Universidad Nacional Autónoma de México

Facultad de Ciencias

Tarea 05:

# Algoritmo genético simple y bloques constructores

*Pablo A. Trinidad Paz* - 419004279

Trabajo presentado como parte del curso de **Cómputo Evolutivo** impartido por el profesor **Mario Iván Jaen Márquez**.

Fecha de entrega: **Jueves 28 de Marzo de 2019**.

1. Tenemos dos cromosomas padres distintos en un SGA con codificación binaria, cada uno con $L$ bits. Seleccionamos aleatoriamente un punto de cruza $c \in 1, ..., L-1$ y realizamos la cruza de un punto. ¿Cuál es la probabilidad de que ambas soluciones hijas sean clones (idénticas) de sus padres?

   **Solución:** La cruza de dos cromosomas padres $a$ y $b$ de longitud $L$, tal que $a \neq b$, en un punto $c \in \{1, ..., L-1\}$ genera dos hijos $p$ y $q$

$$p = a_1 + b_2$$
$$q = b_1 + a_2$$

   donde $a_1$ y $b_1$ son las cadenas de los cromosomas padres $a$ y $b$ respectivamente antes del punto de cruza $c$ y $a_2$ y $b_2$ son las cadenas después del punto de cruza. Dicho lo anterior, se considera que las soluciones hijas $p$ y $q$ son clones de sus padres si $(p = a \wedge q = b) \vee (p = b \wedge q = a)$ recordando que si ambos se cumplen, entonces $a = b$ y eso no es posible en el planteamiento dado. También se puede ver que si las subcadenas de lo padres cumplen que $a_1 = b_1 = k$ o $a_2 = b_2 = k$, donde $k$ es una cadena cualquiera que existe en la misma región dentro de los padres, entonces los hijos serán clones de los padres. Lo anterior es cierto ya que:

$$\text{si } a_1 = b_1 = k, \text{ entonces:}$$
$$a = k + a_2, b = k + b_2$$
$$p = k + b_2, q = k + a_2$$
$$\implies p = b \text{ y } q = a$$

$$\text{o si } a_2 = b_2 = k, \text{ entonces:}$$
$$a = a_1 + k, b = b_1 + k$$
$$p = a_1 + k, q = b_1 + k$$
$$\implies p = a \text{ y } q = b$$

   Por lo tanto, la probabilidad que nos interesa hallar es el evento donde $a_1 = b_1$ o (exclusivo) $a_1 = b_2$. Redefinimos la notación de los cromosomas de los padres $a$ y $b$ como

$$a = a_1 a_2 a_3 \ ... \ a_c a_{c+1} a_{c+2} \ ... \ a_L$$
$$b = b_1 b_2 b_3 \ ... \ b_c b_{c+1} b_{c+2} \ ... \ b_L$$

   De tal forma que nos interesa que $a_1 a_2 \ ... \ a_c = b_1 b_2 \ ... \ b_c$ o (exclusivo) $a_{c+1} a_{c+2} \ ... \ a_L = b_{c+1} b_{c+2} \ ... \ b_L$. Usando una codificación diferente, definamos una nueva serie de cadenas $r$ tal que

$$r = a + b = a_1 a_2 a_3 \ ... \ a_c a_{c+1} a_{c+2} \ ... \ a_L + b_1 b_2 b_3 \ ... \ b_c b_{c+1} b_{c+2} \ ... \ b_L$$
$$r_1 = a_1 a_2 a_3 \ ... \ a_c b_1 b_2 b_3 \ ... \ b_c$$
$$r_2 = a_{c+1} a_{c+2} \ ... \ a_L b_{c+1} b_{c+2} \ ... \ b_L$$

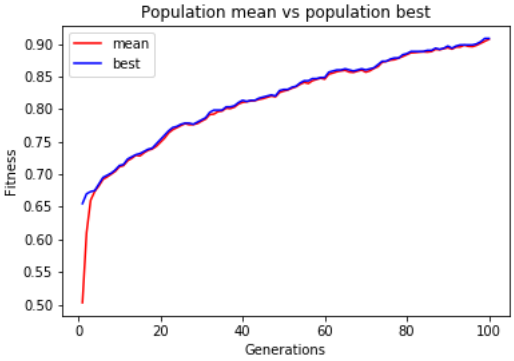
   Donde el número de cadenas posibles de $r$ es igual a $2^{2L}$, las cadenas posibles de $r_1$ es $2^{2c}$ y las cadenas posibles de $r_2$ es $2^{2L-2c}$. Y ya que nos interesa que suceda únicamente un solo caso de $r_1$ o un sólo caso de $r_2$ (descrito arriba), la probabilidad $p$ de que la cruza genere clones es:

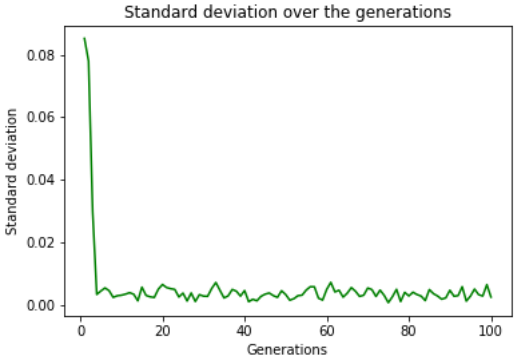
$$p = \frac{1}{2^{2L}} - \frac{1}{2^{2c}} - \frac{1}{2^{2L-2c}}$$

2. El problema ***one-max*** consiste en buscar una cadena binaria de $L$ bits con el mayor número de unos posible, es decir, el fitness de una cadena es el número de unos que contiene, por lo que se quiere maximizar $f(x) = \sum_{i=1}^{L} x_i$ con $x_i \in \{0, 1\}$. Por supuesto que podemos resolver este problema de forma sencilla escribiendo 1s consecutivamente en la solución, pero estamos interesados en ver si el SGA puede resolverlo. Implementa en Python el SGA usando

   - Selección proporcional (FPS)
   - Cruza de un punto con $P_c = 0.7$
   - Mutación bit a bit con $P_m = 0.01$
   - Tamaño del cromosoma $L = 30$
   - Límite de generaciones: 100
   - Tamaño de población: 20

   *a*) Lanza 20 simulaciones del algoritmo y grafica el fitness del mejor individuo y el fitness de la población como función del número de generación.



3. Usando el paquete SCHEMATAX de Python: https://github.com/iSTB/python-schemata vamos a validar de forma experimental el teorema de los esquemas de J. Holland, calculando los esquemas procesados en cada iteración de SGA para observar los bloques constructores.
   Usando el problema anterior

   *a*) Grafica el orden promedio y tamaño de definición promedio de los esquemas procesados.

   *b*) Considerando que un esquema tiene bajo orden si está por debajo del orden promedio y de igual forma tiene bajo tamaño de definición si está por debajo del tamaño de definición promedio. Grafica el orden promedio y tamaño de definición promedio de los bloques constructores.

   *c*) Grafica el histograma de frecuencias sobre los bloques constructores

   *d*) Selecciona los 10 bloques constructores más frecuentes.