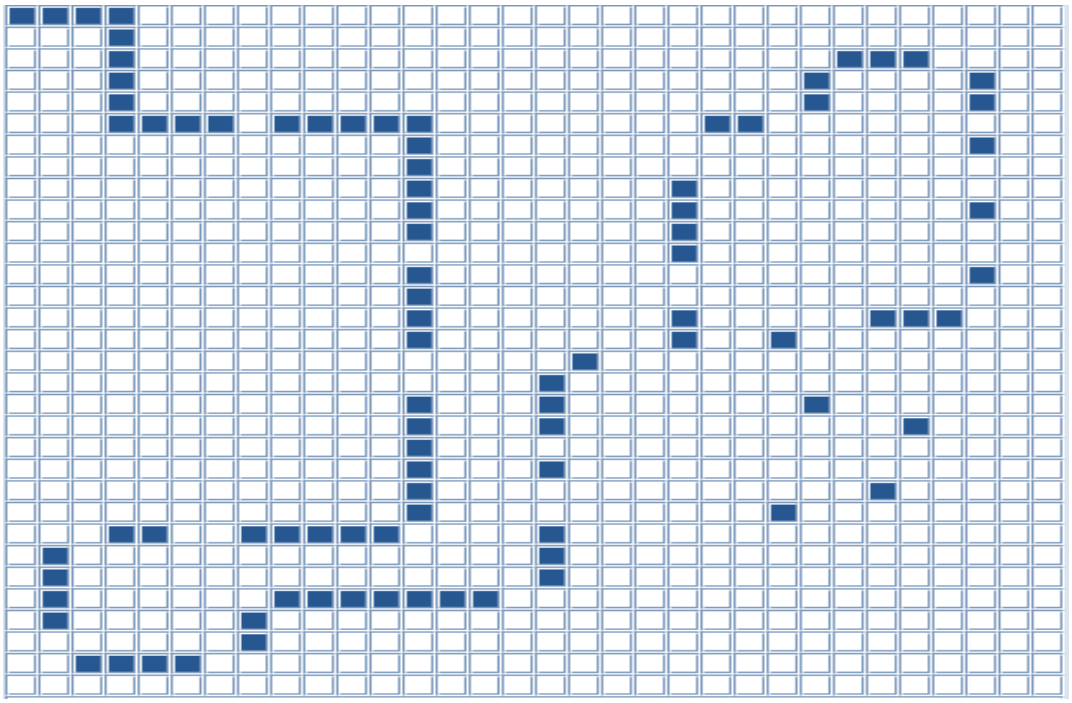
Facultad de Informática

Programación Evolutiva
Curso 2024/2025

Práctica 3: Programación genética/Gramáticas Evolutivas

Parte A: Programación Genética (8 puntos)

La Parte A de la práctica consiste en resolver mediante Programación genética el problema de la hormiga que se mueve por un tablero buscando comida. La hormiga debe ser capaz de encontrar toda la comida situada a lo largo de un rastro irregular. Dicha hormiga se mueve en un tablero (toroidal) de 32 x 32 celdas. La hormiga comienza en la esquina superior izquierda del tablero, identificada por las coordenadas (0,0), y mirando en dirección este. El modelo de rastro propuesto, con el que se han realizado diversos estudios, se conoce como "rastro de Santa Fe", y tiene una forma irregular compuesta de 89 "bocados" de comida. El rastro no es recto y continuo, sino que presenta huecos de una o dos posiciones, que también pueden darse en los ángulos.



Nuestro problema requiere disponer operaciones que permitan a la hormiga moverse hacia delante, girar a uno u otro lado y detectar comida a lo largo de rastro irregular. El conjunto de operandos (terminales) para este problema es:

Terminales = {AVANZA, DERECHA, IZQUIERDA}

donde **AVANZA** mueve la hormiga hacia delante en la dirección a la que mira en ese momento, **DERECHA** gira la hormiga 90° a la derecha, e **IZQUIERDA** la gira 90° a la izquierda.

