UNIVERSIDAD DE GRANADA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

INTELIGENCIA COMPUTACIONAL

CURSO 2024 / 2025

Práctica 2: QAP

Trabajo realizado por:
Mario Martínez Sánchez

Índice

1. Algoritmo genético estándar.....3
1.1. Descripción del Algoritmo Genético..... 3
1.2. Parámetros configurados..... 3
1.3. Resultados obtenidos..... 4
1.4. Conclusiones..... 5
2. Algoritmo genético con variante baldwiniana (v1).....6
2.1. Descripción del Algoritmo Genético Baldwiniano..... 6
2.2. Parámetros configurados..... 7
2.3. Resultados obtenidos..... 7
2.4. Conclusiones..... 8
3. Algoritmo genético con variante baldwiniana (v2).....9
3.1. Descripción del Algoritmo Genético Baldwiniano..... 9
3.2. Parámetros configurados..... 10
3.3. Resultados obtenidos..... 10
3.4. Conclusiones..... 11
4. Algoritmo genético con variante lamarckiana.....12
4.1. Descripción del Algoritmo Genético Lamarckiano..... 12
4.2. Parámetros configurados..... 13
4.3. Resultados obtenidos..... 13
4.4. Conclusiones..... 14
5. Comparación de algoritmos.....15

1. Algoritmo genético estándar

En este caso, se implementó un algoritmo genético (AG) estándar para resolver el problema de asignación cuadrática (QAP). Este tipo de problema surge en numerosas aplicaciones del mundo real, como la optimización de diseños de redes, la organización de recursos en sistemas productivos y la distribución de instalaciones en plantas industriales, entre otras.

El algoritmo genético constituye una metodología metaheurística basada en la biología evolutiva, que utiliza técnicas como selección, cruce y mutación para generar soluciones aproximadas a problemas complejos. En este informe se describen detalladamente los componentes fundamentales del algoritmo genético diseñado, los resultados obtenidos durante las ejecuciones, y los parámetros configurados para lograr un balance adecuado entre exploración y explotación en el espacio de búsqueda.

1.1. Descripción del Algoritmo Genético

El algoritmo genético implementado consta de los siguientes componentes principales:

- **Representación:** La representación de las soluciones se realiza mediante permutaciones de enteros, donde cada elemento del arreglo representa una instalación asignada a una ubicación específica.
- **Mecanismo de selección:** Se emplea selección por torneo con un tamaño de torneo de 5. Este método selecciona aleatoriamente un subconjunto de soluciones y escoge la mejor según su aptitud.
- **Operadores genéticos:**
    - **Cruce:** Se utiliza el operador de cruce por orden (OX). Este operador garantiza que la descendencia herede la estructura de las soluciones padres sin duplicar elementos.
    - **Mutación:** Se aplica mutación por intercambio (swap) con una probabilidad de mutación configurada. Este operador selecciona aleatoriamente dos posiciones en la permutación y las intercambia.
- **Condiciones de parada:** El algoritmo ejecuta un número fijo de generaciones (500), tras el cual devuelve la mejor solución encontrada.

1.2. Parámetros configurados

Para la ejecución del algoritmo genético, se utilizaron los siguientes parámetros:

- Tamaño de la población: 1000
- Número de generaciones: 500
- Probabilidad de cruce: 0.9
- Probabilidad de mutación: 0.05
- Tamaño del torneo: 5
- Semilla aleatoria: 2024

1.3. Resultados obtenidos

El algoritmo fue evaluado con el conjunto de datos proporcionado (“tai256c.dat”), que incluye las matrices de flujos y distancias correspondientes. Los resultados obtenidos fueron los siguientes:

- **Mejor permutación encontrada:**

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

- **Coste asociado a la mejor solución:** 45200784
- **Tiempo empleado:** 17 minutos.

