Práctica 2 - Problemas de optimización con técnicas basadas en poblaciones

Luis Miguel Guirado Bautista

20 de Mayo de 2022 Curso 2021/2022

Correo: luismgb@correo.ugr.es

DNI: 75942712R

Subgrupo: Martes, 3

Problema: Mínima Dispersión Diferencial

Índice

1 Motivación

En esta práctica abordaremos el problema de la Mínima Dispersión Diferencial, que consiste en seleccionar un subconjunto de puntos ya definidos en el plano tal que la dispersión entre ellos sea mínima.

Siendo i y j dos puntos del plano y S el subconjunto de puntos seleccionados, la función objetivo (dispersión de los puntos seleccionados) es:

$$\Delta(S) = max_{i \in S}(\sum_{j \in S} d_{ij}) - min_{i \in M}(\sum_{j \in S} d_{ij})$$

Siendo n el número de posibles destinos, la matriz de distancias entre los n puntos se define como:

$$D = \begin{pmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{12} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & d_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nn} \end{pmatrix}$$

El fichero `datos` de la práctica contiene la diagonal superior de 50 matrices de distancias.
Consideraremos D_b la matriz de distancias del caso b.
Para este problema usaremos los siguientes algortimos:

- Greedy. Ir escogiendo los que menor dispersión supongan a partir de un punto aleatorio.
- Búsqueda Local. A partir de un subconjunto S aleatorio, intercambiar ciertos puntos de S con los del resto de puntos de manera que la dispersión entre la solución antigua y la nueva se vea reducida.
- Algoritmos Genéticos. Simular una población de soluciones P y finalmente escoger la mejor solución de todo P. Utilizaremos dos variantes:
  - Generacional.
  - Estacionario.

  Y cada una de estas variantes tendrá otras subvariantes según el tipo de cruce que se vaya a emplear a la hora de realizar la simulación:
  - Cruce uniforme. (AGG-Uniforme y AGE-Uniforme)
  - Cruce de posición. (AGG-Posición y AGE-Posición)
- Algoritmo Memético. Híbrido entre el AGG-Uniforme y Búsqueda Local. Usaremos tres variantes según la probabilidad de que se aplique Búsqueda Local.
  - AM(10,1). Cada 10 generaciones, se aplica Búsqueda Local en todo P.
  - AM(10,0.1). Cada 10 generaciones, se aplica Búsqueda Local en $0.1|P|$ individuos aleatorios.
  - AM(10,0.1-mej). Cada 10 generaciones, se aplica Búsqueda Local en los mejores $0.1|P|$ individuos.

Nota No se ha realizado el AM

2 Aplicación de los algoritmos al problema

Para todos los algoritmos, salvo el memético, representaremos S como un vector binario, por ejemplo, si en S tenemos los puntos $\{1, 2, 4\}$ y $n = 5$:

$$S_{bin} = [1, 1, 0, 1, 0]$$

2.1 Greedy

Comenzaremos el algoritmo escogiendo un punto aleatorio del plano para añadirlo a nuestra solución S, e iremos añadiendo más puntos a S según el siguiente criterio:

$$u^* = min_{u \notin S}(g(u))$$

Es decir, el punto cuya $g(u)$ sea la menor de todas. El cálculo de $g(u)$ se hace así:

$$u \notin S;\ \partial(u) = \sum_{v \in S} d_{uv}$$

$$v \in S;\ \partial(v) = d_{uv} + SumaAnterior(v)$$

Nota Siendo v' el punto anterior a v, $SumaAnterior(v)$ es $\partial(v')$. Si es el primer punto de S, entonces $SumaAnterior(v) = 0$

$$\partial_{max}(u) = max(\partial(u), max_{v \in S}(\partial(v)))$$

$$\partial_{min}(u) = min(\partial(u), min_{v \in S}(\partial(v)))$$

$$g(u) = \partial_{max}(u) - \partial_{min}(u)$$

Parará cuando $|S| = m$, entonces devolverá la solución y su dispersión (no $g(u)$).

2.2 Búsqueda Local

Comienza con una solución completa y aleatoria. Entonces para cada punto $u \in S$ y $v \notin S$, se genera una solución S' que resulta de intercambiar el elemento u con el elemento v. Esto se repite hasta que se haya alcanzado el máximo de iteraciones $T = 100.000$ o no se haya encontrado una S' que sea mejor que S. S' reemplazará a S si:

$$\Delta Z_{mm} = Z_{mm}(S') - Z_{mm}(S) < 0$$

$$Z_{mm}(S') = (\partial^{'}_{max} - \partial^{'}_{min})$$

$$\partial^{'}_{max} = max(\partial(v), max_{w \in S}(\partial(w)))$$

$$\partial^{'}_{min} = min(\partial(v), min_{w \in S}(\partial(w)))$$

$$\partial(v) = \sum_{w \in S} d_{vw}$$

$$\partial(w) = SumaAnterior(w) - d_{wu} + d_{wv}$$

2.3 Genético

Los algortimos genéticos consisten en generar un conjunto de soluciones P y aplicarles una serie de cómputos hasta que se hayan alcanzado T iteraciones (o generaciones), entonces devolverá el mejor individuo de todos y su dispersión (valoraremos por MDD).

- Selección. Seleccionamos unos individuos de P para aplicar el siguiente paso.
- Cruce. Escoger dos individuos y *escoger lo mejor* de cada uno para generar otro individuo que será el hijo.
- Mutación. Modificar ligeramente o no los individuos del conjunto anterior. Intercambiaremos un elemento x_i con otro x_j t.q el valor de estos no sean iguales.
- Reemplazamiento. Insertar los hijos en P.

Siendo dos padres S_1 y S_2 y su hijo S_h, para cada una de las variantes de los algoritmos genéticos (generacional y estacionario), consideraremos otras subvariantes dependiendo del tipo de cruce que realizen, que puede ser:

- Uniforme. Cada elemento (gen) en común con S_1 y S_2 los hereda S_h. El resto de elementos son escogidos aleatoriamente entre los dos padres. Esto da lugar a soluciones que no son factibles, por lo que requerirá un *reparador* para que vuelvan a ser factibles.
- De posición. Este es más simple, cada elemento de la solución en común con S_1 y S_2 los hereda S_h como en el anterior. Pero ahora los genes restantes se escogen de un padre y se le asignan aleatoriamente a S_h

2.3.1 Generacional

- Selección. Seleccionamos todos los individuos de P. De modo que $P = P'$
- Cruce. Siendo $p_c = 0.7$ la probabilidad de cruce, cruzaremos las primeras $\left\lfloor \frac{|P|}{2} \cdot p_c \right\rfloor$ parejas.
- Mutación. Siendo $p_m = 0.1$ la probabilidad de mutación, mutaremos $\lfloor |P| \cdot p_m \rfloor$ veces a los individuos de P' de forma aleatoria, además, un individuo podrá mutar varias veces por iteración.
- Reemplazamiento. Si el mejor individuo de P no sobrevive, entonces reemplazaremos el peor individuo de P' por el mejor de P.

