Implementación algoritmo genético.

Determinación de las reglas evolutivas debidas al fenómeno de vecindad a partir de secuencias originales y modificadas de DNA.

Rubén Rodríguez Fernández

[8 mayo 2013]

[rrodrf04@estudiantes.unileon.es]

Summary

1.- Programación del algoritmo genético

Para llevar a cabo la programación del algoritmo genético hemos utilizado el lenguaje de programación Perl y la librería AI Genetics. En la guía se pide calcular una matriz M que cumpla las siguientes condiciones :

- C(t+1) = M x C(t)
- Que la matriz M tenga tres unos consecutivos en cada fila y lo demás ceros.

El tamaño de búsqueda inicial era 2^{n*n} siendo n el número de columnas y filas de la matriz M. A partir de las condiciones aportadas hemos conseguido reducir el tamaño de búsqueda con la segunda condición. Como los unos siempre son consecutivos y lo demás son ceros, podemos representar los unos de una fila como un número entero que varía entre 1 y n-2 siendo este número la posición del segundo uno de la serie de 3, con lo cual hemos simplificado el problema a buscar una matriz de dimensiones [1,n] en la cual sus números como anteriormente se dijo varían entre 1 y n-2. Con esto conseguimos que todas las matrices cumplan la primera condición y a partir del grado de cumplimiento de la primera podremos determinar si una matriz es mejor que otra. Un ejemplo de esta mejora sería el siguiente :

$$\boldsymbol{M} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{Mmodificada} = \begin{bmatrix} 1 & 1 & 3 & 2 & 2 \end{bmatrix}$$

Otro ejemplo podría ser :

$$\boldsymbol{M} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{Mmodificada} = \begin{bmatrix} 2 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Como se puede observar la primera columna es 0 y la última n – 1. El número de elementos de la matriz Mmodificada es n y el rango de estos (1, n-2). Por lo cual el espacio de búsqueda es de $(n-2-1)^n$, para n=20, el tamaño del problema se reduciría de 2^{400} a 17^{20} lo cual es $6{,}353599536 \times 10^{95}$ veces mejor.

1.2.- Utilización del programa

El programa por defecto opera con las secuencias c(t)= AAAACGGGGGAAATTTAGCA y c(t+1)=ACGACGGGGGGTTTAAAACA , con una población inicial de 50 individuos y 50 iteraciones (puede no llegar a hacerlas todas si encuentra una matriz válida antes). Estos argumentos pueden ser modificados a la hora de lanzar el programa, pudiendo utilizar otras secuencias y otro número de iteraciones. Unos ejemplos con otras secuencias sería :

perl programa.pl TGTG AAAA

perl programa.pl TGTG AAAA 10

perl programa.pl AAAACGGGGGAAATTTAGCA ACGACGGGGGGTTTAAAACA

perl programa.pl AAAACGGGGGAAATTTAGCA ACGACGGGGGGTTTAAAACA 100

(Nota, el nombre del programa y la forma de lanzar perl pueden cambiar, dependiendo del sistema operativo y del nombre del archivo)

Las consideraciones a la hora de utilizar argumentos son que ambas secuencias tengan la misma longitud (y las letras sean A C G T) sin estar separadas las letras, dentro de una secuencia, y el número de iteraciones mayores que 0 en caso de no cumplirse una de estas condiciones el programa finalizará.

1.2.- Detalles de implementación

Para determinar la puntuación de una matriz seguimos las indicaciones de la guía, la suma de todos los elementos determina la puntuación, cuanto más grande sea este número peor será esta matriz. Nosotros necesitamos que cuanto mejor sea la matriz mayor sea este número, para conseguirlo calculamos la puntuación máxima que es el número de filas (c(t+1) es una matriz de la forma (1, númeroFilas)) por el rango de valores, en nuestro caso sería :

$$puntuaciónMaxima = númeroFilas * 4$$

Por lo cual la puntuación de una matriz sería :

$$puntuacion = puntuaciónMáxima - sumaDeTodosLosElementos$$

En lo que respecta de pasar de nuestra Mmodificada a M para poder multiplicarla por c(t), el algoritmo para conseguir esto en pseudo-código podría sería :

```
para i=0 hasta  filasM - 1
        para j=0 hasta filasM – 1
                si ((Mmodificada[i][1] – j) == ( 0,1,2))
                        M[i][j] = 1
                sino
                        M[i][j] = 0
                fin si
        fin para
fin para
```

Otro aspecto sería el número de iteraciones que realiza, que puede no ser el número establecido ya que si encuentra una matriz que tiene la máxima puntuación el algoritmo genético no necesita realizar más iteraciones.

2.- Conclusión

Hemos abordado la implementación de un algoritmo genético para solucionar el problema presentado. El algoritmo entregado se puede optimizar bastante pero se ha decidido sacrificar rendimiento por claridad y facilidad en la implementación.

En este problema no es necesario utilizar un algoritmo genético y se podría reducir de forma drástica el tiempo de ejecución utilizando un método que no utilice fuerza bruta. Con la pequeña modificación aplicada hemos conseguido reducir el espacio de búsqueda por ejemplo, trabajando con genes reales de 27000 pares de bases (de media) el espacio de búsqueda sería de $(2^{27000})^{27000}$ por fuerza bruta, con nuestra modificación el espacio de búsqueda sería de $(27000 - 2 - 1)^{27000}$ que es abordable aunque este problema se podría reducir utilizando un algoritmo convencional a un algoritmo con complejidad computacional n y lo resolvería en segundos lo que es abrumadoramente mejor que los algoritmos genéticos pero este algoritmo no es el propósito de este trabajo.

Si se desea más información acerca de este algoritmo que podría resolver este problema con 27000 pares de bases en segundos o menos pueden enviarme un correo electrónico a la dirección que aparece en la portada.

3.- Crítica a la guía

La guía es clara y consisa, contiene todos los apartados requeridos y un ejemplo desarrollado completamente.