Captura de pantalla:

```
Mejor solución encontrada (permutación): [  7 172 202  20 227  46 161  38 116 200 176 142  40 218 215 121  13 154
 194  53  63  11 109  82 221  26 246  32 124 189  77 242 174 212  43 152
 118 248   3 187 103 111  18  90  60 163 240  68  80 183 113  35  28  75
  85  49 239  99 254  31  70   5 128 229 144 139 133 204  55 180   9  88
   1 150 165 209  94 147  66  58 207 130 233 135 169 236 250  73 197 157
 208 107 253  76 181  36 170 224 137 192   0   2  92  41   6 237 132  72
 104 173 235 232  65  29  47  91 214 231 125 198 203 230 196  27  89  71
 120  25 127  22  39 247  95  78 238  64 184  24 105 146 220  59 108  15
 186  87 129 178  21   4 151  23 251  93  50 159  79 119 255  14 182 206
 136   8 164 185 211  19 156  52 115 252  44  16 162 222  30 126  34 123
 140 188 102 145  12  42  74  33 138 234  81 193 223 160 158 190 168  84
  62 110  48 241  97 167  56 153 205  96  54 134 216 199 201 117 177 148
 112 106  67  45  83 217 244 100 131 213 101  98 175 191 226 114 171  61
  86 219 166 141  69 210 245 228 195 179 149 155  10 143  17  57 243  37
 225  51 249 122]
Coste asociado a la mejor solución: 45200784
```

1.4. Conclusiones

La implementación del algoritmo genético permitió resolver con éxito el problema de asignación cuadrática, generando soluciones de alta calidad. El uso de operadores genéticos adecuados y una representación eficiente garantizó la preservación de las estructuras óptimas en las soluciones.

La configuración de los parámetros fue fundamental para equilibrar la exploración del espacio de búsqueda y la explotación de las mejores soluciones. No obstante, se podrían obtener mejores resultados utilizando metodologías distintas, tal y como se mostrará a continuación.

2. Algoritmo genético con variante baldwiniana (v1)

En este caso, se implementó una variante baldwiniana de un algoritmo genético (AG) estándar para abordar el problema de asignación cuadrática (QAP). Este problema, conocido por su naturaleza NP-difícil, aparece en aplicaciones prácticas como la optimización de flujos en sistemas logísticos, la asignación de tareas en instalaciones y el diseño de redes eficientes.

La variante baldwiniana del algoritmo genético incorpora técnicas de optimización local, como heurísticas greedy o búsqueda local mediante ascensión de colinas, para dotar a los individuos de la población de capacidad de aprendizaje. En esta estrategia, la evaluación del fitness de cada individuo se realiza después de aplicar una búsqueda local, pero las mejoras obtenidas no alteran el material genético original empleado para generar descendencia. Este enfoque permite mejorar la calidad de las soluciones evaluadas sin comprometer la diversidad genética de la población.

En este informe se describen los componentes clave del algoritmo baldwiniano implementado, los resultados obtenidos en comparación con el algoritmo genético estándar y los parámetros ajustados para maximizar la eficiencia y eficacia del proceso de búsqueda.

2.1. Descripción del Algoritmo Genético Baldwiniano

El algoritmo genético con variante baldwiniana se basa en los mismos principios del algoritmo genético estándar, pero con la inclusión de una fase de optimización local para evaluar el fitness de los individuos. A continuación, se detallan los componentes clave del algoritmo:

- **Representación:** Cada individuo de la población se representa como una permutación de números enteros que indican la asignación de instalaciones a ubicaciones. En este caso, se utiliza una longitud fija de 256 elementos correspondiente al problema analizado.
- **Mecanismo de selección:** Se emplea selección por torneo con un tamaño de torneo de 5. Este método selecciona aleatoriamente un subconjunto de soluciones y escoge la mejor según su aptitud.
- **Operadores genéticos:**
    - **Cruce:** Se utiliza el operador de cruce por orden (OX). Este operador garantiza que la descendencia herede la estructura de las soluciones padres sin duplicar elementos.
    - **Mutación:** Se aplica mutación por intercambio (swap) con una probabilidad de mutación configurada. Este operador selecciona aleatoriamente dos posiciones en la permutación y las intercambia.
- **Condiciones de parada:** El algoritmo ejecuta un número fijo de generaciones (50), tras el cual devuelve la mejor solución encontrada.
- **Optimización Local (Hill Climbing)**: Durante cada generación, el fitness de cada individuo se evalúa aplicando una búsqueda local basada en el método Hill Climbing. Este proceso busca una solución óptima local mejorando iterativamente la solución inicial del individuo.

