PRÁCTICA DE ALGORITMOS GENÉTICOS

Manuel Blanco Rienda

INTELIGENCIA COMPUTACIONAL Práctica 2

Indice

1. Descripción de la práctica realizada

La práctica sobre algoritmos genéticos de la asignatura "Inteligencia Computacional" ha consistido en la implementación de tres algoritmos distintos de aplicación sobre el problema de la asignación cuadrática (QAP), en un contexto de ejecución relacionado con la posición a ocupar de diferentes máquinas dentro de una fábrica, de cara a obtener los mejores resultados con la mínima inversión.

a. Descripción del algoritmo estándar

El algoritmo genético estándar que se ha implementado para la práctica es de muy sencillo comportamiento: indicando el número de individuos de la población y el número de generaciones que queremos ejecutar el algoritmo, se crea una población inicial aleatoria de individuos, tras lo cual se pasa a evaluar cuán buena es la solución que representa cada uno a través de su "fitness". Tras ello, se selecciona mediante torneo a diez individuos del conjunto, de los cuales se elige a los dos mejores, durante un número de iteraciones igual a la mitad del número de individuos (porque los vamos agrupando por parejas y darán parejas de niños al cruzarlos).

Una vez seleccionados los mejores de esta serie de torneos, se procederá a cruzarlos, utilizando como operador de cruce el llamado "cruce de orden". Por cada una de las parejas seleccionadas se producen dos niños, los cuales sustituirán a toda la población de la generación anterior excepto a los cinco mejores de la misma, utilizando un sistema híbrido (por edad y por elitismo) a la hora de decidir los supervivientes de cada generación.

Una vez tenemos a la nueva generación, se procede a mutarla, recorriendo cada gen de cada individuo de la misma (menos los cinco mejores que fueron mutados justo antes de ser seleccionados como tal) y mutando cada uno con una probabilidad de (1/número de genes de cada individuo). Tras esto, se pasa de nuevo a evaluar el fitness de toda la población (menos de los cinco mejores que no han cambiado) y el proceso itera, pasando a elegir los mejores del conjunto.

b. Descripción del algoritmo Baldwiniano

El algoritmo "Baldwiniano" funciona exactamente igual que el estándar, a excepción del proceso de evaluación del fitness. En esta variación del algoritmo estándar, se utiliza una metodología greedy, por ascensión de colinas según la cual: partiendo de la configuración genética original de cada individuo se busca cuál es su mejor configuración para obtener el menor fitness posible, y ese fitness es el que se le asigna. Sin embargo, el individuo mantiene sus genes como estaban originalmente. Esta variación permite seleccionar de mejor forma, padres más prometedores de cara a producir mejores hijos, aunque los fitness que tienen no sean "reales", sino producto de estimaciones.

c. Descripción del algoritmo Lamarckiano

La tercera versión del algoritmo genético, sólo varía en una cosa con respecto al "Baldwiniano", a la hora de evaluar el fitness de cada individuo, después de haber utilizado la metodología greedy para obtener la mejor configuración genética del mismo, ésta se aplica sobre el individuo, modificándolo. El cruce por tanto se produce con individuos modificados para obtener las mejores soluciones posibles.

2. Descripción técnica de la práctica

La práctica se ha desarrollado en el IDE Eclipse, utilizando el lenguaje Java y consta de tres clases: un enumerado para diferenciar entre los tres tipos de algoritmos a usar, una clase "Main" que contiene todo lo necesario para leer los datos de las matrices de los conjuntos de prueba y para ejecutar generación tras generación los algoritmos descritos a través del uso de la clase "Poblacion" que contiene toda la funcionalidad requerida para la representación del algoritmo genético estándar y sus variaciones. Como apunte, para entender cómo se ejecutan éstos desde el método main en la clase "Main", se procede a enumerar los pasos a dar:

