las cadenas y su análisis probabilístico).

Estas cadenas de secuencia que se intentan buscar en este problema se llaman sitios de unión NF-kB, el cual es el factor de activación de los genes de inmunidad que los biólogos intentan determinar.

## Propuesta

Nosotros proponemos el siguiente programa realizado en Python para facilidad de uso y manejo de los arrays, sus iteraciones, y la ayuda de herramientas de las librerías numpy y sys con funciones matemáticas que simplificaron las operaciones dentro de la lógica que exponemos en nuestro código para lograr obtener como salida el patrón de secuencia que más se repite en toda la cadena de ADN que se lee de un archivo dado con las muestras de ADN. El algoritmo debe ser capaz de encontrar el conjunto de secuencia que se repita más, el problema es que no se sabe en que parte de la cadena de ADN se encuentran o como es ni el largo que compone al patrón, por lo que se hace uso de probar todas las combinaciones posibles con el enfoque y aplicación del método BruteForce que ya hemos trabajado anteriormente con los algoritmos PDP. Por obvias razones se está delimitando de entrada el tamaño de la secuencia, y no se trabajará con datos reales, ya que una aplicación a datos reales sería más compleja de procesar computacionalmente y nuestras computadoras no cuentan con la capacidad de procesarlo con facilidad. Al final nuestro programa debe regresarnos toda la secuencia del patrón que más se repite (Consensus) y el Score correspondiente a dicha Consenso (La magnitud del vector de Consensus).

# Código de programa en Python

---

```
1
2 import numpy as np
3 import sys
4
5 #Funcion que lee un archivo de texto y crea una lista multidimensional
6 def GetDNAMatriz(namefile):
7     DNA_Matriz = []
8     for line in open(namefile).readlines():
9         DNA_Matriz.append(list(line)[0:len(line)-1])
10    return DNA_Matriz
11
12    #Funcion que permite calcular el tamao del los patrones y aplicar las funciones
13    Alignment, Profile, Consensus y Score
14 def Motif_Finding_Problem(DNA_Sample, l):
15     lim = len(DNA_Sample[0]) - l + 1
16     align = []; prof = []; consen = []; consensusn = []
17     s=0; best = 0; bestconse = []
18
19     #Se realiza un for por cada muestra de ADN
20     for a in range(lim):
21         for b in range(lim):
22             for c in range(lim):
23                 for d in range(lim):
24                     for e in range(lim):
25                         for f in range(lim):
26                             for g in range(lim):
27
28                                 #Es necesario llamar las funciones para cada combinatoria
29                                 de los patrones.
30                                 Si = [a,b,c,d,e,f,g]
31                                 align = alignment(DNA_Sample, l, Si)
32                                 prof=profile(align)
33                                 consen, consensusn=consensus(prof)
34                                 s=score(consensusn)
35
36                                 #Esto se realiza para obtener el mejor score y su valor de
37                                 consenso
38                                 if (s > best):
39                                     best=s
40                                     bestconse = consen
41
42     #Regresa el mejor score y la lista de los nucleotidos que lo conforman
43     return best, consen
44
45 #Funcion de alineamiento que obtiene las combinatorias y los almacena en una lista
46     multidimensional que funciona como una matriz, el tamao del patron y el vector de
```

```

    combinaciones
42 def alignment(matriz, l, Si):
43
44     align = []
45
46     #La funcion Enumerate regresa el valor de la lista y el indice actual donde se
        trabaja
47     for i, line in enumerate(matriz):
48
49         #Se guarda en una lista llamada align y retorna la lista de acuerdo a la
            combinatoria del vector Si
50         align.append(line[Si[i]:Si[i]+1])
51
52     return align
53
54 #Funcion de matriz de perfil
55 def profile(matriz):
56
57     #Se crea un arreglo el cual se usara para comparar los nucleotidos del ADN
58     DNA = [['A', 'a'], ['T', 't'], ['G', 'g'], ['C', 'c']]
59
60     #Se transpone la matriz para para alinear las posiciones y hacer el conteo mas
        rapido
61     ls=np.asarray(matriz).transpose().tolist()
62
63     #Se crea el arreglo donde se van a guardar los valores obtenidos
64     prof = []
65
66     #Se crea dos for para tomar todos los valores de la matriz transpuesta y comparar
        los valores con el vector de nucleotidos para llevar a cabo la sumatoria de los
        elementos en la matriz transpuesta
67     for i in ls:
68         temp = []
69         for j in DNA:
70             temp.append(i.count(j[0])+i.count(j[1]))
71         prof.append(temp)
72
73     #Se retorna un vector de las repeticiones de lo nucleotidos para cada columna de la
        lista multidimensional
74     return prof
75
76 #Funcion del Consenso, retorna el valor del consenso y el vector con las sumatorias
    perteneciente a cada nucleotido
77 def consensus(matriz):
78
79     DNA = ['A', 'T', 'G', 'C']
80     ls = []
81     ne = []
82