2.2. Parámetros configurados

Para la ejecución del algoritmo genético, inicialmente se mantuvo tanto la población (1000) como el número de generaciones (500). No obstante, el tiempo dedicado era excesivo, por lo que se optó por deducir ambos valores considerablemente, quedando los siguientes parámetros:

- Tamaño de la población: 500
- Número de generaciones: 50
- Probabilidad de cruce: 0.9
- Probabilidad de mutación: 0.05
- Tamaño del torneo: 5
- Hill Climbing: se aplica a cada individuo para evaluar el fitness.

2.3. Resultados obtenidos

El algoritmo fue evaluado con el conjunto de datos proporcionado (“tai256c.dat”), que incluye las matrices de flujos y distancias correspondientes. Los resultados obtenidos fueron los siguientes:

- **Mejor permutación encontrada:**

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

- **Coste asociado a la mejor solución:** 46104902

Captura de pantalla:

```
Mejor solución encontrada (Baldwiniano) (permutación): [238  98  13 109 169 152  25 232 120  62 184  40  94 246  15  18  10 164
 172 196 193  66 217  88 176   5  91 221 116   7 133 166 141 146 206  71
  83  69  44  34  20  92 118 224  80 106  86 194  49 122 201 131 139 252
  32 187  23 197 144 219 226  37 223 244  63 124 129 174  54  73 182 243
 143 251  84 137   1 163  81 253 215  30 150  42 248  57 154 157 103 234
  60 189  72 255 112 138 104 247  95  59 190 211  38 117  28  19  24 199
  21  11 212 186 161  33   4 165  29 180 167 114   3  56 179  75 250 134
 236  26 229  99 102 237 205  22 241  55  17 160 227  53 130 181 135  77
  90 127 200 191 151  52  67  68 155  36  43 213  93 159  35 235   8 188
  45 168 245 147 209  48 100  27 115 175  51 222  89 148 140 132   6  31
 231 162 208  79  61 125  64 110  39  76 220 101  96  70  82   0 216 156
  14 183 126 198 149  46 145  58 107  74 111 203 142  85 233  41   9 123
 254 239 242  47 153 214 192 158  16 225 105 113 202 119   2 108  50 136
 185 121 170 210 171 128 177  97 207 218  65  78 230 249 178  87 228 173
  12 204 195 240]
Coste asociado a la mejor solución (Baldwiniano): 46104902
```

2.4. Conclusiones

Este algoritmo genético con variante baldwiniana no experimentó mejoras respecto al algoritmo anterior. Por tanto, se trató de volver a aumentar tanto el tamaño de la población como el número de generaciones, reduciendo el tiempo de ejecución optimizando el algoritmo.

Así pues, se propusieron posibles mejoras como la paralelización o el uso de un Hill Climbing selectivo, de forma que se redujera tanto el tiempo de ejecución como el coste.

3. Algoritmo genético con variante baldwiniana (v2)

En este caso, se implementó otra variante baldwiniana de un algoritmo genético (AG) estándar para abordar el problema de asignación cuadrática (QAP). A continuación, se describen los componentes clave del nuevo algoritmo baldwiniano implementado, los resultados obtenidos en comparación con el algoritmo genético estándar y los parámetros ajustados para maximizar la eficiencia y eficacia del proceso de búsqueda.