- Se establece en la variable: "nombreDatos" la ruta y el nombre donde se encuentra el conjunto de prueba con el que se va a trabajar.
- Se establece en la variable: "tamanioPoblacion" el número de individuos de la población y por tanto de cada generación (ya que es de tamaño fijo).
- Se establece en la variable "numeroGeneraciones" el número de generaciones a lo largo de las que se va a ejecutar el algoritmo especificado. A continuación puede verse una captura del código del método main (dentro de la clase "Main"), donde puede hacerse todo lo anterior:

```
//Nombre del paquete de datos de prueba a usar
String nombreDatos = "src/datos/tai256c.dat";
tamanioPoblacion = 100;
numeroGeneraciones = 100;
ArrayList<Pair<Integer,Integer>> padres;
////////////////////////////////////////////////////
```

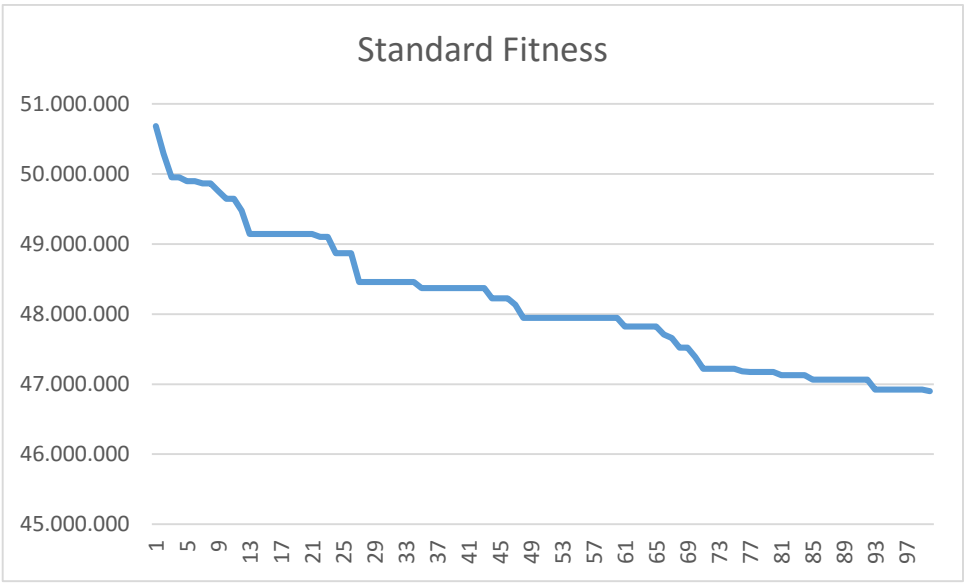
- Se especifica qué algoritmo se va a usar, a través de la clase enumerado que se ha definido arriba, utilizando los siguientes valores para el mencionado atributo a la hora de crear la población: "Standard", "Lamarck" y "Baldwin", tal y como puede verse a continuación:

```
//Se crea la población
Poblacion p = new Poblacion(N,tamanioPoblacion,matrizDistancias,matrizPesos,TipoAlgoritmo.Baldwin);
```

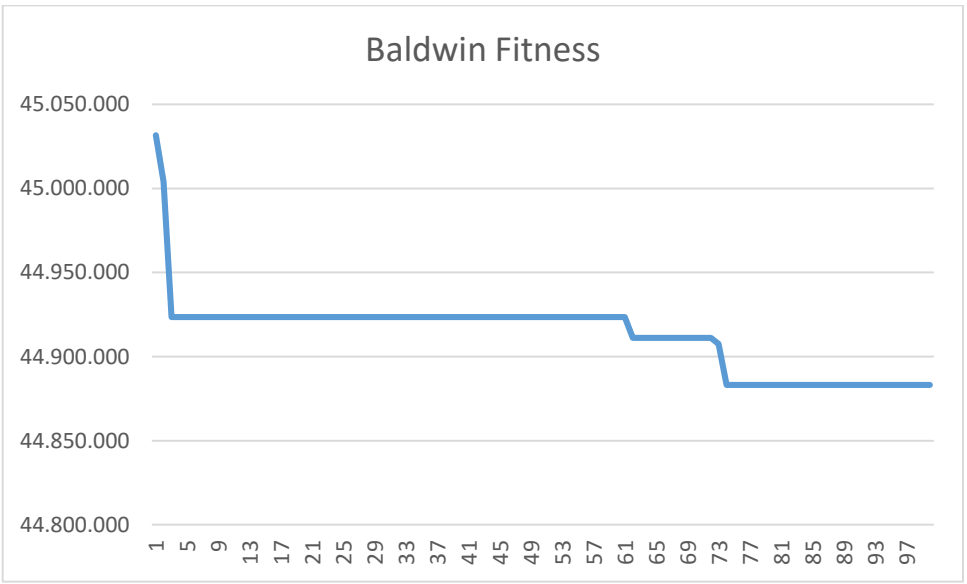
Hay que remarcar el hecho de que para representar el vector de fitness se ha utilizado un "ArrayList" de pares Entero-Flotante, según el cual, cada elemento es una pareja en la que el primer elemento apunta a un individuo en la población y el segundo es su fitness.