2.3.2 Estacionario

- Selección. Seleccionamos dos padres S_1 y S_2 mediante torneo binario, entonces $P' = \{S_1, S_2\}$
- Cruce. Se cruzan P' dos veces, y sus hijos reemplazan a los padres.
- Mutación. Siendo $p_m = 0.1$ la probabilidad de mutación, generaremos un número aleatorio $r \in [0, 1]$ para cada uno de los hijos, si $r < p_m$ entonces ese hijo mutará
- Reemplazamiento. Los hijos compiten para ver quien entra a P, el ganador reemplazará al peor individuo de P.

3 Algoritmos empleados (pseudocódigo)

3.1 Greedy

function EscogerGreedy(D, S)
    $sumas \leftarrow 0_n$ ▷ Vector de n ceros.
    $mejor\_disp \leftarrow \infty$
    $mejor \leftarrow$ `RandomChoice`$([0, n))$ ▷ Función de NumPy. Entero aleatorio entre $[0, n]$
    for u **in** $\{0, 1, \ldots, n\}$ **do**
        $suma\_u \leftarrow 0$
        if $x_u = 0$ **then** ▷ $u \notin S$
            for v **in** $\{0, 1, \ldots, n\}$ **do**
                if $x_v = 1$ **then** ▷ $v \in S$
                    $suma\_u \leftarrow suma\_u + d_{uv}$
                    $sumas[v] \leftarrow sumas[v] + d_{uv}$
                end if
            end for
            $sumas[u] \leftarrow suma\_u$
        end if
    end for
    $max\_v \leftarrow max_{v \in S}(sumas[v])$
    $min\_v \leftarrow min_{v \in S}(sumas[v])$
    for u **in** $\{0, 1, \ldots, n\}$ **do**
        $dmax \leftarrow max(sumas[u], max\_v)$
        $dmin \leftarrow min(sumas[u], min\_v)$
        $disp \leftarrow dmax - dmin$
        if $disp < mejor\_disp$ **then**
            $mejor\_disp \leftarrow disp$
            $mejor \leftarrow u$
        end if
    end for
    return $mejor$ ▷ Devuelve el punto u^*
end function

function Greedy(n, m, D)
    $s \leftarrow 0_n$
    $s[$`RandomChoice`$([0, n))] \leftarrow 1$
    while `Unos`$(s) < m$ **do**
        $idx \leftarrow$ `EscogerGreedy`(D, s)
        $s[idx] \leftarrow 1$
    end while
    $costo \leftarrow$ `Evaluar`(s)
    return $s, costo$
end function

3.2 Búsqueda Local

function EscogerBL(D, S)
  $coste\_actual \leftarrow$ `Evaluar`(S, D)
  for u **in** $\{0, 1, \ldots, n\}$ **do**
    for v **in** $\{0, 1, \ldots, n\}$ **do**
      if $x_u = 1$ **and** $x_v = 0$ **then**
        $sumas \leftarrow 0_n$
        $S' \leftarrow$ `Intercambio`(u, v, S)
        for w **in** $\{0, 1, \ldots, n\}$ **do**
          if $x_w^{'} = 1$ **then**
            $sumas[v] \leftarrow sumas[v] + d_{vw}$
            $sumas[w] \leftarrow sumas[w] - d_{wu} + d_{wv}$
          end if
        end for
        $max'_w \leftarrow max_{i \in S'}(sumas[i])$
        $min'_w \leftarrow min_{i \in S'}(sumas[i])$
        $max_w \leftarrow max_{i \in S}(sumas[i])$
        $min_w \leftarrow min_{i \in S}(sumas[i])$
        $dmax' \leftarrow max(sumas[v], max'_w)$
        $dmin' \leftarrow min(sumas[v], min'_w)$
        $dmax \leftarrow max(sumas[v], max_w)$
        $dmin \leftarrow min(sumas[v], min_w)$
        $diff' \leftarrow dmax' - dmin'$
        $diff \leftarrow dmax - dmin$
        if $diff' - diff < 0$ **then**
          $coste' \leftarrow$ `Evaluar`(S', D)
          return $S', coste'$
        end if
      end if
    end for
  end for
  return $s, coste\_actual$
end function

function BL(n, m, D)
  $actual \leftarrow$ `SolucionAleatoria`(n, m)
  $coste\_actual \leftarrow$ `Evaluar`(s, D)
  $coste\_vecino \leftarrow \infty$
  $iters \leftarrow 0$
  while $iters < T$ **and** $coste\_vecino \geq coste\_actual$ **do** ▷ Recordatorio: $T = 100.000$
    $iters \leftarrow iters + 1$
    $vecino, coste\_vecino \leftarrow$ `EscogerBL`$(D, actual)$
    if $coste\_vecino < coste\_actual$ **then**
      $actual, coste\_actual = vecino, coste\_vecino$
    end if
  end while
  return $actual, coste\_actual$
end function

3.3 Genéticos

3.3.1 AGG-Uniforme

Constantes $p_c = 0.7;\ p_m = 0.1;\ |P| = 50$

function AGGUniforme($n,\ m,\ D$)
- $t \leftarrow 0$
- $cruces\_esperados \leftarrow \left\lfloor \frac{|P|}{2} \cdot p_c \right\rfloor$
- $mutaciones \leftarrow \lfloor |P| \cdot p_m \rfloor$
- $P \leftarrow$ `PoblacionAleatoria`$(n, m, |P|)$
- **while** $t < T$ **do**
  - $t \leftarrow t + 1$
  - $mejor\_anterior \leftarrow$ `EncontrarMejor`$(P, D)[0]$ ▷ `EncontrarMejor` devuelve índice y costo
  - $P' \leftarrow P$
  - $i \leftarrow 0$
  - $cruces\_realizados \leftarrow 0$
  - **while** $cruces\_realizados < cruces\_esperados$ **and** $i < n - 1$ **do**
    - $reemplazado \leftarrow$ `RandomChoice`$([0, 1])$
    - $P'[i + reemplazado] \leftarrow$ `CruceUniforme`$(P'[i], P'[i + 1], n, m, D)$
    - $cruces\_realizados \leftarrow cruces\_realizados + 1$
    - $i \leftarrow i + 2$
  - **end while**
  - $idxs \leftarrow$ `RandomChoice`$([0, |P|), mutaciones)$ ▷ Genera $mutaciones$ números con repetidos
  - **for** idx **in** $idxs$ **do**
    - `Mutar`$(P[idx])$
  - **end for**
  - **for** I **in** P' **do**
    - **if** $P[mejor\_anterior] = I$ **then**
      - $peor \leftarrow$ `EncontrarPeor`(P', D)
      - $P'[peor] \leftarrow P[mejor\_anterior]$
      - **break**
    - **end if**
  - **end for**
  - $P \leftarrow P'$
- **end while**
- $mejor, coste \leftarrow$ `EncontrarMejor`(P, D)
- **return** $P[mejor], coste$