3.1. Descripción del Algoritmo Genético Baldwiniano

Este algoritmo genético con variante baldwiniana se basa en los mismos principios del algoritmo anterior. A continuación, se detallan los componentes clave del algoritmo:

- **Representación:** Cada individuo de la población se representa como una permutación de números enteros que indican la asignación de instalaciones a ubicaciones. En este caso, se utiliza una longitud fija de 256 elementos correspondiente al problema analizado.
- **Mecanismo de selección:** Se emplea selección por torneo con un tamaño de torneo de 5. Este método selecciona aleatoriamente un subconjunto de soluciones y escoge la mejor según su aptitud.
- **Operadores genéticos:**
    - **Cruce:** Se utiliza el operador de cruce por orden (OX). Este operador garantiza que la descendencia herede la estructura de las soluciones padres sin duplicar elementos.
    - **Mutación:** Se aplica mutación por intercambio (swap) con una probabilidad de mutación configurada. Este operador selecciona aleatoriamente dos posiciones en la permutación y las intercambia.
- **Condiciones de parada:** El algoritmo ejecuta un número fijo de generaciones (500), tras el cual devuelve la mejor solución encontrada.
- **Optimizaciones adicionales:**
    - **Cálculo de Deltas:** Se implementó una técnica de cálculo incremental para reducir el tiempo de evaluación durante la búsqueda local.
    - **Paralelización:** El código utiliza la librería multiprocessing para distribuir la evaluación de la búsqueda local entre varios procesos, acelerando la ejecución.
    - **Hill Climbing Selectivo:** Solo se aplica la optimización local al 10% de los mejores individuos de la población, lo que equilibra la calidad de la solución y el tiempo de cómputo.

3.2. Parámetros configurados

Para la ejecución del algoritmo genético, se utilizaron los siguientes parámetros:

- Tamaño de la población: 1000
- Número de generaciones: 500
- Probabilidad de cruce: 0.9
- Probabilidad de mutación: 0.05
- Tamaño del torneo: 5
- Porcentaje de Hill Climbing: 10% de la población en cada generación.

3.3. Resultados obtenidos

El algoritmo fue evaluado con el conjunto de datos proporcionado (“tai256c.dat”), que incluye las matrices de flujos y distancias correspondientes. Los resultados obtenidos fueron los siguientes:

- **Mejor permutación encontrada:**

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

- **Coste asociado a la mejor solución:** 44952966

- **Tiempo empleado:** 1 hora y 47 minutos.

Captura de pantalla:

```
Mejor solución encontrada (Baldwiniano): [173 171 225 242 193 156 110  20 207 178 196 201  44 203 159  96  81  76
 251  10  87 236 249 153 190  50 213 145 125  42 142 246 115 221   5  69
 150 147 135  29 216 244 231  27 186 183  62  54  18  93  52  48 128 208
 198 227   0  84  22 181  56   7 160 188 253  59  73 164 105  24 107 130
 120  33  47  98 234  39 168 255  67 132 117 139 102 122 238  90  95  35
 210  14 177 149 180  60  74   1 127  91 228   2 254 217 241 101  86 191
 245 113 218 224  80  75  38  45 133 175 194 167 195 184 229 112 151 197
 247 202   3  58  89  15  94  88 106 100  53  55 222  25  34  51 121 199
 108 146  92  17 126 166  40  65 111  82 187 119  31  41 250 233 204  97
  36  13 215 226 152  63  26 109  28 169 235  32 140  83  71 116  16 157
 131 158 200  66 118 252  37  49 170   9 214 176  12  72 209 230   4 123
 220  85  57 165  23 179  99 138  78 212 104 124   8  21 237  79  77 172
   6 192 223 240 219 185 174 163 114 148 239 205  61 189 182 206 129  43
 161  46 134 137 155  70  30  19 248  64 136 243 103 141 162  11 144 211
 232  68 143 154]
Coste asociado a la mejor solución (Baldwiniano): 44952966
```

3.4. Conclusiones

El algoritmo genético con variante baldwiniana demostró ser una mejora significativa respecto al algoritmo genético estándar para el problema de asignación cuadrática. La inclusión de optimización local mejoró la calidad de las soluciones obtenidas, aunque a costa de un mayor tiempo de cómputo.