```

```
83     #Se realiza un for para obtener el valor maximo de cada columna de la matriz de
      perfil
84 for i in matriz:
85
86     #En esta parte se obtiene el valor mximo del arreglo y el indice donde se
      encuentra en la lista
87     ma = np.argmax(np.array(i))
88     m = max(i)
89     #Se guarda los valores obtenidos en otras listas para retornarlas al final
90     ls.append(DNA[ma])
91     ne.append(m)
92
93     return ls, ne
94
95 #Funcion del Score realiza la suma del vector de consenso y regresa el valor de toda la
      sumatoria del vector de consenso
96 def score(lista):
97     s = 0
98     for i in lista:
99         s+=i
100     return s
101
102 #Inicio de mi funcion main
103
104 def main():
105     # Tamao del patron a buscar
106     l=8
107
108     #Declaracion de la variable de score y lista que se van a obtener de la funcion
      Motif
109     sco=0
110     conse=[]
111
112     #Lectura del archivo de texto y creacin de una lista multidimensional
113     sample=GetDNAMatriz("Arreglo.txt")
114
115     #Aplicacion de la funcion de busqueda de motivos
116     sco, conse = Motif_Finding_Problem(sample,l)
117
118     #Impresion del mejor score y concenso pertenecientes a la muestra de ADN a estudiar
119     print("Best Score:", sco)
120     print("Consensus:", conse)
121     return 0
122
123 if __name__ == "__main__":
124     sys.exit(int(main() or 0))
```

---

# Tiempo de ejecución

```

jj@pcerdo:~$ python Motif\Finding\Problem.py
top - 15:20:16 up 1 day, 20:21, 1 user, load average: 3.63, 3.26, 2.72
Tasks: 201 total, 2 running, 148 sleeping, 0 stopped, 0 zombie
%Cpu(s): 77.4 us, 7.1 sy, 0.0 ni, 15.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 3922624 total, 953988 free, 1714064 used, 1254572 buff/cache
KiB Swap: 4079612 total, 2793288 free, 1286324 used, 1854552 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+   COMMAND
 22529 jj         20   0   32852   2388   1788  R   87.9   0.1   1351:38 python
  9886 jj         20   0 2079860 328332 142084 S   22.9   8.4   11:23:03 Web Content
  4270 jj         20   0 2970884 374504 99164 S   16.3   9.5   78:43.22 firefox
  1636 root         20   0 335316  42672 23824 S   15.7   1.1 261:18.77 Xorg
 13977 jj         20   0 376456  26776 20792 S    6.9   0.7   0:00.21 xfce4-scre+
  2931 jj         20   0  903064  10528  8476 S    3.9   0.3   2:54.08 pulseaudio
 12170 jj         20   0 703088  44788 34768 S    3.3   1.1   3:17.28 pavucontrol
 22517 jj         20   0 606860  17860 10908 S    2.3   0.5   1:49.79 xfce4-term+
 2535 jj         20   0 188968  15560 11468 S    2.0   0.4   3:01.62 xfwm4
 2538 jj         20   0 269152  14580 11080 S    1.3   0.4   0:56.40 xfce4-panel
 2659 mssql       20   0 2076852 193936  3828 S    1.3   4.9   34:36.95 sqlservr
 22534 jj         20   0 2780192 473784 60444 S    0.7  12.1   38:52.63 Web Content
 23730 jj         20   0 2024488 184544 70440 S    0.7   4.7   15:50.11 Web Content
 26548 jj         20   0  43660  2696  1916 R    0.7   0.1   7:41.82 top
  1388 rstudio+    20   0  271452   364   312 S    0.3   0.0   0:29.05 rserver
  1505 root        20   0  217136   576   468 S    0.3   0.0   0:08.53 php-fpm7.0
  1769 root        20   0  833388  19300  9740 S    0.3   0.5   5:30.49 dockerd
  
