Práctica 3

Programación Evolutiva

Aleix Garrido Oberink
Sergio Salmerón Majadas

Curso 10/11 Ing. Informática

El objetivo de esta práctica es aplicar la técnica de programación evolutiva a un problema de planificación. Disponemos de cubos distribuidos en una mesa y una pila, los cuales deberemos mover hasta conseguir la palabra objetivo "UNIVERSAL" en la pila mediante los operadores especificados en el enunciado.

Explicaremos en este informe puntos que consideramos destacables para comprender el funcionamiento de la práctica.

Construcción árbol correcto:

En esta práctica, los individuos que forman nuestra población son árboles. Los árboles se generan al principio de la ejecución, pudiendo mutar y cruzarse entre ellos a lo largo de las sucesivas generaciones.

Teniendo en cuenta la especificación dada en el enunciado, podemos ver que no cualquier árbol es correcto. Por lo tanto, debemos determinar en el momento de su creación qué condiciones debe cumplir un árbol para ser correcto.

Primeramente imponemos mediante una variable la profundidad máxima que puede alcanzar el árbol al ser generado. Además, sabemos que las hojas del árbol deben ser terminales, es decir, **CP, BS o SN**. Destacar también que los terminales **CP, BS o SN** solamente pueden ser hijos de las funciones **MM o MP**, y así mismo **MM o MP** sólo pueden tener como hijos **CP, BS o SN**.

Por otro lado, las funciones **EQ, NOT o DU** pueden tener como hijos cualquier función (**EQ, NOT, DU, MM, MP**), pero no podrán tener un terminal (CP, BS, o SN) como hijo.

A parte, recordar también el número de hijos que debe tener cada nodo. Dos hijos en los casos de **EQ y DU**; un hijo en el caso de **NOT, MM y MP**; ningún hijo en el caso de **CP, BS o SN**.

Tabla de resumen de árboles correctos

Elemento	Nº Hijos	**¿Puede ser hijo?**							
		CP	BS	SN	MM	MP	NOT	EQ	DU
CP	0	No	No	No	No	No	No	No	No
BS	0	No	No	No	No	No	No	No	No
SN	0	No	No	No	No	No	No	No	No
MM	1	**Sí**	**Sí**	**Sí**	No	No	No	No	No
MP	1	**Sí**	**Sí**	**Sí**	No	No	No	No	No
NOT	1	No	No	No	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
EQ	2	No	No	No	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**
DU	2	No	No	No	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**

Evaluación:

La evaluación de un cromosoma de la población se realiza mediante la ejecución de los casos de prueba. En la lista de casos de prueba contamos con 160 con situaciones de partida planteadas. La aptitud de un árbol será el número de casos que ha conseguido resolver, teniendo en cuenta el número de iteraciones que le esté permitido hacer a la función DU en el caso que estemos planteando.

Cruce:

El cruce entre dos árboles, en caso de producirse, debe cumplir una serie de requisitos. Consiste en el intercambio de dos ramas de los árboles que se desean cruzar. Obviamente, el resultado del cruce debe ser correcto, así que no pueden intercambiarse dos ramas cualesquiera, si no aquellas que sean compatibles. Con esto queremos decir que dos nodos (con sus respectivos hijos) son intercambiables siempre que los dos sean funciones, o siempre que los dos sean terminales.

Además, para evitar la construcción de un árbol demasiado grande, tenemos una variable de profundidad máxima que no debe sobrepasar un árbol al cruzarse. En caso de que el cruce planteado aleatoriamente no sea factible ya que se generaría un árbol demasiado grande, se busca otra vía de cruce posible, y sucesivamente hasta encontrar un punto que cumpla los requisitos (que siempre va a existir, aunque sea en los terminales).

Mutaciones:

- Terminal: La mutación terminal es la más sencilla. Consiste en elegir aleatoriamente una hoja del árbol y modificarla también aleatoriamente por otra de las hojas posibles. Por ejemplo, una hoja **CP** podrá mutar en una **BS** o una **SN**.
- Función: La mutación por función consiste en intercambiar un nodo función por otra función que cumple las condiciones necesarias para ocupar su lugar. De este modo **MM y MP** pueden mutar entre ellas, ya que las dos tienen un único hijo terminal. **DU y EQ** podrán mutar entre ellas ya que las dos tienen dos hijos que son funciones. **NOT** no podrá mutar ya que no existe ninguna función equivalente (que tenga el mismo número de hijos del tipo requerido).
- Árbol: La mutación por árbol consiste en la sustitución entera de una rama por otra. Para ello se elige un nodo al azar, y se crea a partir de ese nivel de profundidad un nuevo árbol (subárbol) que cumpla las condiciones necesarias para ocupar esa posición, teniendo en cuenta también que el tamaño del árbol resultante no exceda los límites impuestos.