La paralelización y el uso selectivo de hill climbing permitieron mitigar parcialmente este costo adicional, logrando un equilibrio entre eficiencia y rendimiento. Este enfoque resulta especialmente útil en problemas donde la calidad de la solución es prioritaria sobre el tiempo de ejecución.

4. Algoritmo genético con variante lamarckiana

En esta sección se describe en detalle la implementación de un algoritmo genético con variante lamarckiana, el cual ha sido diseñado específicamente para abordar el problema de asignación cuadrática (QAP). Esta variante lamarckiana combina los principios evolutivos de los algoritmos genéticos con técnicas de optimización local, permitiendo que las mejoras logradas mediante un procedimiento de Hill Climbing optimizado se integren directamente en el material genético de los individuos. Esto significa que las adaptaciones obtenidas durante el proceso de optimización local son transmitidas a las siguientes generaciones, en línea con el principio lamarckiano de herencia de características adquiridas.

El enfoque propuesto se diferencia de otras variantes genéticas al explotar de manera intensiva la capacidad de refinamiento local de cada solución, acelerando la convergencia hacia mínimos locales de alta calidad. Esto lo convierte en una herramienta efectiva para resolver instancias complejas del QAP. Además, esta integración directa de mejoras permite aprovechar al máximo el potencial de las técnicas de búsqueda local, logrando un equilibrio entre la exploración global y la explotación local del espacio de soluciones.

4.1. Descripción del Algoritmo Genético Lamarckiano

A continuación, se detallan los componentes clave del algoritmo:

- **Representación:** Cada individuo de la población se representa como una permutación de números enteros que indican la asignación de instalaciones a ubicaciones. En este caso, se utiliza una longitud fija de 256 elementos correspondiente al problema analizado.
- **Mecanismo de selección:** Se emplea selección por torneo con un tamaño de torneo de 5. Este método selecciona aleatoriamente un subconjunto de soluciones y escoge la mejor según su aptitud.
- **Operadores genéticos:**
    - **Cruce:** Se utiliza el operador de cruce por orden (OX). Este operador garantiza que la descendencia herede la estructura de las soluciones padres sin duplicar elementos.
    - **Mutación:** Se aplica mutación por intercambio (swap) con una probabilidad de mutación configurada. Este operador selecciona aleatoriamente dos posiciones en la permutación y las intercambia.
- **Condiciones de parada:** El algoritmo ejecuta un número fijo de generaciones (200), tras el cual devuelve la mejor solución encontrada.
- **Optimización Local (Hill Climbing)**: Cada individuo de la población es mejorado mediante un proceso de Hill Climbing, que explora su vecindario para encontrar soluciones con menor costo. Este procedimiento optimizado emplea cálculos de deltas para reducir el tiempo de evaluación en cada iteración.
- **Heurística Greedy:** La mejora se evalúa mediante el cálculo del cambio incremental en la función objetivo (fitness), conocido como delta fitness, al intercambiar dos posiciones en la permutación que representa una solución. Solo se aceptan los movimientos que resulten en una mejora inmediata del valor del fitness, lo que se alinea con la lógica greedy de

tomar la mejor decisión local en cada paso. Este enfoque asegura que cada individuo sea refinado hacia una solución localmente óptima antes de pasar a la siguiente generación, aprovechando así la estructura del problema para mejorar la calidad de las soluciones de manera eficiente. Al mismo tiempo, al aplicar la heurística de manera iterativa dentro del algoritmo genético, se mantiene un equilibrio entre las mejoras locales y la capacidad de explorar nuevas regiones del espacio de búsqueda mediante los operadores de cruce y mutación.