3. Análisis de los resultados obtenidos (100 individuos)

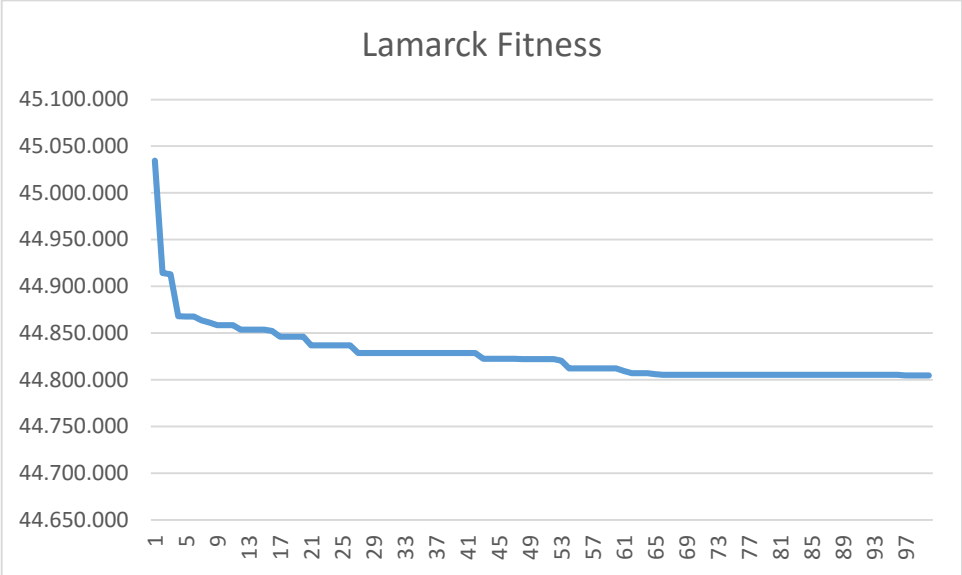
A continuación pueden verse las gráficas correspondientes a los resultados obtenidos con los tres algoritmos para una población fija de 100 individuos (de ahora en adelante todas las pruebas se realizan a lo largo de 100 generaciones):



Vemos cómo el algoritmo estándar mejora constantemente, cada muy pocas generaciones, sin apenas estancarse. Su ejecución es muy rápida (dentro del contexto de ejecución: i7-3770k y 8 GB de RAM), dado que apenas tarda 1 segundo para cada 5 o 6 generaciones. Obtiene el resultado final mínimo de un fitness de 47.129.268.



