Francisco Serrano Carmona – 75165929 C – IC: Práctica 2

Implementación

Para esta práctica se ha implementado una estructura para resolver problemas usando Algoritmos Evolutivos. Se ha usado Java como lenguaje de implementación. A continuación se explica la estructura seguida para implementar el programa:

Clases usadas en la implementación:

- **IndividuoAbstracto**
  Un individuo es una solución del problema genético, con un "cromosoma" solución determinado. Esta clase es una clase abstracta (clase que no puede instanciarse, y ha de implementarse en otra clase) que representa los Individuos de una población en un algoritmo evolutivo. Los métodos que contiene esta clase son:
  - Fitness: calcula el Fitness del Individuo.
  - Aprender: método que realiza un algoritmo (normalmente greddy) el cual cambia la solución para optimizarla ("adaptarla" al problema para buscar un óptimo).
  - Fitness Original: Si la solución "aprende" (con el método anterior), este valor guarda el Fitness que tenía el individuo antes de realizar el aprendizaje.
  - Cruzar: Este método devuelve un nuevo Individuo tras realizar el cruce entre el individuo instanciado y otro individuo pasado por parámetro.
  - Comparar: comparar el fitness de este invidivuo con el fitness de otro individuo para poder ordenar individuos dentro de una población, a partir de su adaptación al medio (fitness).
  - Representación: Es el objeto que guarda los datos de la solución del problema a representar.
  - Generar Aleatorio: Genera un individuo aleatorio, para la generación inicial de individuos de una población.

- **PoblacionAbstracta**
  Una población es un grupo de individuos que van evolucionando en diversas generaciones (iteraciones) para conseguir una solución optimizada del problema propuesto. Esta clase abstracta (clase que no puede instanciarse, y ha de implementarse en otra clase) representa una población de individuos, y contiene todo lo necesario para implementar el algoritmo evolutivo. Está compuesto de los siguientes métodos y atributos:
  - Atributos:
    - Tipo: guarda el tipo de evolución que se utiliza (1 = evolución estándar; 2 = evolución Baldwiniana; 3 = evolución Lamarkiana).
    - Probabilidad de mutación: guarda la probabilidad (entre 0 y 1) de que un individuo mute en una iteración.
    - Probabilidad de cruce: guarda la probabilidad (entre 0 y 1) de que un individuo se cruce con otro individuo en una iteración.
    - Fracción de selección de la población: guarda qué fracción de la población debe de sobrevivir en una selección. Por ejemplo, si la fracción es 4, sobrevivirá ¼ (25%) de la población.
  - Métodos:
    - Seleccionar el mejor de la última iteración: selecciona el individuo con mayor fitness de la última generación de la población (para mostrar la mejor solución hasta el momento).

        - Iterar: Realiza la iteración de una generación. Una iteración consiste en:
            1. Si el tipo de algoritmo evolutivo es Baldwiniano ó Lamarkiano, los individuos de la generación actual “aprenden” (se adaptan al medio).
            2. Se realiza la “selección” de individuos (donde se seleccionan los individuos que mejor se adaptan al medio).
            3. Se crea una nueva generación, cruzando y mutando los individuos seleccionados.
        - Seleccionar: método que implementa la selección de individuos (paso 2 de la iteración).
        - Cruzar y Mutar: método que implementa el cruce y mutación de los individuos de una generación (paso 3 de la iteración).

- **Representación**
  Es la clase que guarda los datos para la representación del problema y la solución, para ser integrada en los individuos del algoritmo evolutivo. Para nuestro problema concreto, se han usado los siguientes atributos:
    - Solución: guarda una lista de enteros (posiciones en el problema) para representar la solución al problema.
    - Solución Primitiva: si el individuo “aprende”, este atributo guarda la Solución que tenía el individuo cuando se creó.
    - Distancias: Guarda la matriz de distancias del problema.
    - Flujos: Guarda la matriz de flujos del problema.