4.2. Parámetros configurados

Para la ejecución del algoritmo genético, se redujo tanto la población (500) como el número de generaciones (200), pues el tiempo dedicado era excesivo y el menor coste se observaba bastante antes, quedando los siguientes parámetros:

- Tamaño de la población: 500
- Número de generaciones: 200
- Probabilidad de cruce: 0.9
- Probabilidad de mutación: 0.05
- Tamaño del torneo: 5
- Hill Climbing: se aplica a toda la población (herencia directa).

4.3. Resultados obtenidos

El algoritmo fue evaluado con el conjunto de datos proporcionado (“tai256c.dat”), que incluye las matrices de flujos y distancias correspondientes. Los resultados obtenidos fueron los siguientes:

- **Mejor permutación encontrada:**

```
[ 22  83 239  90  39 110 181 219 141  20 236 139 158  54 188 212  18  98
 146  62 254 227  13 127 231  24 195  52 193 222 144  66  77 137 169 178
 225  16  71  15  64   9  86  69  58 120  49  35 190 248 199 205 156 250
  73  11 124  30 151 134 163   5 131 171 175 113 186 208 100 166 201 233
  96   1 184  60  92 246  28 214 242 148 107  95  43 105  41  47 117 103
 244 176 132 207 183  94 220 170  48  81  25 142 240 173   4 149 115 125
 119 229  93  14   0 118 192  44  76 182 252 161  56   3 216  85  88 202
 123 249 235 185 104 108 191   2  57 198 101  45 243 154  80   6 102 221
  82 234 209 116 203  74 218  26 172 129 122 155  99  37 128 135  87 167
 111  29  72  10 200 247 145 147  70  46 140 133  31 130 160  53 189  61
 150 114  40 204 245 226 238 255  84 228 177 168 211  36  19  97 143 179
```

```
  7  17 197 109 153 138 106 232  68  33  67 187 174   8 112  50 224  63

 79  12  65 217  91 196 223  42  23 251  78 253  55  27  34 164 215 162

 21  51 206 210 230  75 157  32 159 126 152 165  59 237 121 213  38 136

 89 194 180 241]
```

- **Coste asociado a la mejor solución:** 44901164
- **Tiempo empleado:** aproximadamente 3 horas.

Captura de pantalla:

```
Mejor solución encontrada (Lamarckiano): [ 22  83 239  90  39 110 181 219 141  20 236 139 158  54 188 212  18  98
 146  62 254 227  13 127 231  24 195  52 193 222 144  66  77 137 169 178
 225  16  71  15  64   9  86  69  58 120  49  35 190 248 199 205 156 250
  73  11 124  30 151 134 163   5 131 171 175 113 186 208 100 166 201 233
  96   1 184  60  92 246  28 214 242 148 107  95  43 105  41  47 117 103
 244 176 132 207 183  94 220 170  48  81  25 142 240 173   4 149 115 125
 119 229  93  14   0 118 192  44  76 182 252 161  56   3 216  85  88 202
 123 249 235 185 104 108 191   2  57 198 101  45 243 154  80   6 102 221
  82 234 209 116 203  74 218  26 172 129 122 155  99  37 128 135  87 167
 111  29  72  10 200 247 145 147  70  46 140 133  31 130 160  53 189  61
 150 114  40 204 245 226 238 255  84 228 177 168 211  36  19  97 143 179
   7  17 197 109 153 138 106 232  68  33  67 187 174   8 112  50 224  63
  79  12  65 217  91 196 223  42  23 251  78 253  55  27  34 164 215 162
  21  51 206 210 230  75 157  32 159 126 152 165  59 237 121 213  38 136
  89 194 180 241]
Coste asociado a la mejor solución (Lamarckiano): 44901164
```

4.4. Conclusiones

El algoritmo genético con variante lamarckiana demostró una mejora significativa en la calidad de las soluciones obtenidas en comparación con las variantes anteriores. Esto se debe a que el enfoque lamarckiano incorpora directamente las mejoras locales generadas por el procedimiento de Hill Climbing optimizado en el material genético de los individuos, acelerando la convergencia hacia soluciones de alta calidad.

