Complutense de Madrid

Práctica Programación Evolutiva.

Tercera práctica.

Raúl Torrijos & Lukas Häring

May 14, 2019

Tabla de contenidos

1 Introducción

En está versión hemos implementado una modificación del Algoritmo Genético en la que utilizamos cromosomas en forma de árbol para buscar una solución al problema de la Hormiga sobre el Rastro de Santa Fe, usando como función de evaluación el número de alimentos devorados por la hormiga sobre el tablero siguiendo las instrucciones de un programa generado de manera evolutiva.

1.1 Algoritmos implementados

Los algoritmos de selección, cruce y mutación que hemos utilizado para la resolución del problema de la hormiga son:

1. Algoritmos de Selección
   - Selección por Ruleta
   - Selección por Torneo Determinista
   - Selección por Torneo Probabilista
   - Selección por Ranking
   - Selección por Truncamiento
2. Algoritmos de Cruce
   - Método Subárboles
3. Algoritmos de Mutación
   - Terminal Simple
   - Funcional Simple
   - Subárbol
   - Permutación (no accesible en la interfaz)
4. Mejoras adicionales al guión
   - Pestaña de visualización de árboles

1.2 Aclaraciones

Nos gustaría puntualizar un par de pequeñas decisiones que hemos tomado a la hora de implementar el ejercicio:

- Hemos decidido eliminar todas las tildes de la interfaz de la aplicación por un motivo de problemas con nuestro software de control de versiones.

En cualquier caso nuestra implementación está preparada para ejecutar el problema con más o menos ciudades.

- La manera en la que controlamos el bloating es un con una operación que penaliza el fitness de un programa restando al número de alimentos devorados por la hormiga el tamaño del árbol en cuestión por un factor K, que hemos establecido en 0.5. Esto evita la aparición de árboles muy grandes.

Es por esto que con el control de bloating activado, el fitness de ciertos individuos se puede ver afectado por su tamaño y en la pestaña de gráficas puede no coincidir el máximo absoluto con el número de alimentos comidos en realidad.

Hemos hecho pruebas con el control de bloating desactivado y los árboles que genera son demasiado grandes.

2 Paneles de resultados

2.1 Gráficas

Idéntico al de las prácticas anteriores. Puede no reflejar resultados fieles a la cantidad de alimentos comidos por la hormiga si tenemos el control de bloating activo.

2.2 Pestaña Mapa

Nuestra aplicación incorpora una visualización del tablero toroidal de 32x32 con rastro de alimento en color negro, y tras ejecutar el AG mostrará la ejecución del mejor programa absoluto sobre el tablero, mostrando en color naranja el alimento devorado por la hormiga, y en color azul el recorrido de la hormiga realizado fuera del Rastro.

También se muestra la posición final de la hormiga tras representada con un cuadrado rojo con ojos negros, que indican la posición donde mira.

2.3 Pestaña Mejor Árbol

Como ya hicimos en la Práctica 2, hemos querido añadir un pequeño extra y en está ocasión hemos adjuntado una pestaña con un panel que, de igual manera, muestra el mejor resultado tras ejecutar el algoritmo y muestra el árbol de manera gráfica.

Esta característica nos fué extremadamente útil a la hora de comprobar que los algoritmos de cruce y mutación funcionaban correctamente.

Hemos usado la libreria org.abego.treelayout (https://github.com/abego/treelayout) para generar el árbol en Java Swing.

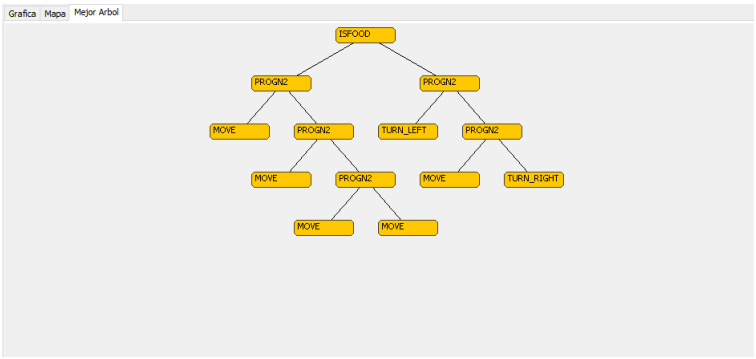

Figure 1: Pestaña de Mejor Árbol

3 Análisis de ejecuciones

Para realizar un análisis exhaustivo del comportamiento de cada uno de los algoritmos usados hemos realizado una gran cantidad de ejecuciones y hemos obtenido las siguientes conclusiones.