Parámetros a tener en cuenta:

Hay tres parámetros en esta práctica que influyen directamente en el resultado y no forman parte del algoritmo evolutivo en sí, si no de los cromosomas.

Por una lado, **la profundidad del árbol de creación**, que determina el rango de profundidad que van a poder alcanzar los árboles generados aleatoriamente al inicializar la población.

También contamos con la **profundidad del árbol tras cruce o mutación**, que determina la profundidad máxima que permitimos que pueda alcanzar un árbol a lo largo de las generaciones tras someterse a operaciones de cruce y mutación. Esta profundidad previene que se generen individuos con ramas demasiado grandes como para ser tratadas.

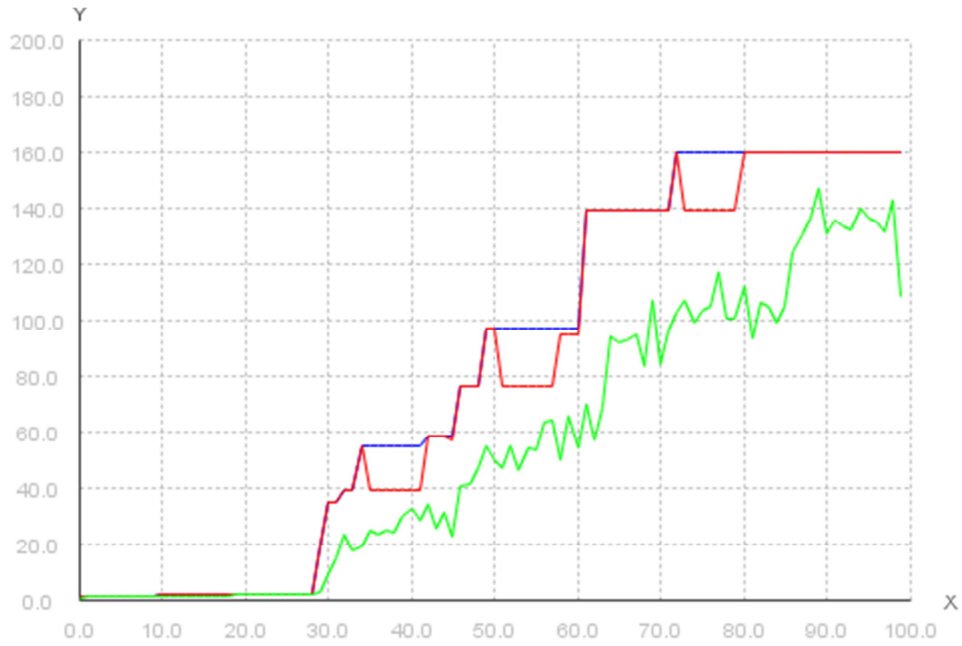
En último lugar también tenemos un limitador de **iteraciones máximas** que pueden producirse dentro de una operación DU. Por un lado permite la salida de lo que podrían ser bucles infinitos, o de costes computacionales demasiado elevados. Por otro lado, si se impone un límite pequeño la búsqueda de la solución puede verse dirigida a planificaciones más eficientes.

Casos estudiados:

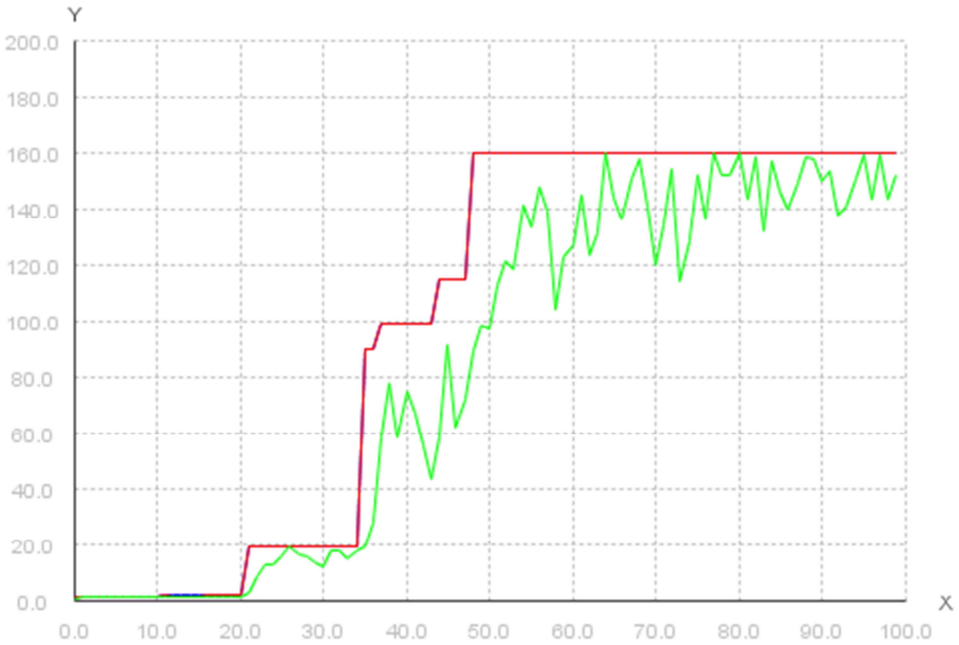
Hemos probados distintas combinaciones con distintos parámetros. El resultado ideal (160) es alcanzable con prácticamente todas las combinaciones, pero no al mismo coste.