end function

3.3.2 AGG-Posición

Constantes $p_c = 0.7;\ p_m = 0.1;\ |P| = 50$

function AGGPosicion($n,\ m,\ D$)
- $t \leftarrow 0$
- $cruces\_esperados \leftarrow \left\lfloor \frac{|P|}{2} \cdot p_c \right\rfloor$
- $mutaciones \leftarrow \lfloor |P| \cdot p_m \rfloor$
- $P \leftarrow$ `PoblacionAleatoria`$(n, m, |P|)$
- **while** $t < T$ **do**
  - $t \leftarrow t + 1$
  - $mejor\_anterior \leftarrow$ `EncontrarMejor`$(P, D)[0]$ ▷ `EncontrarMejor` devuelve índice y costo
  - $P' \leftarrow P$
  - $i \leftarrow 0$
  - $cruces\_realizados \leftarrow 0$
  - **while** $cruces\_realizados < cruces\_esperados$ **and** $i < n - 1$ **do**
    - $reemplazado \leftarrow$ `RandomChoice`$([0, 1])$
    - $P'[i + reemplazado] \leftarrow$ `CrucePosicion`$(P'[i], P'[i + 1], n, m)$
    - $cruces\_realizados \leftarrow cruces\_realizados + 1$
    - $i \leftarrow i + 2$
  - **end while**
  - $idxs \leftarrow$ `RandomChoice`$([0, |P|), mutaciones)$ ▷ Genera $mutaciones$ números con repetidos
  - **for** idx **in** $idxs$ **do**
    - `Mutar`$(P[idx])$
  - **end for**
  - **for** I **in** P' **do**
    - **if** $P[mejor\_anterior] = I$ **then**
      - $peor \leftarrow$ `EncontrarPeor`(P', D)
      - $P'[peor] \leftarrow P[mejor\_anterior]$
      - **break**
    - **end if**
  - **end for**
  - $P \leftarrow P'$
- **end while**
- $mejor, coste \leftarrow$ `EncontrarMejor`(P, D)
- **return** $P[mejor], coste$

end function

3.3.3 AGE-Uniforme

Constantes $p_c = 1;\ p_m = 0.1;\ |P| = 50$

function AGEUniforme(n, m, D)
  $t \leftarrow 0$
  $P \leftarrow$ `PoblacionAleatoria`$(n, m, |P|)$
  while $t < T$ **do**
    $t \leftarrow t + 1$
    $P' \leftarrow$ [`TorneoBinario`(P, D), `TorneoBinario`(P, D)]
    for p **in** P' **do**
      $hijo \leftarrow$ `CruceUniforme`$(P'[0], P'[1], n, m, D)$
      if `Rand`$() < p_m$ **then** ▷ Función de NumPy. Devuelve un número entre [0,1]
        `Mutar`$(hijo)$
      end if
      $p \leftarrow hijo$
    end for
    for p **in** P' **do**
      $peor \leftarrow$ `EncontrarPeor`(P, D)
      if `Evaluar`$(P[peor], D) \geq$ `Evaluar`(P', D) **then**
        $P[peor] = p$
        break
      end if
    end for
  end while
  $mejor, coste \leftarrow$ `EncontrarMejor`(P, D)
  return $P[mejor], coste$
end function

3.3.4 AGE-Posición

Constantes $p_c = 1;\ p_m = 0.1;\ |P| = 50$

function AGEPosicion(n, m, D)
  $t \leftarrow 0$
  $P \leftarrow$ `PoblacionAleatoria`$(n, m, |P|)$
  while $t < T$ **do**
    $t \leftarrow t + 1$
    $P' \leftarrow$ [`TorneoBinario`(P, D), `TorneoBinario`(P, D)]
    for p **in** P' **do**
      $hijo \leftarrow$ `CrucePosicion`$(P'[0], P'[1], n, m, D)$
      if `Rand`$() < p_m$ **then** ▷ Función de NumPy. Devuelve un número entre [0,1]
        `Mutar`$(hijo)$
      end if
      $p \leftarrow hijo$
    end for
    for p **in** P' **do**
      $peor \leftarrow$ `EncontrarPeor`(P, D)
      if `Evaluar`$(P[peor], D) \geq$ `Evaluar`(P', D) **then**
        $P[peor] = p$
        break
      end if
    end for
  end while
  $mejor, coste \leftarrow$ `EncontrarMejor`(P, D)
  return $P[mejor], coste$
end function

3.4 Otras funciones (operadores, evaluadores, etc.)

function EVALUAR(S, D)
  $sumas \leftarrow []$
  for i **in** $\{0, 1, \ldots, |S|\}$ **do**
    if $x_i = 1$ **then**
      $suma \leftarrow 0$
      for j **in** $\{0, 1, \ldots, |S|\}$ **do**
        if $x_j = 1$ **and** $j \neq i$ **then**
          $suma \leftarrow suma + d_{ij}$
        end if
      end for
      $sumas \leftarrow sumas \cup \{suma\}$
    end if
  end for
  return $max(sumas) - min(sumas)$
end function

function INTERCAMBIO(i, j, S)
  $ret \leftarrow S$
  assert $ret[i] = 1$ **and** $ret[j] = 0$
  $ret[i], ret[j] = ret[j], ret[i]$
  return ret
end function

function SOLUCIONALEATORIA(n, m)
  assert $n \geq m$
  $ceros \leftarrow 0_{(n-m)}$
  $unos \leftarrow 1_m$
  $x \leftarrow ceros \cup unos$
  `Shuffle`(x)
  return x
end function

function ENCONTRARMEJOR(P, D)
  $resultados \leftarrow []_{|P|}$ ▷ Lista de tamaño $|P|$ vacía
  for i **in** $\{0, 1, \ldots, |P|\}$ **do**
    $resultados[i] \leftarrow$ `Evaluar`$(P[i], D)$
  end for
  $idx \leftarrow resultados$.`Index`$(min(resultados))$
  return idx, $resultados[idx]$
end function