```

Daft Punk - Instant Crush (Video) ft. Julian Casablancas

238.355.105 visualizaciones 1,2 M

En la imagen anterior podemos apreciar que el proceso lleva más de 24 horas de ejecución, empezó a correr a las 3 pm del día anterior y seguía realizando el mismo procedimiento, a partir de este momento no sabíamos si el algoritmo estaba funcionando de manera correcta o no, por lo que decidimos seguir esperando para esperar un resultado.

The screenshot shows a terminal window titled "Terminal - jj@pcerdo: ~" with the following output:

```

top - 16:56:21 up 1 day, 21:58, 1 user, load average: 2.80, 2.19, 1.83
Tasks: 202 total, 2 running, 147 sleeping, 0 stopped, 0 zombie
%Cpu(s): 69.2 us, 5.6 sy, 0.0 ni, 25.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 3922624 total, 1092644 free, 1564564 used, 1265416 buff/cache
KiB Swap: 4079612 total, 2806600 free, 1273012 used, 2016252 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND
 22529 jj        20   0   32852    2388    1788  R   87.8   0.1   1446.36 python
 22534 jj        20   0   2777120 469524   60592  S   12.9  12.0   41:22.18 Web Content
 23730 jj        20   0   2026120 184948   87556  S   11.6   4.7   17:16.94 Web Content
 1636 root        20   0   338276   47556   28440  S   10.6   1.2   262:22.90 Xorg
 4270 jj        20   0   2938864 342736  103356  S    8.3   8.7   81:48.88 firefox
 16479 jj        20   0   376440   26980   21004  S    5.9   0.7    0:00.18 xfce4-scre+
 22517 jj        20   0   606860   17860   10908  S    3.6   0.5    1:58.83 xfce4-term+
 2535 jj        20   0   188968   15572   11476  S    1.3   0.4    3:06.47 xfwm4
 2538 jj        20   0   269152   15224   11476  S    1.3   0.4    0:57.09 xfce4-panel
 2659 mssql      20   0   2085032 198696   3812  S    1.3   5.1   35:32.68 sqlservr
 2931 jj        20   0   208064   10796   8744  S    1.3   0.3    4:10.22 pulseaudio
 2587 jj        20   0   486724   124304   18264  S    1.0   0.6    0:10.10 panel-l-wh+
 12170 jj        20   0   703088   44788   34768  S    0.7   1.1    3:56.87 pavucontrol
 26548 jj        20   0   43660    2696    1916  R    0.7   0.1    8:19.17 top
 1388 rstudio    20   0   271452    364     312  S    0.3   0.0    0:29.70 rserver
 2261 jj        20   0   50700    2996    2004  S    0.3   0.1    0:13.87 dbus-daemon
 2417 jj        20   0   341224    8544    5416  S    0.3   0.2    0:04.79 xfce4-sess+
  