- **Individuo**
  Es la implementación para este problema concreto de la clase IndividuoAbstracto. Se implementa con la Representación explicada anteriormente. Los métodos que se han implementado han sido:
    - Calcular Fitness: calcula el fitness según lo indicado en el enunciado de la práctica como función de coste.
    - Función de aprendizaje: implementa un algoritmo greddy basado en 2-opt para el problema QAP, con un límite de “Profundidad” del algoritmo de 5 niveles (si se dejaba que encontrara la solución óptima, el algoritmo tardaba demasiado en finalizar y no merecía la pena dada la optimización).
    - Cruzar: Implementa el operador de cruce para el problema. Para ésta práctica se ha implementado un cruce de tipo “1-point crossover” (cruce de un punto), con selección aleatoria del punto de cruce en cada cruce que se haga.
    - Mutar: Implementa el operador de mutación para el problema. Para esta práctica se ha implementado una mutación del tipo “Intercambio”, donde elige aleatoriamente 2 genes del cromosoma y los intercambia.

- **Población**
  Es la implementación para este problema concreto de la clase PoblacionAbstracta. Se implementa con la clase Individuo explicada anteriormente. Los métodos que se han implementado han sido:
    - Seleccionar: Implementa el modelo de selección del problema. Para este problema se ha elegido un modelo de tipo “Elitista”: Ordena a los individuos por su “Fitness” y sobreviven los individuos indicados según la Fracción de Selección de la Población.
    - Cruzar y Mutar: Implementa el modelo de cruce y mutación del problema. Para este problema, se ha elaborado el siguiente modelo:
        - Los padres seleccionados pasan a la siguiente generación.
        - Estos padres, con la probabilidad de cruce indicada en el atributo, se cruzan entre

ellos (si no se cruzan, se duplica el segundo padre del segundo cruce).
        - El hijo creado mutará (con la probabilidad indicada en probabilidad de mutación) utilizando el operador de mutación.
        - Se crearán nuevos hijos hasta tener tantos individuos como la generación anterior.

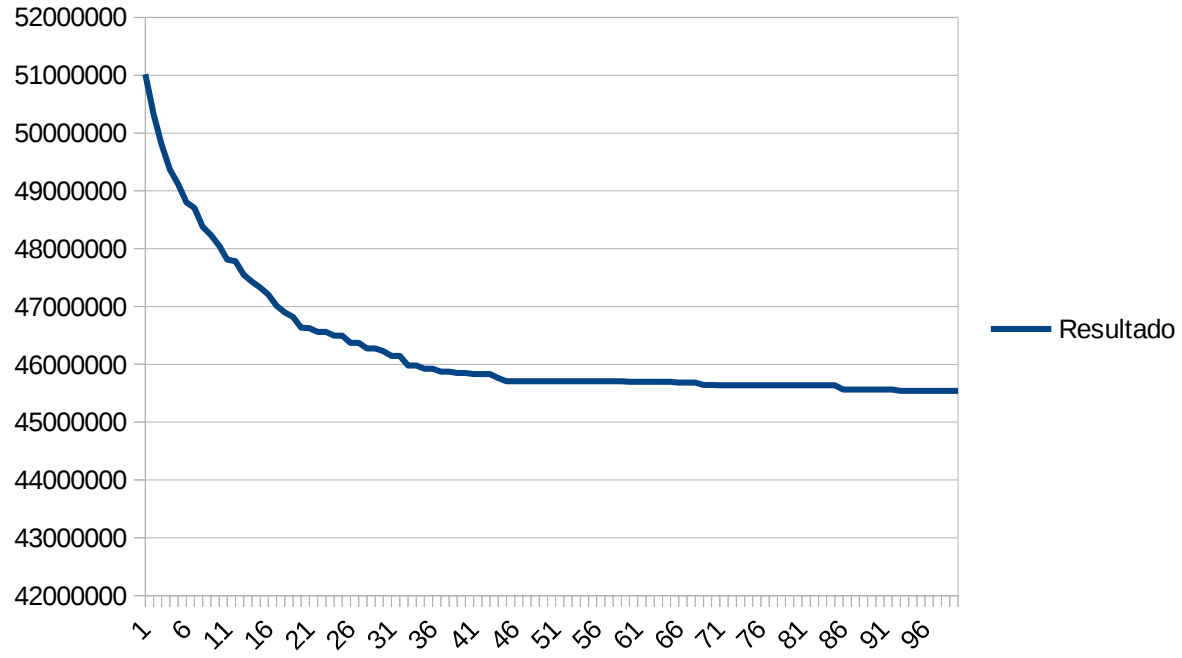
- **Principal**
    - Esta clase implementa el método “Main” del programa. En esta clase se lee el archivo de datos del problema, se generan los individuos de la primera generación (aleatoriamente) y se hacen las iteraciones de la población, mostrando los resultados.