Sin embargo, esta integración de mejoras locales aumentó notablemente el tiempo de ejecución del algoritmo debido a la aplicación intensiva de la optimización local a toda la población en cada generación. Este aumento en el costo computacional representa un desafío clave, especialmente en problemas de gran escala como el QAP.

5. Comparación de algoritmos

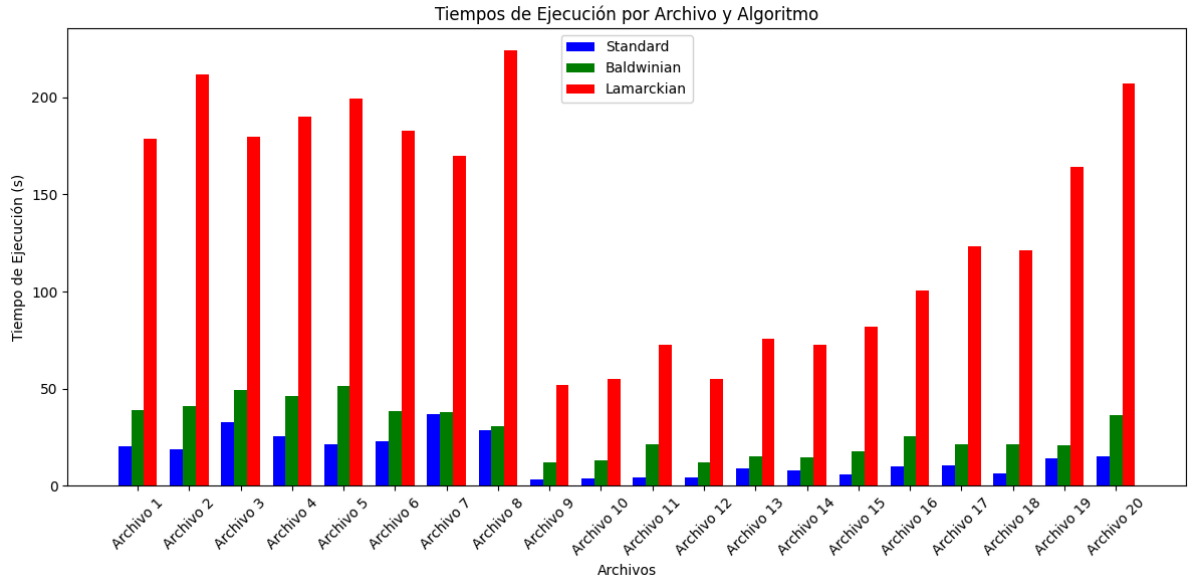
Para realizar la comparación se han seleccionado tres variantes de algoritmos genéticos: estándar, baldwiniano y lamarckiano. Estas variantes fueron ejecutadas con todos los conjuntos de prueba proporcionados. Con el objetivo de optimizar los tiempos de ejecución y evitar evaluaciones innecesarias, se estableció un criterio de parada común: la ejecución se detiene si el mejor coste no mejora durante 15 generaciones consecutivas.

A continuación, se describen brevemente las características de cada variante:

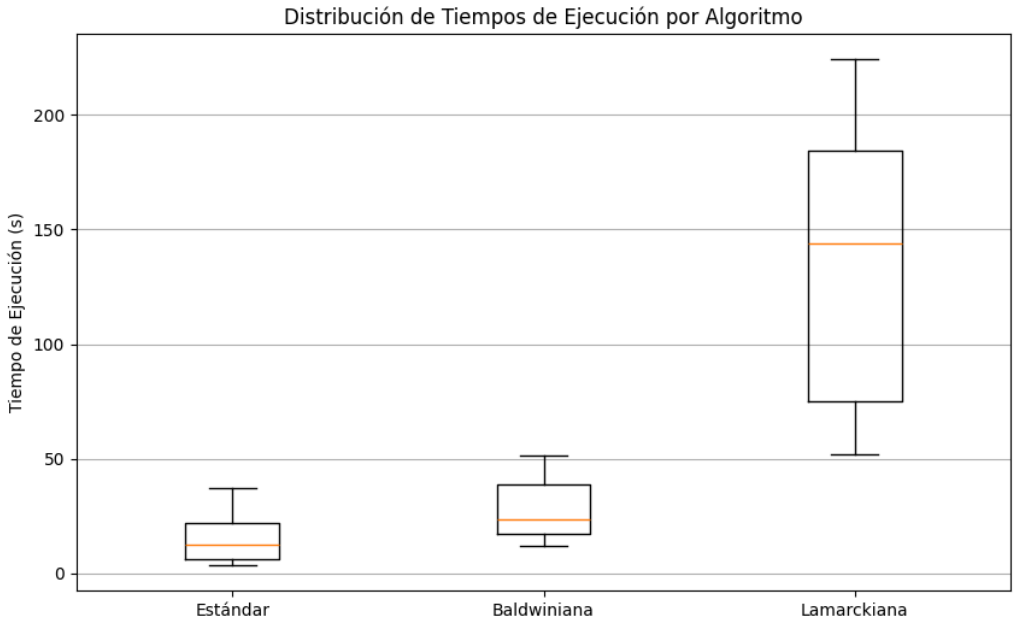
- **Algoritmo Genético Estándar**: Este enfoque utiliza selección por torneo, cruce OX (Order Crossover) y mutación por intercambio. No incorpora técnicas de aprendizaje o búsqueda local.
- **Algoritmo Genético Baldwiniano**: Similar al AGE, pero incorpora una fase de aprendizaje local en cada individuo sin alterar directamente su representación genética. Esto simula la evolución basada en la adaptabilidad de los individuos.
- **Algoritmo Genético Lamarckiano**: Integra una búsqueda local basada en hill climbing aplicada a cada individuo, modificando directamente su representación genética. Este enfoque persigue una mejora directa de la calidad de las soluciones.

Los resultados de las ejecuciones se han evaluado considerando dos aspectos clave: la evolución del fitness (coste de la mejor solución en cada generación) y los tiempos de ejecución de las tres variantes.

En la siguiente gráfica, se observa que el tiempo requerido por el algoritmo lamarckiano es significativamente mayor en comparación con las otras dos variantes. Esto se debe al costo computacional adicional que implica la optimización local aplicada a cada individuo en cada generación. Por otro lado, los algoritmos estándar y baldwiniano presentan tiempos de ejecución más reducidos, siendo el estándar el más eficiente en general.



La segunda gráfica refuerza esta observación, mostrando que el algoritmo lamarckiano tiene una mayor dispersión en los tiempos de ejecución, mientras que las variantes estándar y baldwiniana presentan tiempos más homogéneos.



Respecto a la calidad de las soluciones (fitness), los resultados sugieren que el algoritmo lamarckiano tiende a encontrar soluciones de mejor calidad en la mayoría de los casos, gracias a la búsqueda local. Por su parte, el algoritmo baldwiniano también presenta mejoras en la calidad en comparación con el estándar, pero con menor impacto que el lamarckiano. Sin embargo, el tiempo extra requerido por el algoritmo lamarckiano puede no justificar la mejora marginal en algunos escenarios, especialmente para problemas donde se prioriza el tiempo de ejecución sobre la calidad absoluta.

En conclusión, en términos de calidad de las soluciones, el algoritmo lamarckiano es el más efectivo, seguido del baldwiniano y, finalmente, del estándar. Sin embargo, el costo computacional asociado con el enfoque lamarckiano lo hace menos atractivo en escenarios donde el tiempo de ejecución es crítico. El algoritmo estándar destaca por su eficiencia en tiempo, siendo una opción adecuada para problemas donde la calidad de las soluciones obtenidas no es tan crítica. Por otro lado, el algoritmo baldwiniano, al ofrecer un balance entre tiempo de ejecución y calidad de las soluciones, se presenta como una alternativa intermedia interesante.