3.1 Peores resultados

De nuevo el método de selección de ruleta es el que menos favorece a la evolución por gran componente aleatoria.

Un número menor a 300 en el número de generaciones puede no ser suficiente para evolucionar de manera correcta por encima de los >70 alimentos comidos.

La contractividad en este experimento juega un mal papel ya que de por sí tenemos unos algoritmos muy costosos (gran cantidad de accesos a memoria) y no hace mejorar el proceso.

Figure 2: Ruleta sin elitismo (50 alimentos)

La única manera de obtener resultados mas razonables utilizando ruleta es aumentando el número de generaciones y usar elitismo para así aumentar la variedad en la población.

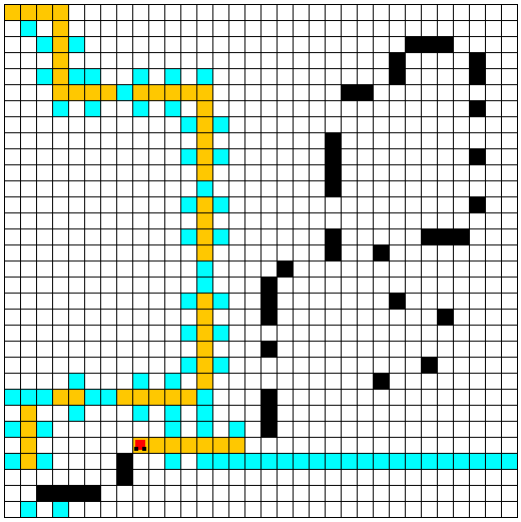

Figure 3: Ruleta sin elitismo (50 alimentos)

3.2 Mejores resultados

Los mejores resultados que hemos obtenido han sido aumentando el tamaño de población a 400, usando selección mediante Torneo Determinista o Ranking y mutación y cruce mediante subárboles.

En 10 ejecuciones hemos obtenido 4 veces el que consideramos el valor máximo **solución: 90 (o 89) alimentos**(esto es porque consideramos que en la posición inicial la hormiga come 1 alimento).

También nos ha ayudado mucho aumentar los valores de probabilidad de mutación y cruce hasta valores de 0.3 y 0.7 0.8 respectivamente y elitismo con valor 5.

Es importante que exista variedad en la población de individuos para aumentar la probabilidad de que se genere un programa que recorra el Rastro completo.

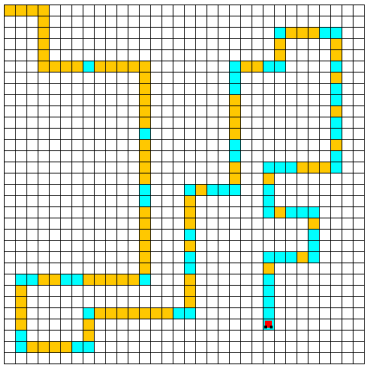

Figure 4: Representación en el tablero. Mejor resultado obtenido. 90 alimentos

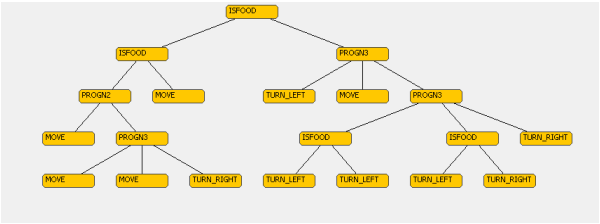

Figure 5: Grafo del mejor cromosoma.

Figure 6: Gráficas de evolución.