Resultados

Dada la implementación descrita, se han obtenido los siguientes resultados con diferentes opciones probadas:

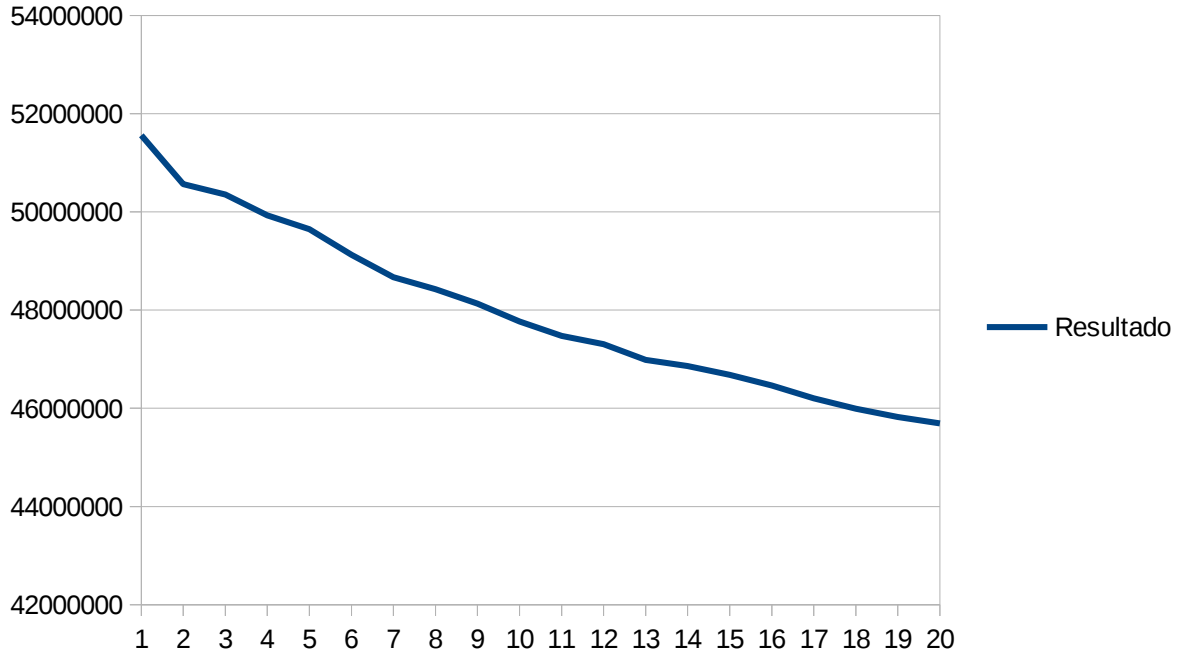
- Resultado 1:
    - Datos:
        - Tipo de Algoritmo: 1 (normal).
        - Tamaño de Población: 500.
        - Número de Iteraciones: 100.
        - Probabilidad de Mutación: 0,3.
        - Probabilidad de Cruce: 0,6.
        - Factor de Población para Selección: 30 (500/30 = 17 mejores individuos seleccionados).
    - Resultados:
        - Fitness de la Mejor Solución: 45540808.
        - Solución: (249 161 113 26 114 53 176 92 241 65 218 119 153 198 81 78 45 252 36 127 156 80 152 130 64 231 27 222 3 206 170 220 244 8 191 230 79 146 68 147 202 150 221 17 132 14 18 149 165 239 9 174 189 57 240 59 237 83 248 138 160 94 254 77 232 1 167 46 190 144 63 247 75 129 195 70 24 236 71 185 210 188 226 203 76 52 159 115 118 135 33 102 250 229 251 51 23 154 30 54 162 124 31 133 50 171 137 82 106 67 173 172 166 62 107 105 37 228 208 42 211 6 224 215 5 246 13 238 10 196 213 141 121 184 61 151 142 128 56 235 110 74 125 48 183 233 19 55 155 38 193 2 212 58 177 90 223 253 100 164 87 182 103 157 97 116 20 180 34 181 99 256 88 145 73 245 243 169 4 214 39 242 179 85 95 219 28 29 225 47 143 32 91 158 139 7 204 227 21 205 49 194 98 96 35 134 187 234 199 25 126 175 22 122 120 60 140 123 43 197 136 40 112 89 178 101 207 16 255 72 109 186 86 192 41 131 12 93 84 117 15 148 44 200 201 209 11 108 163 216 217 69 111 66 104 168)
        - Tiempo de Ejecución: 182065 ms.

