Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Técnicas de inteligencia artificial

Reporte: Ejemplo de un algoritmo genético

BUAP

Docente: Abraham Sánchez López

Alumno	Matrícula
Taisen Romero Bañuelos	202055209

Algoritmo genético en C

Para poder identificar más fácilmente los pasos del algoritmo revisé de nuevo el algoritmo que venía en las diapositivas de "Computación evolutiva".

- Ejemplo de un algoritmo genético simple:

```
Generar una población inicial
Calcular la función de evaluación de cada individuo
Mientras no TERMINADO hacer
Inicio   // producir nueva generación
    Para tamaño_población/2 hacer
  Inicio   // ciclo reproductivo
    Seleccionar dos individuos de la anterior generación, para el cruce (probabilidad de selección
    proporcional a la función de evaluación del individuo).
    Cruzar con cierta probabilidad los dos individuos obteniendo dos descendientes.
      Mutar los dos descendientes con cierta probabilidad.
    Calcular la función de evaluación de los dos descendientes mutados.
      Insertar los dos descendientes mutados en la nueva generación.
    Fin
  Si la población  ha convergido entonces TERMINADO = CIERTO
    Fin
Fin
```

Esta estrategia fue realmente efectiva. En el código de geneti.c pude identificar que primero se definen las variables que se mencionan al inicio de la diapositiva (tamaño de la población, probabilidad de cruza, probabilidad de mutación, número máximo de generaciones, etc.), esto pues es obvio, siempre en un programa C se empieza definiendo variables, pero hasta en nuestro algoritmo de las diapositivas se manifiesta esta inicialización de las cosas. Lo interesante viene después, pues, entramos al bucle para crear una generación con base a los parámetros asignados. Cada generación selecciona dos individuos (mates) para hacer un torneo para seleccionar al mejor (con select() se hace el torneo) y luego realiza un cruce con la probabilidad de PC (o sea, cabe la posibilidad de que no se reproduzcan). En caso de que hayan tenido un cruce exitoso se aplica la mutación con la probabilidad PM (que por curiosidad le puse 5 para que salgan cromosomas bien mutantes). Luego se calcula su aptitud con base a la función $f(x) = x^2$ normalizada, después se hace un reporte de la nueva generación y se actualiza la población intercambiando oldpop ←→ newpop. Este proceso se repite Gmax veces (o sea, el número de generaciones que el usuario introduzca).

Ahora, pasemos de la teoría a lo empírico para entender mejor cómo funciona el algoritmo.

Para este ejemplo usé los siguientes parámetros:

- Tasa de mutación (Pm): 5.0.
- Probabilidad de cruce (Pc): 0.8.
- Número máximo de generaciones (Gmax): 2.

- Semilla para aleatoriedad (Rseed): 0.7.
- Tamaño de la población (popsize): 100.
- Longitud del cromosoma (codesize): 16.

