Universidad de Burgos

Computación Neuronal y Evolutiva

P4: AirTafficController

Analizar y desarrollar un algoritmo genético que realice el cálculo automático de la distribución más conveniente de vuelos que solicitan aterrizar en un determinado aeropuerto.

Estudiantes:

David Miguel Lozano
Javier Martínez Riberas

Profesor de la asignatura:

Bruno Baruque Zanón

1º semestre 2016

Índice

A. Introducción

El objetivo de esta práctica es desarollar un programa que implemente un algoritmo genético que permita al usuario realizar el calculo automático de la distribución más conveniente de vuelos que solicitan aterrizar en un determinado aeropuerto. Así mismo, se analizará la conveniencia de la solución propuesta y se reflexionará sobre los resultados obtenidos.

La aplicación tiene que ser capáz de adaptarse a las siguientes condiciones:

- El número de pistas del aeropuerto debe ser un parámetro configurable.
- El número de aviones en cada situación puede variar.
- Los aviones tienen un programa de vuelo que incluye:
  - ETA (*Estimated Time of Arrival*): tiempo estimado de llegada calculado en el momento del despegue.
  - Tipo de avión: heavy / big / small. Condiciona el tiempo necesario para su aterrizaje.

El programa tiene que conseguir obtener de forma automática la mejor asignación posible de vuelos a aterrizar en cada pista, de forma que el tiempo de espera de los vuelos en su conjunto sea el menor posible.

Para su resolución se hará uso de la librería para Java JCLEC (Java Class Library for Evolutionary Computation) [1]. La cual, proporciona un framework para programación evolutiva que da soporte, entre otras cosas, a los algoritmos genéticos.

B. Solución propuesta

A continuación se detalla la codificación y configuración del problema para ser resuelto con un algoritmo genético.

B.1. Representación de los individuos

Cada individuo representa los diferentes aviones que desean aterrizar en el aeropuerto en una situación determinada. Cada individuo se ha codificado como un genotipo binario, su longitud es igual al número de aviones y está ordenado de acuerdo al orden de aterrizaje.

Por ejemplo, dado el siguiente genotipo:

$$\begin{bmatrix} 2 & 3 & 1 & 4 \end{bmatrix}$$

Se representa que el primer vuelo en aterrizar fue el 2, seguido del 3, 1 y 4.

B.2. Individuos no factibles

Los individuos que no cumplen los requisitos son tratados en la fase de evaluación, donde se les valora negativamente el fitness. De esta manera, nos aseguramos que no forman parte de la siguiente generación.

B.3. Esquema evolutivo

Se ha utilizado el algoritmo SGE (*Simple Generational and Elitist*). Se trata de un algoritmo elitista que asegura que sólo los mejores individuos pasen a la siguiente generación en cualquier momento.

B.4. Función de fitness

La función de fitness se ha calculado el sumatorio de los retrasos de todos los vuelos. Entendiendo por retraso la diferencia entre el ATA de cada avión y su mínimo ETA.

B.5. Inicialización

La población inicial se inicializa de forma aleatoria.

B.6. Criterio de parada

El criterio de parada se ha establecido en 1.000 generaciones.

B.7. Criterio de selección

Para seleccionar un subconjunto de la población se han analizado los siguientes algoritmos:

1. **RouletteSelector**: selección por ruleta.

B.8. Operador de cruce

Para obtener un nuevo individuo basado en el genotipo de sus padres se han analizado los siguientes algoritmos:

1. **OnePointCrossover**: cruce monopunto.

B.9. Operador de mutación

Cada gen del genotipo de un individuo tiene una cierta probabilidad de mutar. Se han analizado los siguientes algoritmos de mutación:

1. **OneLocusMutator**: TODO.

B.10. Criterio de reemplazo

Los hijos reemplazan directamente a los padres. Para preservar el elitismo, si la mejor sulución de la generación anterior no sobrevive, la peor solución se reemplaza por una nueva.

C. Análisis

A continuación exponemos los resultados obtenidos y realizamos un análisis de estos.

TODO

D. Conclusiones

TODO

Referencias

[1] Sebastián Ventura, Cristóbal Romero, Amelia Zafra, Jose Antonio Delgado, and César Hervás. JCLEC - java class library for evolutionary computation, 2008. URL http://jclec.sourceforge.net/.