- Evolución de Resultados por Iteración:



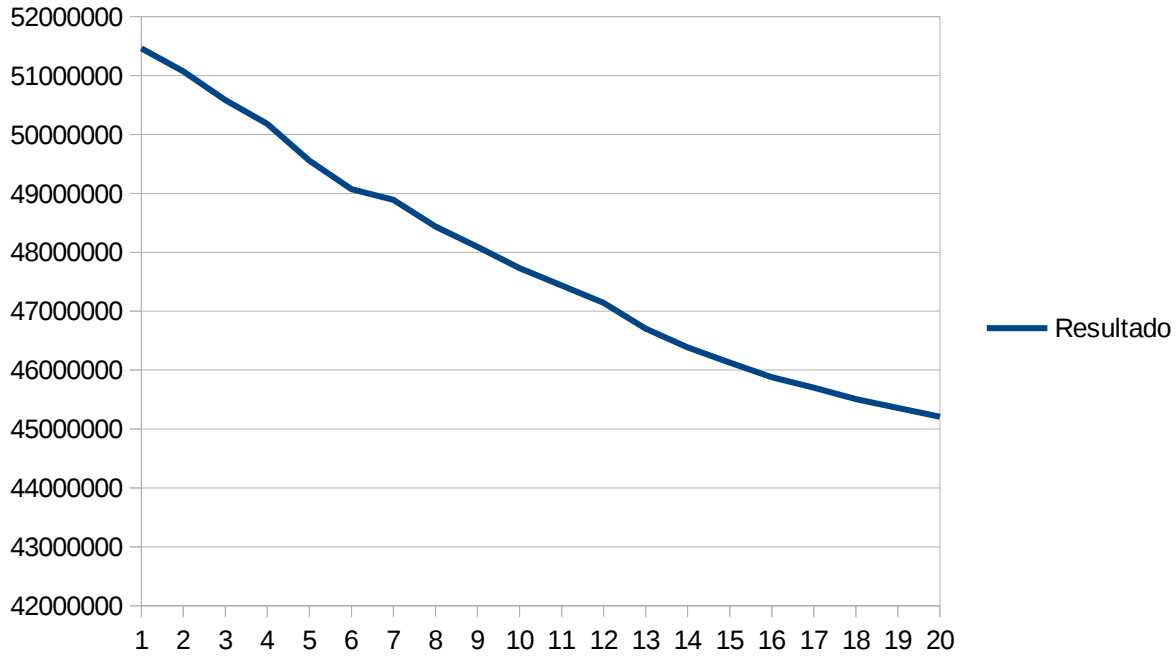
- Resultado 2:
  - Datos:
    - Tipo de Algoritmo: 2 (Baldwiniano).
    - Tamaño de Población: 30.
    - Número de Iteraciones: 20.
    - Probabilidad de Mutación: 0,1.
    - Probabilidad de Cruce: 0,6.
    - Factor de Población para Selección: 3 (30/3 = 10 mejores individuos seleccionados).
  - Resultados:
    - Fitness de la Mejor Solución: 45692334.
    - Solución: (49 195 34 235 61 115 144 38 186 242 54 116 245 74 172 119 224 36 141 220 193 246 65 126 108 82 129 83 9 114 252 29 15 156 253 69 148 2 187 89 130 113 55 138 213 125 4 214 171 135 87 238 78 146 86 200 176 79 183 191 76 184 256 124 71 185 189 90 99 194 149 70 237 68 122 6 254 222 44 147 151 16 101 212 48 167 21 218 45 223 121 39 199 43 227 40 225 67 177 57 239 8 168 11 204 229 56 201 150 240 58 251 207 145 165 152 19 236 160 164 92 250 134 59 93 28 192 72 52 159 81 188 180 13 226 142 97 1 140 143 41 234 216 241 248 155 118 37 170 120 255 53 98 169 162 22 205 64 219 103 105 50 221 232 30 117 51 127 85 211 32 208 158 217 3 173 136 247 244 77 131 46 196 157 154 10 231 179 24 175 209 14 88 230 60 197 75 206 20 139 18 132 62 104 166 84 66 102 233 47 249 63 112 73 123 25 161 23 163 174 27 243 106 137 12 111 35 198 153 215 100 42 178 96 94 17 203 5 110 128 95 80 190 31 107 7 228 109 33 182 133 91 202 181 26 210)
    - Tiempo de Ejecución: 70993 ms.