Las funciones disponibles son:

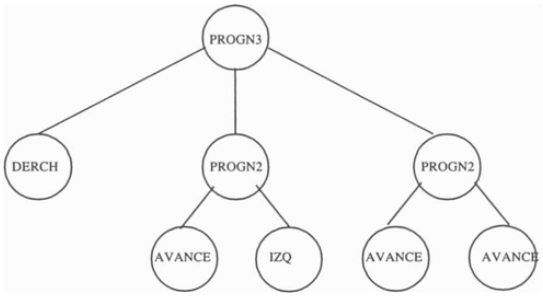
Funciones = { SICOMIDA(a,b) , PROG2 (a,b) , PROG3 (a,b,c) }

En el conjunto de funciones incluiremos también conectivas que fuerzan la ejecución de sus argumentos en un determinado orden. PROGN2 y PROGN3, que toman dos y tres argumentos respectivamente. Por lo tanto disponemos de las siguientes funciones:

- **SICOMIDA(a,b) :** la función **SICOMIDA (a,b)** toma dos argumentos y ejecuta **a** si se detecta comida delante y **b** en otro caso.
- **PROG2(a,b)** : Conectiva para encadenar dos acciones. Ejecuta sus dos argumentos secuencialmente y devuelve el resultado de la última evaluación. Evalúa **a**, luego **b**, y devuelve el valor de **b**.
- **PROG3(a,b,c)** : Conectiva para encadenar tres acciones. Ejecuta sus tres argumentos y devuelve el resultado de la última evaluación. Evalúa **a**, **b** y luego **c**, devolviendo el valor de **c.**

Un ejemplo de programa:

```
(PROG3 (DERECHA)
       (PROG2  (AVANZA) (IZQUIERDA))
       (PROG2  (AVANZA) (AVANZA))
```



El objetivo final es “comer” los 89 bocados mediante la ejecución de un programa o estrategia. La hormiga termina su trabajo cuando ha comida todos los bocados o cuando ha ejecutado 400 pasos (cada operación de movimiento o giro es un paso).

Procedimiento de resolución:

- Se genera una población inicial de programas aleatorios (árboles) usando el conjunto de funciones y terminales posibles. Estos árboles deben ser sintácticamente correctos. Limitamos la profundidad mínima a 2 y se debe poder modificar desde la interfaz). También se debe poder elegir el método de inicialización: **Creciente**, **Completa** y **Ramped and half**.
- **Fitness**: será la cantidad de alimento comido por la hormiga dentro de un espacio de tiempo razonable al ejecutar el programa a evaluar. Se considera que cada operación de movimiento o giro consume una unidad de tiempo. Limitaremos el tiempo a 400 pasos. El tablero se va actualizando a medida que desaparece la comida.
- Operador de Cruce: Intercambiar dos subárboles (elegidos aleatoriamente) entre los dos árboles padres.
- Operadores de Mutación: **al menos 4 de los vistos en el material de la asignatura**.
- Algún método de controlar el **bloating** para evitar programas muy largos
- Se mostrarán las gráficas de evolución y la expresión final obtenida, junto con su fitness.
- Al final de la ejecución se mostrará el recorrido realizado por la hormiga.

Parte B: Gramáticas Evolutivas (2 puntos)

- Resolver el mismo problema utilizando **Gramáticas Evolutivas**.
- Se utilizarán los operadores tradicionales sobre cromosomas de enteros.
- Se podrá seleccionar el número de **wraps** desde la interfaz.
- Se ha de definir una gramática adaptada al problema.

Normas de Entrega

- **Plazo de entrega: 25 de abril a las 16:00**. Se debe entregar mediante la tarea de entrega del Campus Virtual un archivo comprimido con el código java de la aplicación (**proyecto en Eclipse** o **NetBeans)** que incluya una breve memoria que contenga el estudio de las gráficas y los resultados obtenidos con cada función. Aquí se valorarán las conclusiones y observaciones que se consideren interesantes respecto al resultado obtenido.
- No olvidéis nombrar correctamente el proyecto e incluir en el código todas las librerías necesarias. El archivo comprimido y el nombre del proyecto Eclipse tienen que ser **GXXP3**, donde XX es el número de grupo.
- Debes incluir una memoria similar a las entregas anteriores e incluir al final de la memoria una breve descripción del **reparto de tareas** para reflejar lo que ha hecho cada miembro del grupo.
- Debes incluir en la memoria un ejemplo de ejecución (con estadísticas y gráficas) de cada uno de los dos problemas anteriores utilizando el entorno HeuristicLab:

  http://dev.heuristiclab.com/

- La evaluación de la práctica se complementará con un test de evaluación en laboratorio el día **miércoles 30 de abril entre las 14:00 y las 16:00 h.**

Importante:

- Recuerda que la calificación de las prácticas obligatoria (**P1, P2 y P3**) supone un **80 %** de la nota total de la asignatura.
- El **20%** de nota por actividad adicional se puede obtener mediante **una combinación** de actividades extra:
  - ✓ Realización de un trabajo-presentación o demo sobre algún tema de los propuestos en el **Tema 8 de la asignatura** o sobre **cualquier algún artículo científico** relacionado con la materia de la asignatura.
  - ✓ Resultados obtenidos en Kahoot.
  - ✓ Calificación de los ejercicios enviados como tareas.