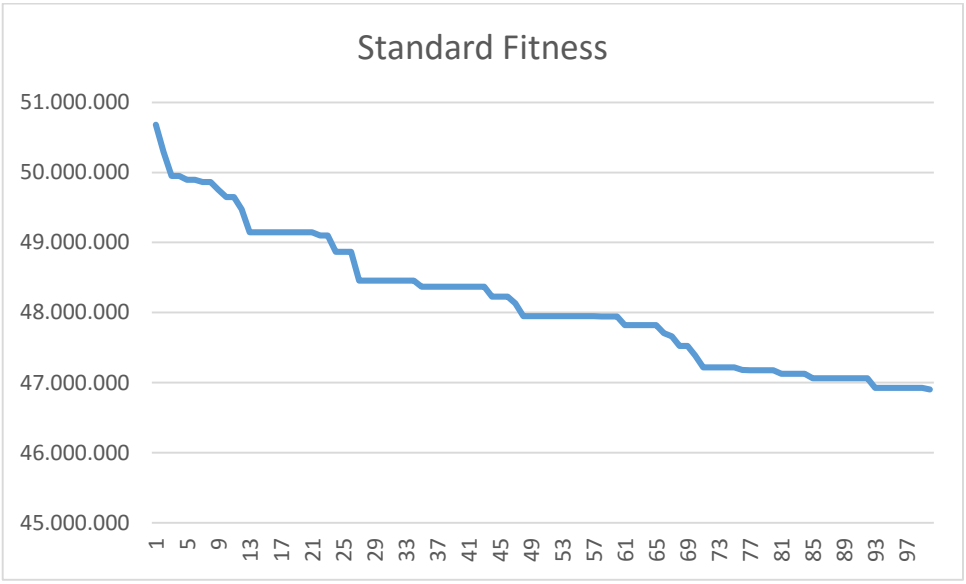
Por otro lado, vemos como el algoritmo de Baldwin mejora rápidamente cuando lo hace, pero está muchas generaciones estancado, cuando lo hace. Es más lento que el estándar, dado que tarda unos 20 segundos para cada generación. Obtiene un resultado final de fitness de: 44.902.096.



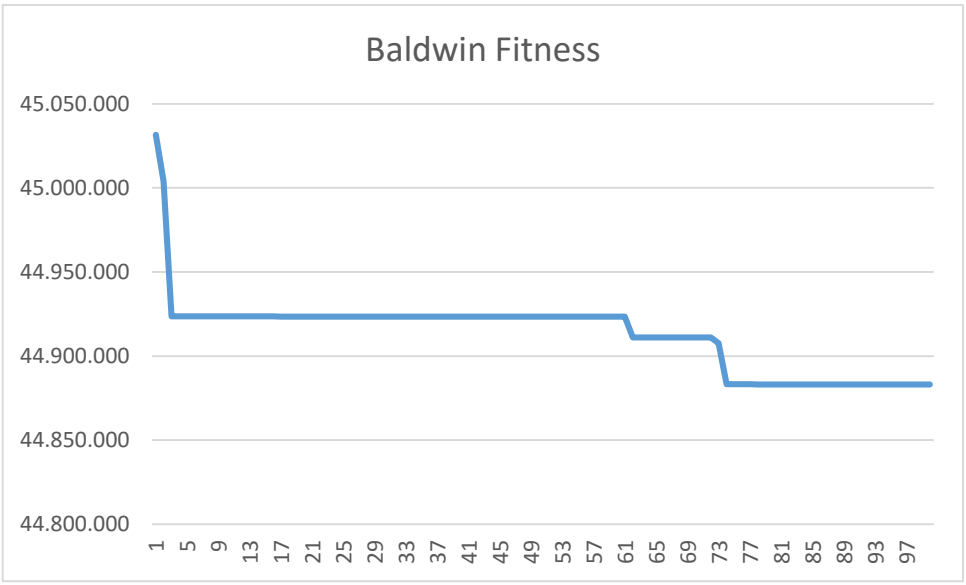
Por último en Lamarck vemos como se mejora rápidamente al comienzo (al igual que en Baldwin), no estancándose casi nada durante su tiempo de ejecución, hasta llegar a su solución óptima. Su velocidad de ejecución es más lenta que el Baldwin (unos 25 o 30 segundos para cada generación). Obtiene el resultado final de fitness de: 44.831.016.