function ENCONTRARPEOR(P, D)
  $resultados \leftarrow []_{|P|}$
  for i **in** $\{0, 1, \ldots, |P|\}$ **do**
    $resultados[i] \leftarrow$ `Evaluar`$(P[i], D)$
  end for
  return $resultados$.`Index`$(max(resultados))$
end function

```
function TorneoBinario(P, D)
    p_1 ← RandomChoice([0, |P|))
    p_2 ← RandomChoice([0, |P|))
    if Evaluar(P[p_1], D) ≤ Evaluar(P[p_2], D) then
        return P[p_1]
    end if
    return P[p_2]
end function
function PoblacionAleatoria(n, m, hab)
    ret ← []_hab
    for i in {0, 1, hab} do
        ret[i] ← SolucionAleatoria(n, m)
    end for
    return ret
end function
function Mutar(S)
    i ← RandomChoice([0, |S|))
    j ← RandomChoice([0, |S|))
    while i = j and x_i = x_j do
        j ← RandomChoice([0, |S|))
    end while
    x_i, x_j ← x_j, x_i
end function
function CrucePosicion(p_1, p_2, n, m)
    if p_1 = p_2 then
        return p_1
    end if
    h ← 0_n
    restos ← []
    primer_padre ← RandomChoice([0, 1])
    for i in {0, 1, ..., n} do
        if p_1[i] == p_2[i] then
            h[i] ← p_1[i]
        else
            if primer_padre > 0 then
                restos[i] ← (i, p_1[i])
            else
                restos[i] ← (i, p_2[i])
            end if
        end if
    end for
    Shuffle(restos[][1])
    for (i, x) in restos do
        h[i] ← x
    end for
    return h
end function
```

```
function CruceUniforme(p_1, p_2, n, m, D)
    if p_1 = p_2 then
        return p_1
    end if
    h ← 0_n
    for i in {0, 1, ..., n} do
        if p_1[i] = p_2[i] then
            h[i] ← p1[i]
        else
            primer_padre ← RandomChoice([0, 1])
            if primer_padre > 0 then
                h[i] ← p_1[i]
            else
                h[i] ← p_2[i]
            end if
        end if
    end for
    Reparar(h, n, m, D)
    return h
end function
function Average(S, D)
    suma ← S
    for i in {0, 1, ..., |S|} do
        for j in {0, 1, ..., |S|} do
            if S[i] = 1 and S[j] = 1 and i ≠ j then
                suma ← suma + d_ij
            end if
        end for
    end for
    return suma
end function
```

```
function REPARAR(S, n, m, D)
    v ← m − Unos(S)
    avrg ← Average(S, D)
    if v ≠ 0 then
        while v < 0 do
            for j in {0, 1, ..., n} do
                max ← −∞
                suma ← 0
                for i in {0, 1, ..., n} do
                    if i ≠ j then
                        suma ← suma + |d_ij − avrg|
                    end if
                end for
                if suma > max then
                    max ← suma
                    escogido ← j
                end if
            end for
        end while
        while v > 0 do
            for j in {0, 1, ..., n} do
                min ← ∞
                suma ← 0
                for i in {0, 1, ..., n} do
                    if i ≠ j then
                        suma ← suma + |d_ij − avrg|
                    end if
                end for
                if suma < min then
                    min ← suma
                    escogido ← j
                end if
            end for
        end while
    end if
end function
```

4 Desarrollo de la práctica

4.1 Lenguajes y librerías

La práctica se ha desarrollado en Python 3.10.4, utilizando como editor de texto VSCode y utilizando librerías de Python IPDB (depurador), IPython (intérprete interactivo), NumPy (matemáticas y aleatoriedad) y Numba (optimización del código).

Gracias a Numba, la ejecución de todos los casos ha pasado a ser de aproximadamente 5 horas a menos de una. Aun así, adaptar el código como quería Numba me llevó varios días. Mi última práctica de MH en Python.

4.1.1 Estructura del directorio de la práctica

- `src`. Código fuente de la práctica.
  - `gkdmh.py` Funciones para la gestión de ficheros de datos GKD
  - `pobalg.py` Todos los algoritmos y funciones principales usadas en la práctica
  - `main.py` Programa principal
- `datos`. Instancias de datos empleadas para la ejecución del programa.
- `informe.pdf`. Este archivo :)

4.1.2 Como ejecutar el programa

Ejecutar desde el directorio de la práctica:

```
$ python src/main.py
```

Es necesario tener instaladas las librerías NumPy y Numba, mencionadas anteriormente. Puedes instalarlas con:

```
$ pip install numba numpy
```

5 Resultados

5.1 Tablas

5.1.1 Greedy

Greedy			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	0,00E+00
GKD-b_2_n25_m2	0,0000	0,00	0,00E+00
GKD-b_3_n25_m2	0,0000	0,00	0,00E+00
GKD-b_4_n25_m2	0,0000	0,00	0,00E+00
GKD-b_5_n25_m2	0,0000	0,00	0,00E+00
GKD-b_6_n25_m7	282,8634	95,50	0,00E+00
GKD-b_7_n25_m7	169,4182	91,68	0,00E+00
GKD-b_8_n25_m7	89,2878	81,23	0,00E+00
GKD-b_9_n25_m7	130,9580	86,97	0,00E+00
GKD-b_10_n25_m7	157,7188	85,25	0,00E+00
GKD-b_11_n50_m5	94,2099	97,96	0,00E+00
GKD-b_12_n50_m5	109,7549	98,07	0,00E+00
GKD-b_13_n50_m5	24,5726	90,39	0,00E+00
GKD-b_14_n50_m5	153,4959	98,92	0,00E+00
GKD-b_15_n50_m5	77,0270	96,30	0,00E+00
GKD-b_16_n50_m15	296,9996	85,61	0,00E+00
GKD-b_17_n50_m15	176,9595	72,81	0,00E+00
GKD-b_18_n50_m15	255,0555	83,06	0,00E+00
GKD-b_19_n50_m15	449,3218	89,67	0,00E+00
GKD-b_20_n50_m15	329,4570	85,52	0,00E+00
GKD-b_21_n100_m10	195,1019	92,91	0,00E+00
GKD-b_22_n100_m10	369,4803	96,30	0,00E+00
GKD-b_23_n100_m10	539,7462	97,16	0,00E+00
GKD-b_24_n100_m10	323,1817	97,33	0,00E+00
GKD-b_25_n100_m10	216,2661	92,05	1,00E-03
GKD-b_26_n100_m30	778,2323	78,32	1,00E-03
GKD-b_27_n100_m30	877,0421	85,51	1,00E-03
GKD-b_28_n100_m30	875,3977	87,85	1,00E-03
GKD-b_29_n100_m30	570,9338	75,92	1,00E-03
GKD-b_30_n100_m30	991,9203	87,15	0,00E+00
GKD-b_31_n125_m12	257,5482	95,44	0,00E+00
GKD-b_32_n125_m12	295,1335	93,63	0,00E+00
GKD-b_33_n125_m12	407,6001	95,45	0,00E+00
GKD-b_34_n125_m12	220,0174	91,14	0,00E+00
GKD-b_35_n125_m12	316,4681	94,28	1,00E-03
GKD-b_36_n125_m37	1168,2008	86,69	2,00E-03
GKD-b_37_n125_m37	1615,3068	87,69	1,00E-03
GKD-b_38_n125_m37	931,6132	79,82	1,00E-03
GKD-b_39_n125_m37	1238,9199	86,39	2,00E-03
GKD-b_40_n125_m37	772,5538	76,93	1,00E-03
GKD-b_41_n150_m15	527,5433	95,57	1,00E-03
GKD-b_42_n150_m15	336,1816	92,03	1,00E-03
GKD-b_43_n150_m15	465,1800	94,25	0,00E+00
GKD-b_44_n150_m15	362,8242	92,85	0,00E+00
GKD-b_45_n150_m15	405,7743	93,16	1,00E-03
GKD-b_46_n150_m45	1648,8491	86,19	2,00E-03
GKD-b_47_n150_m45	1309,7789	82,55	1,00E-03
GKD-b_48_n150_m45	1218,6844	81,39	2,00E-03
GKD-b_49_n150_m45	2113,7594	89,29	1,00E-03
GKD-b_50_n150_m45	1644,7062	84,87	2,00E-03