Obtuve los siguientes resultados para la generación 0:

```
                                P O P U L A T I O N          R E P O R T
Generation #   0                                Generation #   1
num      string          x       fitness    parents    x1   x2  mut place     string          x       fitness
 1)  0100111100100011 20259.0000 0.0956 | (  1,  4)   12    14   Y   2  0110111100100111 28455.0000 0.1885
 2)  0010100101000010 10562.0000 0.0260 | (  1,  4)   12    14   Y  13  1011001110010111 45975.0000 0.4921
 3)  0001111011110011  7923.0000 0.0146 | (  6,  7)    0     0   Y   4  0101010011110111 21751.0000 0.1102
 4)  1011001110010111 45975.0000 0.4921 | (  6,  7)    0     0   Y   7  1101110100011100 56604.0000 0.7460
 5)  0010010100010110  9494.0000 0.0210 | (  9, 11)    0     0   Y   9  1110110000000111 60423.0000 0.8501
 6)  0101110011110111 23799.0000 0.1319 | (  9, 11)    0     0   Y   1  1011001001011011 45659.0000 0.4854
 7)  1101110000011100 56348.0000 0.7393 | ( 14, 15)   13    15   Y  12  1001101010100010 39586.0000 0.3649
 8)  0010101011001101 10957.0000 0.0280 | ( 14, 15)   13    15   Y  12  1110100101010111 59735.0000 0.8308
 9)  1110110001000111 60487.0000 0.8519 | ( 18, 46)    4     5   Y  12  1110001010100000 58016.0000 0.7837
10)  1001110111111110 40446.0000 0.3809 | ( 18, 46)    4     5   Y  11  1111111000111100 65084.0000 0.9863
11)  1111001001011011 62043.0000 0.8963 | ( 56, 23)    0     2   Y   9  1000111100001001 36617.0000 0.3122
12)  1010000010100011 41123.0000 0.3938 | ( 56, 23)    0     2   Y   3  1011001101000110 45894.0000 0.4904
13)  0111111111011011 32731.0000 0.2494 | ( 71, 28)    4     8   Y   1  1100000100011000 49432.0000 0.5689
14)  1001101010101110 39598.0000 0.3651 | ( 71, 28)    4     8   Y   9  1001110010101101 40109.0000 0.3746
15)  1110100101011011 59739.0000 0.8309 | ( 30, 32)    3    10   Y  14  1100110111101100 52716.0000 0.6471
16)  0010001100110110  9014.0000 0.0189 | ( 30, 32)    3    10   Y   0  1111000011100010 61666.0000 0.8854
17)  0011100001010100 14420.0000 0.0484 | ( 24, 36)   10    11   Y   8  1010100111100010 43490.0000 0.4404
18)  1110101010101000 60072.0000 0.8402 | ( 24, 36)   10    11   Y   9  1100100111110110 51702.0000 0.6224
19)  0000111000110101  3637.0000 0.0031 | ( 86, 39)    5    10   Y   8  1111110010010110 64662.0000 0.9735
```

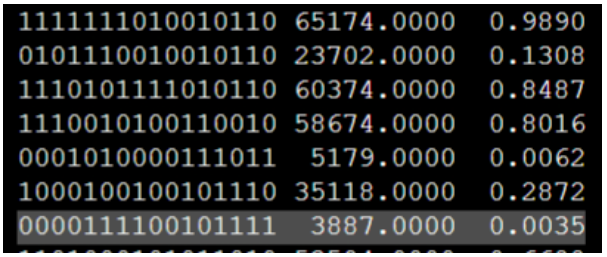
```
Generation # 0 Accumulated Statistics:
Total Crossovers = 38, Total Mutations = 100
min = 0.006600   max = 0.986300   avg = 0.536906   sum = 53.690600
Best individual in this generation = 1111111000111100
Global Best Individual so far, Generation # 0:
String = 1111111000111100    Fitness = 0.986300
```

Ya desde esta generación la aptitud promedio mejoró, pasó de 0.5369 a 0.6464. El macho alfa de esta generación fue “1111111000111100”, que al parecer tuvo de hijo al mejor de la siguiente generación (pronto llegaremos a eso).

En la segunda generación hubo un aumento en la variabilidad genética porque se hicieron 74 cruces y 200 mutaciones (un gran aumento con respecto a la generación pasada que tuvo 38 cruces y 100 generaciones). También, la aptitud máxima (0.98) roza el 1.0, lo que sugiere que el algoritmo ha encontrado una solución óptima o cercana a ella. Lo más interesante es que el mejor individuo de esta generación parece conservar una buena parte del cromosoma del mejor individuo de la generación pasada.