- Evolución de Resultados por Iteración:



- Resultado 3:
  - Datos:
    - Tipo de Algoritmo: 3 (Lamarkiano).
    - Tamaño de Población: 30.
    - Número de Iteraciones: 20.
    - Probabilidad de Mutación: 0,1.
    - Probabilidad de Cruce: 0,6.
    - Factor de Población para Selección: 3 (30/3 = 10 mejores individuos seleccionados).
  - Resultados:
    - Fitness de la Mejor Solución: 45210890.
    - Solución: (41 144 87 153 165 89 235 11 238 104 81 101 43 126 187 208 132 24 143 78 129 226 26 148 95 15 123 34 110 142 4 227 74 240 215 174 72 161 131 14 168 219 55 210 23 195 256 252 242 230 52 102 159 25 182 124 39 216 98 38 247 35 200 47 2 99 224 106 13 120 134 61 214 66 190 169 27 166 122 147 172 57 138 91 244 103 22 154 68 231 211 88 186 127 58 177 201 53 181 84 155 46 139 64 152 245 42 202 206 18 150 48 217 203 59 205 251 175 50 94 149 21 171 62 128 158 17 137 20 112 157 246 7 198 243 90 249 145 156 164 121 10 115 136 196 8 248 86 253 79 96 254 40 237 30 209 31 146 100 51 5 111 65 97 167 119 184 77 179 33 173 54 176 69 250 204 192 67 140 116 19 191 9 163 1 135 6 241 225 199 3 113 36 228 70 255 220 76 114 73 223 109 117 107 49 178 218 151 194 45 236 83 125 105 63 180 71 183 56 130 185 85 108 82 162 229 28 212 213 16 222 118 29 170 193 92 133 93 60 221 239 37 141 32 232 207 75 188 189 44 197 234 233 80 160 12)
    - Tiempo de Ejecución: 475888 ms.

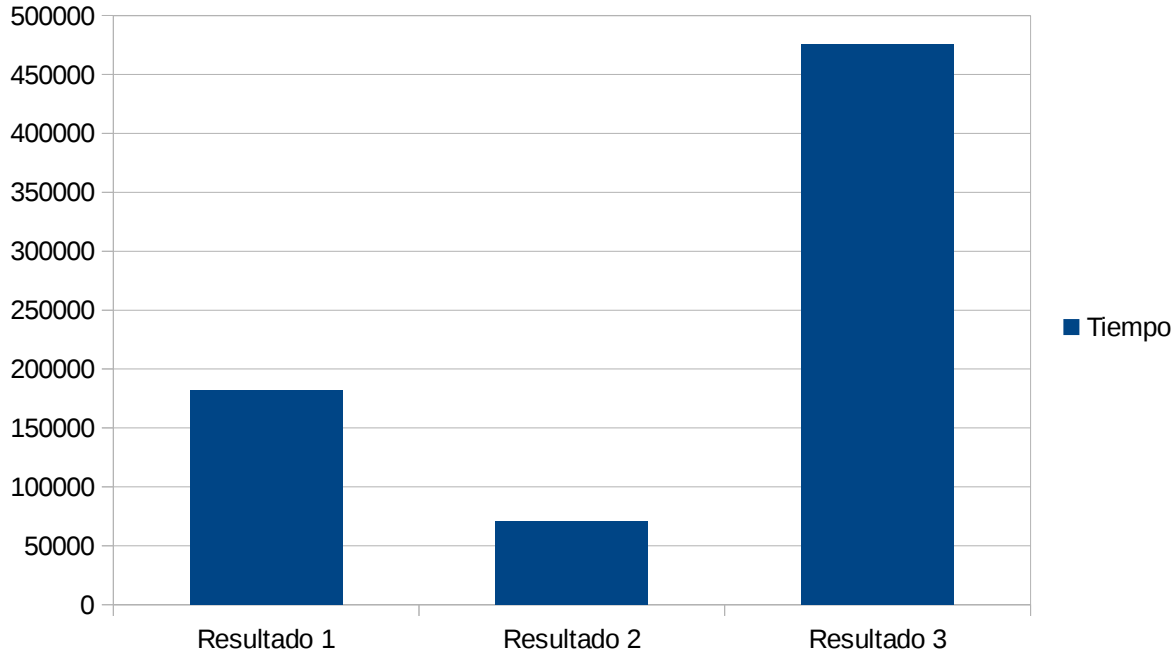
- Evolución de Resultados por Iteración:



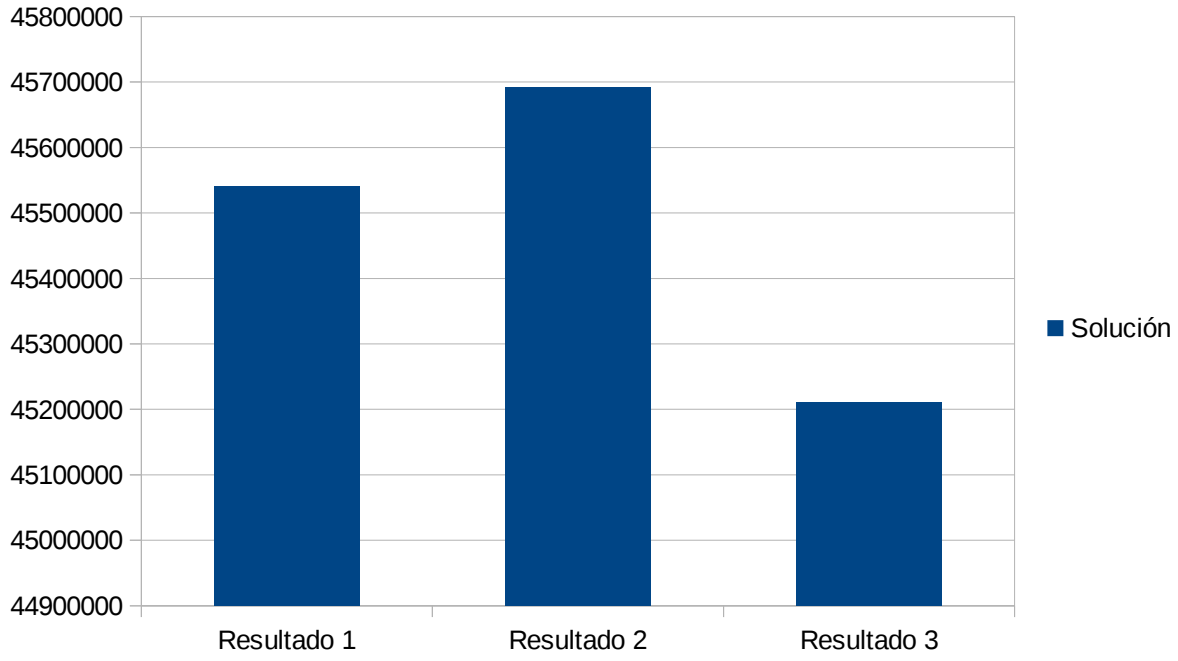
Comparativa de Resultados

A continuación se muestra una comparativa con los 3 resultados obtenidos:

- Comparativa de Tiempos:



- Comparativa de Fitness de Soluciones:



Conclusiones

Una vez realizado el estudio, se extraen las siguientes conclusiones:

- Comparados con el algoritmo evolutivo estándar, los algoritmos Lamarkiano y Baldwiniano pueden suponer una mejora de resultados (tanto en buscar la solución óptima como en tiempo), pero dependerá mucho de qué algoritmo de aprendizaje implementemos.
- Hay una mejora sustancial en la solución Lamarkiana respecto a la Baldwiniana, dado que las soluciones aprendidas se heredan, dando lugar a cruzar mejores individuos.
- En las soluciones Lamarkiana y Baldwiniana se pueden usar un menor número de generaciones y un menor número de individuos en la población par conseguir unos resultados iguales al algoritmo estándar. El algoritmo estándar tarda más en converger, puesto que no existe ninguna función de optimización que ayude al algoritmo.