5.1.2 Búsqueda Local

Búsqueda Local			
Caso	**Coste medio obtenido**	**Desv**	**Tiempo**
GKD-b_1_n25_m2	0,0000	0,00	0,591
GKD-b_2_n25_m2	0,0000	0,00	0,6
GKD-b_3_n25_m2	0,0000	0,00	0,602
GKD-b_4_n25_m2	0,0000	0,00	0,59
GKD-b_5_n25_m2	0,0000	0,00	0,5901
GKD-b_6_n25_m7	219,2392	94,20	1,102
GKD-b_7_n25_m7	163,7637	91,39	1,081
GKD-b_8_n25_m7	119,6398	85,99	1,088
GKD-b_9_n25_m7	86,3641	80,24	1,111
GKD-b_10_n25_m7	166,5597	86,03	1,084
GKD-b_11_n50_m5	88,7450	97,83	1,483
GKD-b_12_n50_m5	98,9776	97,86	1,481
GKD-b_13_n50_m5	65,6528	96,40	1,488
GKD-b_14_n50_m5	147,1714	98,87	1,476
GKD-b_15_n50_m5	59,0091	95,16	1,488
GKD-b_16_n50_m15	354,1333	87,93	2,691
GKD-b_17_n50_m15	309,5614	84,46	2,643
GKD-b_18_n50_m15	391,3675	88,96	2,893
GKD-b_19_n50_m15	351,2557	86,79	2,681
GKD-b_20_n50_m15	475,9670	89,98	2,659
GKD-b_21_n100_m10	248,3606	94,43	3,8089
GKD-b_22_n100_m10	272,9594	94,99	3,813
GKD-b_23_n100_m10	169,7734	90,96	3,818
GKD-b_24_n100_m10	277,4251	96,89	3,876
GKD-b_25_n100_m10	214,4569	91,98	3,8111
GKD-b_26_n100_m30	1076,2283	84,32	7,067
GKD-b_27_n100_m30	1123,5965	88,69	7,0836
GKD-b_28_n100_m30	1453,7566	92,68	7,282
GKD-b_29_n100_m30	1008,6594	86,37	7,204
GKD-b_30_n100_m30	1111,3264	88,53	7,168
GKD-b_31_n125_m12	286,2256	95,90	5,49
GKD-b_32_n125_m12	261,1290	92,80	5,416
GKD-b_33_n125_m12	380,9827	95,14	5,361
GKD-b_34_n125_m12	278,3220	93,00	5,386
GKD-b_35_n125_m12	506,0794	96,42	5,3861
GKD-b_36_n125_m37	1405,9188	88,94	10,05
GKD-b_37_n125_m37	1652,8848	87,97	9,875
GKD-b_38_n125_m37	1319,3359	85,75	9,794
GKD-b_39_n125_m37	1254,2883	86,56	9,783
GKD-b_40_n125_m37	1469,1299	87,87	10,1068
GKD-b_41_n150_m15	443,8840	94,74	7,467
GKD-b_42_n150_m15	467,1423	94,27	7,367
GKD-b_43_n150_m15	379,4768	92,95	7,265
GKD-b_44_n150_m15	310,1372	91,64	7,2299
GKD-b_45_n150_m15	513,4070	94,59	7,231
GKD-b_46_n150_m45	1649,4039	86,19	13,422
GKD-b_47_n150_m45	1759,6606	87,01	13,507
GKD-b_48_n150_m45	1518,6448	85,07	13,719
GKD-b_49_n150_m45	1666,1065	86,41	13,48
GKD-b_50_n150_m45	1546,0282	83,90	13,372

5.1.3 AGG-Uniforme

AGG-Uniforme			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	2,974
GKD-b_2_n25_m2	0,0000	0,00	2,956
GKD-b_3_n25_m2	0,0000	0,00	2,965
GKD-b_4_n25_m2	0,0000	0,00	2,963
GKD-b_5_n25_m2	0,0000	0,00	2,976
GKD-b_6_n25_m7	74,8144	83,00	4,497
GKD-b_7_n25_m7	80,8707	82,57	4,471
GKD-b_8_n25_m7	69,9507	76,04	4,464
GKD-b_9_n25_m7	92,5260	81,55	4,497
GKD-b_10_n25_m7	100,8670	76,93	4,47
GKD-b_11_n50_m5	13,2540	85,47	5,254
GKD-b_12_n50_m5	19,5872	89,17	5,258
GKD-b_13_n50_m5	14,1291	83,28	5,2301
GKD-b_14_n50_m5	7,6547	78,27	5,185
GKD-b_15_n50_m5	10,9404	73,92	5,222
GKD-b_16_n50_m15	271,3311	84,25	11,795
GKD-b_17_n50_m15	180,7586	73,39	11,8013
GKD-b_18_n50_m15	251,6083	82,83	12,196
GKD-b_19_n50_m15	269,3550	82,77	11,873
GKD-b_20_n50_m15	154,0753	69,03	11,751
GKD-b_21_n100_m10	67,4312	79,49	14,4733
GKD-b_22_n100_m10	39,8633	65,72	14,404
GKD-b_23_n100_m10	45,7973	66,49	14,321
GKD-b_24_n100_m10	35,7531	75,83	14,555
GKD-b_25_n100_m10	33,0084	47,89	14,5361
GKD-b_26_n100_m30	608,9846	72,29	40,553
GKD-b_27_n100_m30	571,7437	77,77	40,1105
GKD-b_28_n100_m30	572,4173	81,42	40,0996
GKD-b_29_n100_m30	520,7319	73,60	40,9489
GKD-b_30_n100_m30	529,2084	75,91	40,5481
GKD-b_31_n125_m12	286,2256	95,90	20,361
GKD-b_32_n125_m12	59,2606	68,29	20,4839
GKD-b_33_n125_m12	35,7674	48,19	19,868
GKD-b_34_n125_m12	45,8020	57,45	19,8523
GKD-b_35_n125_m12	50,0081	63,78	20,6016
GKD-b_36_n125_m37	708,0342	78,05	61,4504
GKD-b_37_n125_m37	970,7096	79,51	60,3176
GKD-b_38_n125_m37	642,3276	70,74	61,111
GKD-b_39_n125_m37	725,0881	76,75	61,7809
GKD-b_40_n125_m37	771,2041	76,89	61,5167
GKD-b_41_n150_m15	55,6479	58,05	28,779
GKD-b_42_n150_m15	98,1470	72,70	29,336
GKD-b_43_n150_m15	151,4296	82,33	28,66
GKD-b_44_n150_m15	115,9388	77,63	29,14
GKD-b_45_n150_m15	43,5925	36,29	28,8299
GKD-b_46_n150_m45	973,9567	76,62	86,8323
GKD-b_47_n150_m45	886,8199	74,22	87,9321
GKD-b_48_n150_m45	803,8482	71,79	87,367
GKD-b_49_n150_m45	530,4122	57,31	89,764
GKD-b_50_n150_m45	853,6890	70,85	87,8932