```

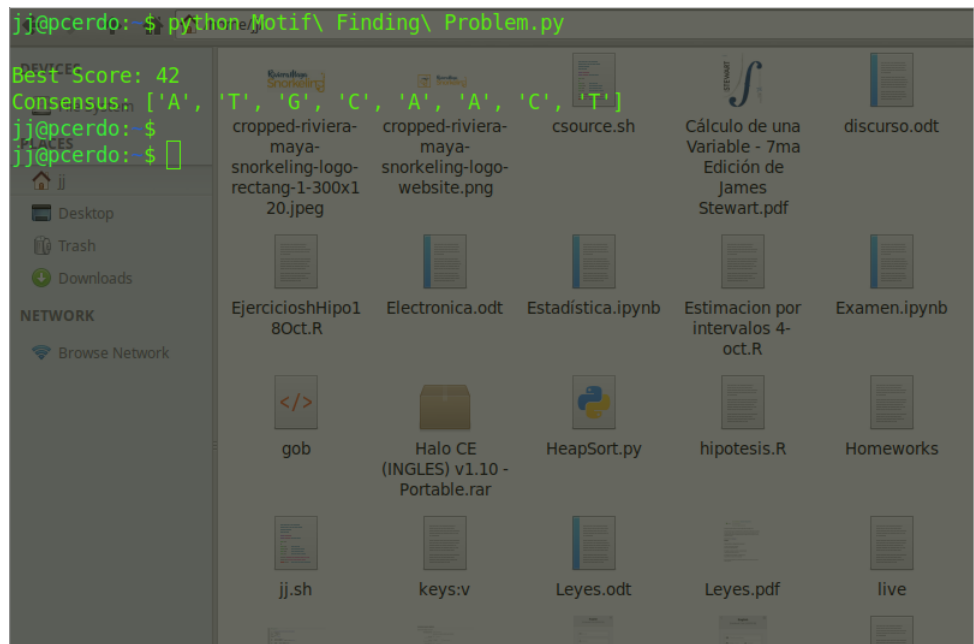
The background shows a web browser with the University of the Caribbean website, featuring a banner for a "29th WORLDWIDE CONFERENCE" and logos for "Contraloría Social" and "Portal de Transparencia".

Seguíamos esperando a que se terminara pronto el proceso ya que es muy difícil de esta manera saber que ocurre dentro del programa, antes de realizar la prueba con una muestra de 7, utilizamos una muestra de 2 para visualizar si el código trabajaba de manera correcta, lo cual si se logró, pero al ser una cantidad muy grande de combinatorias el programa se toma mucho tiempo en resolver el problema.



# Output del programa

```
jj@pcerdo:~$ python Motif\ Finding\ Problem.py
Best Score: 42
Consensus: ['A', 'T', 'G', 'C', 'A', 'A', 'C', 'T']
jj@pcerdo:~$
```



Al finalizar la ejecución del código pudimos observar que efectivamente funciona en la resolución del problema de búsqueda de motivos.

# Conclusión de Motif Finding Problem

A pesar de los considerables esfuerzos realizados hasta la fecha, la búsqueda de motivos de ADN sigue siendo un desafío complejo para los biólogos y los científicos informáticos. Los investigadores han adoptado muchos enfoques diferentes para desarrollar herramientas de descubrimiento de motivos y el progreso realizado en esta área de investigación es muy alentador. La comparación de rendimiento de diferentes herramientas y algoritmos de búsqueda de motivos, así como la identificación de los mejores patrones han demostrado ser una tarea difícil porque las herramientas están diseñadas con base que los algoritmos al ser de forma cerrada a una sola solución hace que este sea un problema complejo de resolver, siendo la implementación muy simple pero al mismo tiempo inexacto, los modelos de motivos son diversos y complejos para diferentes tamaños de muestras y patrones.

Esta primera solución que propusimos con el programa de búsqueda de motivos conlleva a una bastante ineficiencia de proceso, ya que revisa combinación por combinación, todos los ciclos for que itera por cada línea de secuencia, cada comparación lleva un determinado tiempo considerable, haciendo que al final el programa dure en su ejecución más de 24 horas.

La cantidad de combinaciones es:  $33^7$  (42,618,442,977) todo el tiempo que tomó procesar todas esas combinaciones nos dio a entender que el tiempo de ejecución y de procesamiento, consume demasiados recursos y este algoritmo no es tan eficaz para computadoras de pocas prestaciones de hardware, por lo que deducimos que hay maneras más funcionales de abordar el problema tomando en cuenta cuestiones probabilísticas y que delimiten un margen de combinación más eficaz.