Jugando con tamaños de árbol e iteraciones:

Por ejemplo, con árboles pequeños (profundidad inicial 3, y máxima profundidad permitida 5), y con pocas iteraciones (número de iteraciones 2) tenemos una ejecución muy rápida del problema, lo que nos permite realizar más generaciones. Pero con una población pequeña (20 ejemplares), las probabilidades de alcanzar una buena solución son muy bajas, pese a que en ocasiones ocurre, tal y como vemos en la siguiente imagen:



Si incrementamos el número de iteraciones permitidas, la probabilidad de llegar a la solución exacta aumenta, del mismo modo que aumenta el tiempo de ejecución. En el siguiente ejemplo aumentamos las iteraciones a 4.



Si subimos las iteraciones 10, por ejemplo, la probabilidad de obtener 160 aumenta, pero en algunas ejecuciones se convierte en una acción temporalmente inviable.

Probamos ahora con crear árboles mayores, por ejemplo, de tamaño 5 inicial y 7 tras el cruce. Con pocas iteraciones se ejecuta rápidamente, pero también es baja la probabilidad de encontrar una buena solución. Otra vez, subiendo el número de iteraciones la probabilidad aumenta, pero tiene más posibilidades de quedarse colgado buscando soluciones. Al ser el tamaño de los árboles mayor que antes, este problema puede darse más a menudo.

Por tanto, si dejamos un rango de árboles de tamaño medio y limitando las iteraciones a un tamaño número entorno a 4, se encontrará el equilibrio entre efectividad y resultados para poblaciones relativamente pequeñas (entre 20 y 50) y generaciones por debajo de 100.

Jugando con cruces y mutaciones:

La probabilidad de cruce influye directamente en las posibilidades de conseguir el resultado óptimo. Con una probabilidad de cruce muy baja, las probabilidades de alcanzar la solución son pequeñas, pero sigue sucediendo en algunas ocasiones. Nuevamente, al aumentar la probabilidad de cruce aumenta la posibilidad de llegar al resultado idóneo. Pero con valores demasiado altos podemos abusar del cruce forzando a cambiar elementos buenos empeorándolos. De modo que lo dejaremos en un término medio, alrededor del 40%.

Con la mutación sucede lo mismo que con el cruce. Para mutaciones muy bajas, no se nota el efecto, pero para mutaciones demasiado altas la media general de la población es baja. Con una mutación en torno al 7.5% se obtienen buenos resultados.

Si nos metemos a comprobar la eficiencia de los tres tipos de mutación, vemos que claramente el que mejor funciona es la mutación de árbol. Al tener una población pequeña, la mutación de árbol puede traer bastante novedad y mejoras. La mutación terminal no aporta prácticamente mejora. El cambio de enfoque que ofrece la mutación por función también da muy buenos resultados en situaciones puntuales, pero por lo general no aporta mejora.

Jugando con selección, elitismo y mejoras:

El elitismo en esta práctica no es tan decisivo como en otras, ya que guarda al principio individuos que resuelven pocos casos, pero que pueden no aportar nada a la evolución de la población. Pero en cierto modo sí que puede ayudar en un porcentaje bajo de elitismo.

Tanto la selección por torneo como por ruleta dan muy buenos resultados. Con las dos suelen alcanzarse los 160 casos resueltos. En cambio, el ranking empeora los resultados considerablemente respecto a los dos anteriores.

Por último, la contractividad, pese a su coste, nos permite alcanzar mejores resultados, al tener en cuenta su pase de una generación a otra únicamente cuando haya mejora. Esto permite reducir el número de generaciones necesarias a valores mucho menores.

Ejemplos de árbol resultante:

Valor 160.0 en el árbol (con iteraciones limitadas a 4):

```
DU
 NOT
  EQ
   MM
    CP
   MM
    BS
 DU
  EQ
   MM
    SN
   MM
    BS
  DU
   MP
    SN
   MP
    BS
```

Valor 160.0 en el árbol (con iteraciones limitadas a 6):

```
DU
 MP
  BS
 DU
  NOT
   NOT
    EQ
     MM
      SN
     MM
      SN
  DU
   MP
    SN
   MP
    BS
```