5.1.4 AGG-Posicion

AGG-Posicion			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	2,898
GKD-b_2_n25_m2	0,0000	0,00	2,842
GKD-b_3_n25_m2	0,0000	0,00	2,862
GKD-b_4_n25_m2	0,0000	0,00	2,836
GKD-b_5_n25_m2	0,0000	0,00	2,846
GKD-b_6_n25_m7	76,2824	83,33	4,191
GKD-b_7_n25_m7	74,7188	81,13	4,1949
GKD-b_8_n25_m7	124,1413	86,50	4,204
GKD-b_9_n25_m7	105,1537	83,77	4,209
GKD-b_10_n25_m7	89,4936	74,00	4,217
GKD-b_11_n50_m5	33,0282	94,17	4,877
GKD-b_12_n50_m5	55,8351	96,20	4,891
GKD-b_13_n50_m5	38,4525	93,86	4,919
GKD-b_14_n50_m5	120,9695	98,63	4,808
GKD-b_15_n50_m5	58,8991	95,16	4,935
GKD-b_16_n50_m15	350,0358	87,79	10,82
GKD-b_17_n50_m15	194,0424	75,21	10,809
GKD-b_18_n50_m15	284,7096	84,83	11,2848
GKD-b_19_n50_m15	253,3548	81,68	10,83
GKD-b_20_n50_m15	371,0856	87,14	10,842
GKD-b_21_n100_m10	179,2024	92,28	13,194
GKD-b_22_n100_m10	181,3850	92,47	13,204
GKD-b_23_n100_m10	162,2560	90,54	13,192
GKD-b_24_n100_m10	193,0696	95,52	13,266
GKD-b_25_n100_m10	155,2538	88,92	13,193
GKD-b_26_n100_m30	725,8654	76,75	34,7328
GKD-b_27_n100_m30	695,2123	81,72	34,6631
GKD-b_28_n100_m30	1227,0194	91,33	34,6737
GKD-b_29_n100_m30	797,4192	82,76	35,234
GKD-b_30_n100_m30	817,3028	84,40	35,2663
GKD-b_31_n125_m12	190,7993	93,84	18,5199
GKD-b_32_n125_m12	276,3633	93,20	18,64
GKD-b_33_n125_m12	243,9167	92,40	18,602
GKD-b_34_n125_m12	248,0781	92,14	18,5371
GKD-b_35_n125_m12	218,1612	91,70	18,515
GKD-b_36_n125_m37	998,3902	84,43	53,845
GKD-b_37_n125_m37	1141,4542	82,58	51,881
GKD-b_38_n125_m37	987,3674	80,96	52,123
GKD-b_39_n125_m37	872,1589	80,67	53,1053
GKD-b_40_n125_m37	897,9237	80,15	55,3626
GKD-b_41_n150_m15	518,6492	95,50	25,6174
GKD-b_42_n150_m15	301,6056	91,12	26,095
GKD-b_43_n150_m15	400,6110	93,32	25,056
GKD-b_44_n150_m15	553,8535	95,32	25,204
GKD-b_45_n150_m15	262,9860	89,44	25,861
GKD-b_46_n150_m45	1891,5239	87,96	75,2359
GKD-b_47_n150_m45	1348,7456	83,05	74,8619
GKD-b_48_n150_m45	794,7450	71,47	74,8287
GKD-b_49_n150_m45	1228,1454	81,56	75,365
GKD-b_50_n150_m45	1281,4764	80,58	76,2464

5.1.5 AGE-Uniforme

AGE-Uniforme			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	1,971
GKD-b_2_n25_m2	0,0000	0,00	1,9521
GKD-b_3_n25_m2	0,0000	0,00	1,9466
GKD-b_4_n25_m2	0,0000	0,00	1,942
GKD-b_5_n25_m2	0,0000	0,00	1,942
GKD-b_6_n25_m7	73,9186	82,79	2,177
GKD-b_7_n25_m7	61,0219	76,90	2,198
GKD-b_8_n25_m7	102,7004	83,68	2,2047
GKD-b_9_n25_m7	104,8629	83,72	2,2049
GKD-b_10_n25_m7	134,4448	82,70	2,188
GKD-b_11_n50_m5	37,0803	94,81	2,578
GKD-b_12_n50_m5	21,3000	90,04	2,5871
GKD-b_13_n50_m5	33,3075	92,91	2,5797
GKD-b_14_n50_m5	47,4376	96,49	2,599
GKD-b_15_n50_m5	32,2685	91,16	2,58
GKD-b_16_n50_m15	333,0961	87,17	5,654
GKD-b_17_n50_m15	222,7528	78,40	5,64
GKD-b_18_n50_m15	280,0132	84,57	5,629
GKD-b_19_n50_m15	245,5388	81,10	5,606
GKD-b_20_n50_m15	318,1157	85,00	5,655
GKD-b_21_n100_m10	133,4677	89,64	6,846
GKD-b_22_n100_m10	158,5949	91,38	6,841
GKD-b_23_n100_m10	124,0385	87,63	6,843
GKD-b_24_n100_m10	154,6098	94,41	6,841
GKD-b_25_n100_m10	100,6732	82,91	6,842
GKD-b_26_n100_m30	742,3174	77,27	18,1015
GKD-b_27_n100_m30	749,7127	83,05	19,047
GKD-b_28_n100_m30	977,5269	89,12	17,993
GKD-b_29_n100_m30	764,0183	82,01	18,076
GKD-b_30_n100_m30	841,9627	84,86	18,132
GKD-b_31_n125_m12	237,2886	95,05	9,442
GKD-b_32_n125_m12	170,1991	88,96	9,464
GKD-b_33_n125_m12	204,9625	90,96	9,465
GKD-b_34_n125_m12	228,1493	91,46	9,6661
GKD-b_35_n125_m12	188,9535	90,41	9,552
GKD-b_36_n125_m37	1100,0398	85,87	27,126
GKD-b_37_n125_m37	1297,5432	84,67	27,248
GKD-b_38_n125_m37	991,8775	81,05	27,114
GKD-b_39_n125_m37	998,1739	83,11	27,227
GKD-b_40_n125_m37	973,0992	81,69	27,4735
GKD-b_41_n150_m15	280,0107	91,66	13,894
GKD-b_42_n150_m15	306,5241	91,26	13,445
GKD-b_43_n150_m15	305,6898	91,25	13,463
GKD-b_44_n150_m15	282,2536	90,81	13,5098
GKD-b_45_n150_m15	255,9421	89,15	13,399
GKD-b_46_n150_m45	1326,2085	82,83	38,982
GKD-b_47_n150_m45	1268,2212	81,97	39,1886
GKD-b_48_n150_m45	1195,0361	81,03	39,329
GKD-b_49_n150_m45	1454,8920	84,44	39,8213
GKD-b_50_n150_m45	1464,4910	83,01	39,1445