4. Análisis de los resultados obtenidos (256 individuos)

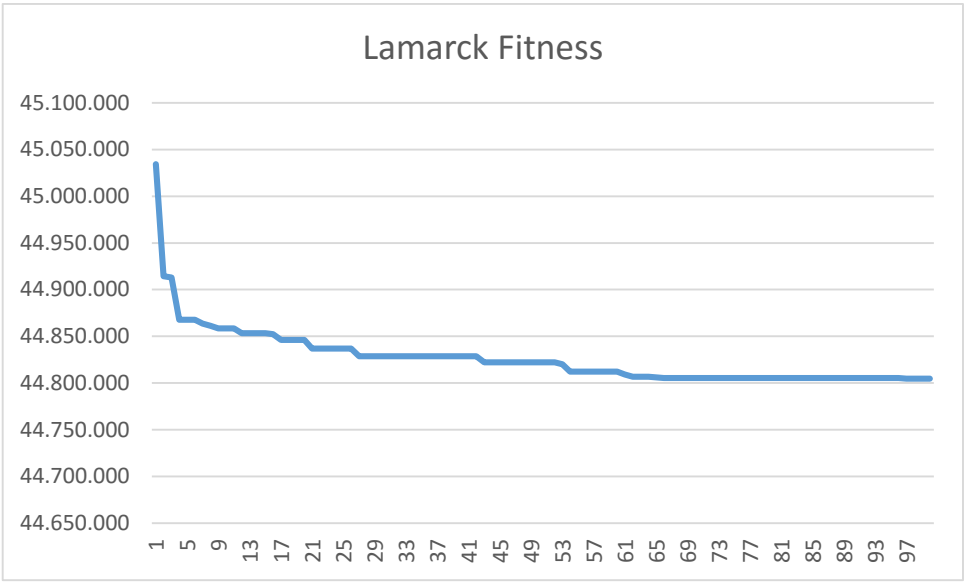
Con la premonición de obtención de mejores resultados (ya que las ejecuciones anteriores lo presagiaban), probé a ejecutar poblaciones de 256 individuos sobre los tres algoritmos, dando los siguientes resultados:



En la gráfica podemos ver cómo al igual que con 100 individuos, con 256 el algoritmo estándar mejora cada pocas generaciones de forma casi constante, tardando un poco más que con 100 individuos (algo normal) y obteniendo el resultado final de fitness de: 46.900.176.



En Baldwin, al igual que en estándar obtenemos el mismo comportamiento que con 100 individuos, pero tardando un poco más por generación, y obteniendo el resultado final de fitness de: 44.883.104.



En Lamarck ocurre exactamente lo mismo que en los anteriores casos en cuanto a su evolución con 256 individuos, tardando un poco más por generación que su homónimo y obteniendo el resultado final de fitness de: 44.804.736.

5. Conclusiones

Tras analizar los resultados descritos en el anterior apartado, podemos llegar a la conclusión de que con poblaciones mayores (en cuanto a individuos) obtenemos mejores resultados pese a tener que esperar más tiempo por generación. Por otro lado, es evidente la escala en cuanto a resultados más prometedores que ofrecen los tres algoritmos descritos: Standard, Baldwin, Lamarck (de peor a mejor fitness obtenible). Analizando cada uno por su lado, se puede ver que tanto Standard como Lamarck mejoran casi constantemente generación tras generación, mientras que Baldwin se suele estancar durante un gran número de generaciones.

Además, es con Lamarck con quien obtenemos los mejores resultados en un menor número de generaciones, ya que Standard obtiene su solución final casi al terminar las cien generaciones, mientras que Baldwin lo hace un poco antes que Standard y después de Lamarck. Hay que remarcar que no hay apenas diferencias entre los comportamientos que ofrecen los algoritmos para 100 o 256 individuos.

Por otro lado, la mejor solución a la que se ha llegado con Lamarck (que ha sido la mejor de todas las obtenidas) es 44.804.736. Teniendo la probabilidad de mutación bastante baja (1/número de genes por individuo), es posible que si se subiera obtendría mejores resultados. Sin embargo esto no ha podido ser demostrado por falta de tiempo, aunque se tendrá en cuenta para el futuro.