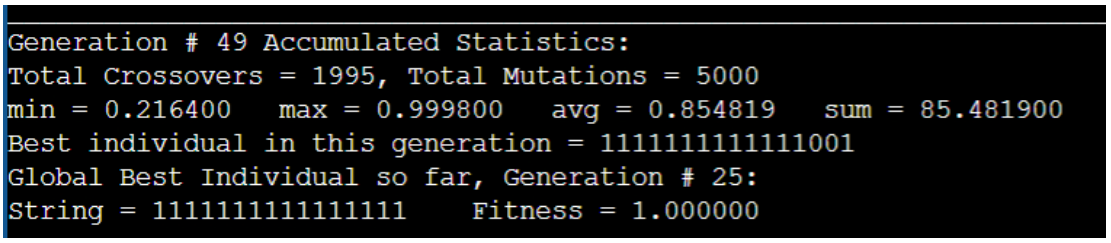
```
Generation # 1 Accumulated Statistics:
Total Crossovers = 74, Total Mutations = 200
min = 0.003500   max = 0.998700   avg = 0.646376   sum = 64.637600
Best individual in this generation = 1111111111010110
Global Best Individual so far, Generation # 1:
String = 1111111111010110    Fitness = 0.998700
```

Algo que noté mientras revisaba la lista de individuos es que aquellos que tenían muchos unos tenían una mejor aptitud. Véase el contraste entre el individuo subrayado (con una aptitud del 0.3%) y el primero, con una aptitud notablemente superior.



Si nos ponemos a pensar, obtuvimos una aptitud elevada (0.98) demasiado rápido, en tan sólo dos generaciones nos acercamos a la solución óptima, ¿Cómo fue posible esto?, quizás se deba a que me excedí en la probabilidad de mutación, pues, los valores típicos oscilan entre 1.0 y 0. Esto pudo haber favorecido la exploración del espacio de soluciones, pero también pudo haber causado una mayor aleatoriedad en la evolución. Además, puse una buena tasa de cruce (0.8) pensando en que es raro que los individuos se reproduzcan sin éxito, seguramente eso también ayudó a la mezcla de los cromosomas.

Me causa ruido que según nuestro algoritmo tener una mutación excesiva es bueno para la aptitud de los individuos. Si nos ponemos a pensar con detenimiento, la mutación excesiva es igual de perjudicial o más que el elitismo de individuos, así que para poder observar mejor la tendencia del comportamiento de nuestros parámetros voy a extender las generaciones a 50.



Definitivamente se corroboró mi teoría de que cuantos más unos mejor es la aptitud. Sin embargo, creo que tener un fitness=1.0 es una señal de sobreentrenamiento. En promedio, la aptitud es de 0.8, lo cual está bien pero si consideramos que nuestro mejor individuo es prácticamente un dios frente a los demás creo que no estamos generando un buen modelo. Nuestro Dr. Manhattan (1111111111111111) tiene un comportamiento particular. Apareció en la generación 25 pero rara vez aparece dentro de las generaciones, a excepción de la generación 41, ahí tuvo hijo pero de forma anti-intuitiva tuvo un hijo con una aptitud extremadamente baja (0.2).

```
62)  1111111111111111 65535.0000  1.0000 | ( 21, 23)    0     0   Y   0  0111101010010000 31376.0000  0.2292
63)  1110111101101101 61293.0000  0.8747 | ( 26, 27)    0     0   Y  13  1111111111101011 65515.0000  0.9994
64)  1111111110001010 65418.0000  0.9964 | ( 26, 27)    0     0   Y   2  1101111100001110 57102.0000  0.7592


97)  1110101101000110 60230.0000  0.8447 | ( 97, 62)
98)  0111111100010111 32535.0000  0.2465 | ( 97, 62)
99)  1111011111010010 63442.0000  0.9371 | ( 96, 66)
```

Bien, además de que hasta a los dioses los castigan, ¿Qué interpretación podemos sacar de este resultado?, pues que los individuos con una aptitud perfecta son escasos entre generaciones porque al reproducirse lo más probable es que se reproduzcan con un individuo con una peor aptitud, lo que significa que la aptitud de sus hijos se verá comprometida. Sacándolo del contexto del programa, quizá eso explique por qué a veces las leyendas humanas casi no tienen hijos que sean igual de impresionantes, y no sólo eso, también explicaría por qué es extraño que surjan individuos de este estilo, bien pueden pasar generaciones hasta que se de otro caso de un individuo excepcional.