5.1.6 AGE-Posición

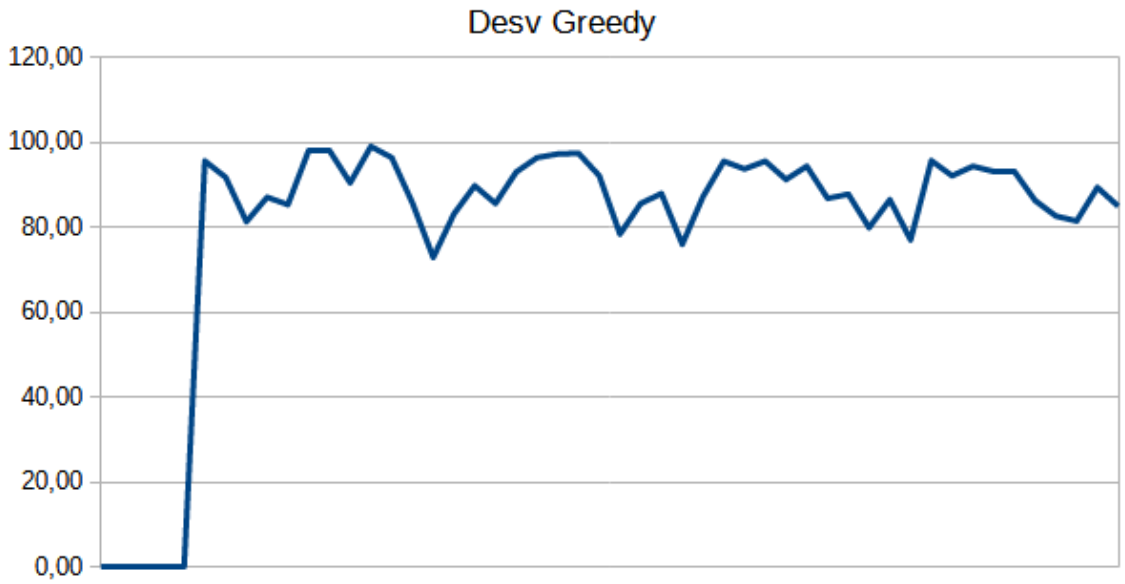
AGE-Posicion			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	1,971
GKD-b_2_n25_m2	0,0000	0,00	1,9521
GKD-b_3_n25_m2	0,0000	0,00	1,9466
GKD-b_4_n25_m2	0,0000	0,00	1,942
GKD-b_5_n25_m2	0,0000	0,00	1,942
GKD-b_6_n25_m7	152,9146	91,68	2,161
GKD-b_7_n25_m7	192,2849	92,67	2,188
GKD-b_8_n25_m7	161,7305	89,64	2,1723
GKD-b_9_n25_m7	112,6508	84,85	2,165
GKD-b_10_n25_m7	201,6346	88,46	2,178
GKD-b_11_n50_m5	85,0464	97,74	2,584
GKD-b_12_n50_m5	98,7243	97,85	2,577
GKD-b_13_n50_m5	87,7485	97,31	2,566
GKD-b_14_n50_m5	38,5560	95,69	2,556
GKD-b_15_n50_m5	284,3752	99,00	2,581
GKD-b_16_n50_m15	454,7675	90,60	5,6041
GKD-b_17_n50_m15	396,3067	87,86	5,605
GKD-b_18_n50_m15	449,2241	90,38	5,599
GKD-b_19_n50_m15	431,2158	89,24	5,633
GKD-b_20_n50_m15	433,2601	88,99	5,603
GKD-b_21_n100_m10	323,7269	95,73	6,853
GKD-b_22_n100_m10	274,5058	95,02	6,853
GKD-b_23_n100_m10	355,1992	95,68	6,847
GKD-b_24_n100_m10	445,2387	98,06	6,861
GKD-b_25_n100_m10	176,0255	90,23	6,869
GKD-b_26_n100_m30	763,9938	77,91	18,074
GKD-b_27_n100_m30	1127,0154	88,72	18,3195
GKD-b_28_n100_m30	1680,4851	93,67	18,1007
GKD-b_29_n100_m30	1090,4042	87,39	18,197
GKD-b_30_n100_m30	1043,5822	87,78	18,095
GKD-b_31_n125_m12	296,2320	96,04	9,425
GKD-b_32_n125_m12	376,0029	95,00	9,416
GKD-b_33_n125_m12	273,6207	93,23	9,43
GKD-b_34_n125_m12	293,8184	93,37	9,4259
GKD-b_35_n125_m12	376,0691	95,18	9,458
GKD-b_36_n125_m37	1760,5835	91,17	27,161
GKD-b_37_n125_m37	1262,2924	84,24	27,172
GKD-b_38_n125_m37	1387,6302	86,45	27,765
GKD-b_39_n125_m37	1283,2174	86,86	27,3719
GKD-b_40_n125_m37	1373,2375	87,02	27,3176
GKD-b_41_n150_m15	323,3688	92,78	13,4925
GKD-b_42_n150_m15	446,5429	94,00	13,429
GKD-b_43_n150_m15	415,8870	93,57	13,398
GKD-b_44_n150_m15	341,2394	92,40	13,3877
GKD-b_45_n150_m15	292,9518	90,52	13,6
GKD-b_46_n150_m45	1566,9430	85,47	39,078
GKD-b_47_n150_m45	1423,4978	83,94	39,1414
GKD-b_48_n150_m45	1440,7659	84,26	39,575
GKD-b_49_n150_m45	1824,4049	87,59	39,3381
GKD-b_50_n150_m45	1857,4849	86,60	39,0691

5.1.7 Medias

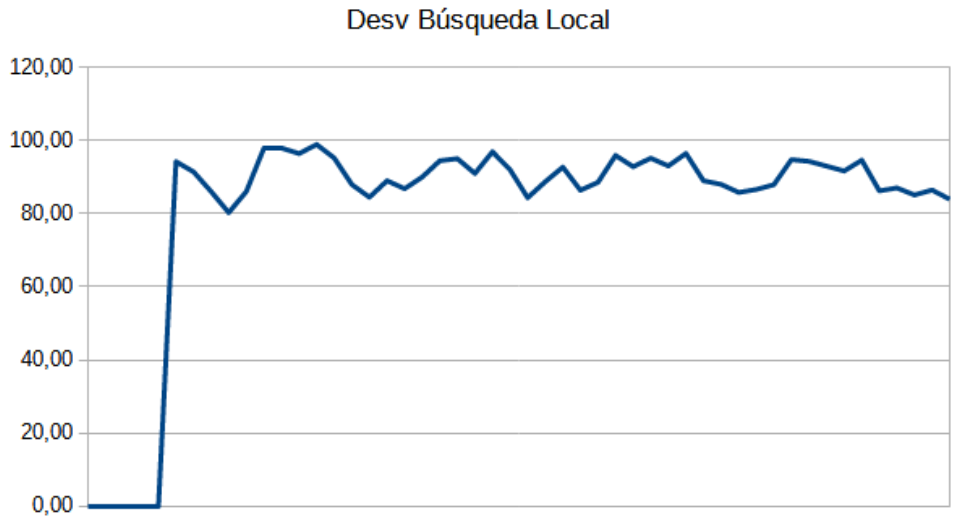
Algoritmo	Desviación	Tiempo (s)
Greedy	80.18	$4.8 \cdot 10^{-4}$
BL	81.58	5.3
AGG-Uniforme	66.24	20.78
AGG-Posición	78.43	24.4
AGE-Uniforme	77.89	12.7
AGE-Posición	81.84	12.7

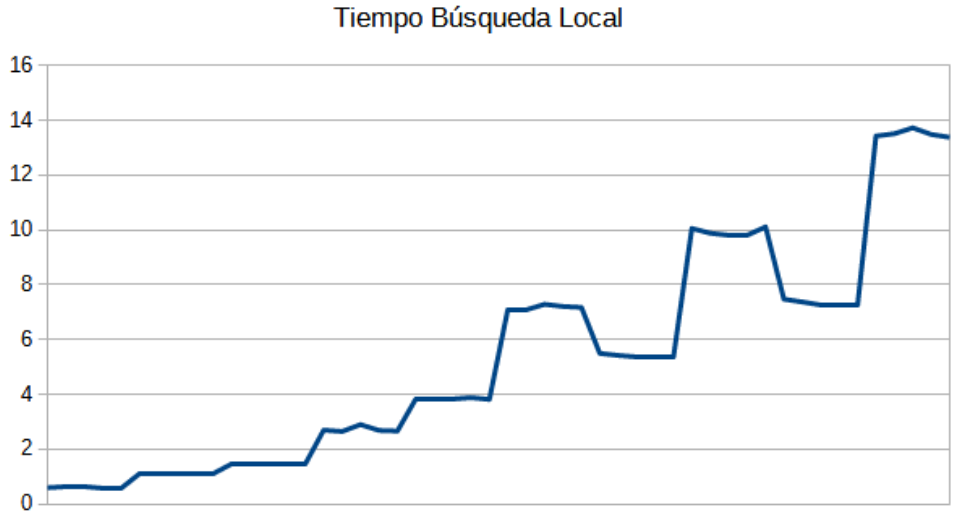
5.2 Gráficas

5.2.1 Greedy

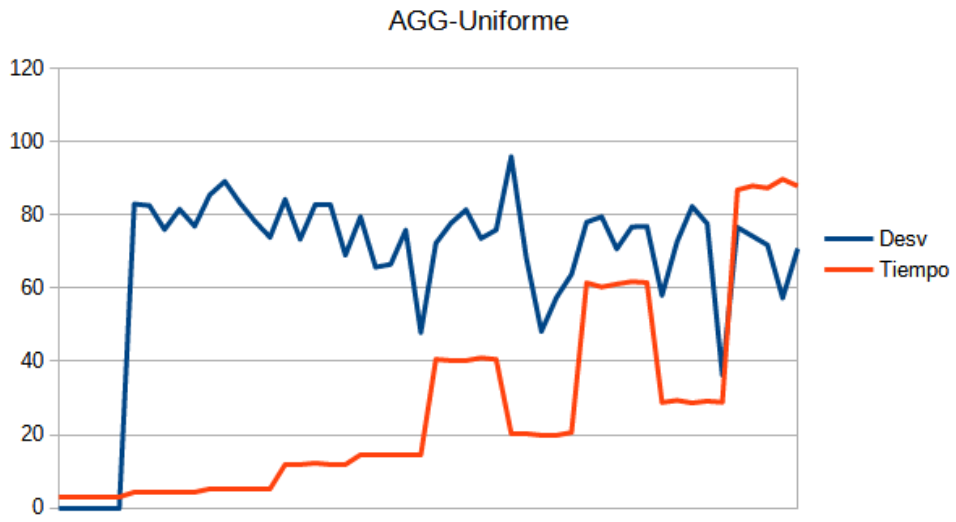


5.2.2 Búsqueda Local

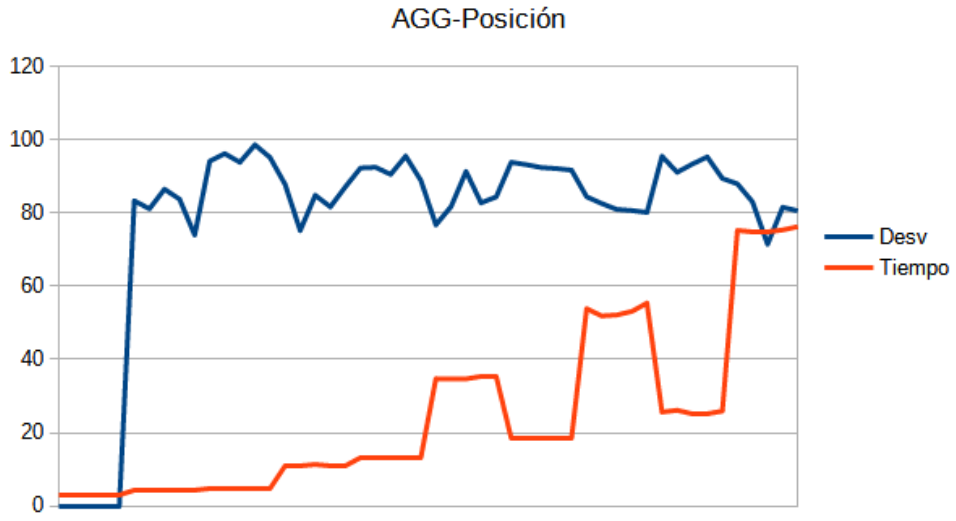




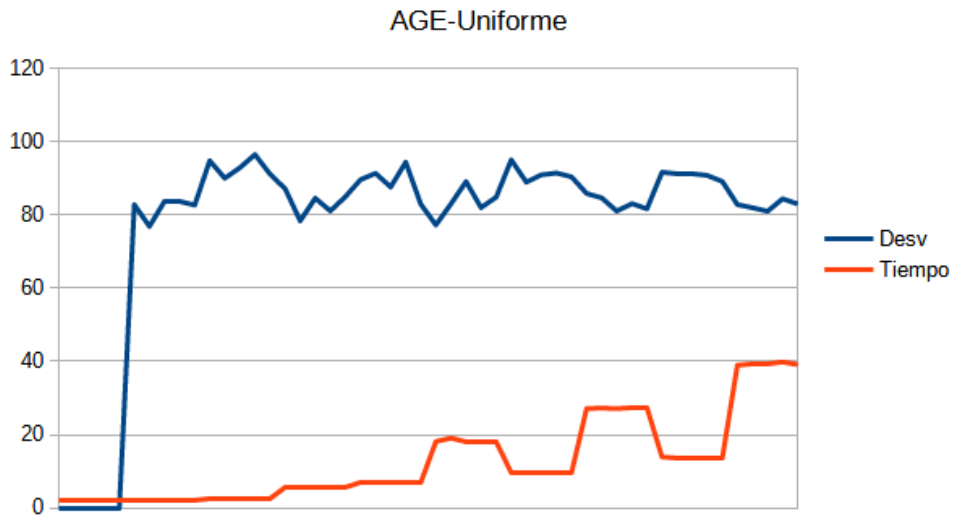
5.2.3 AGG-Uniforme



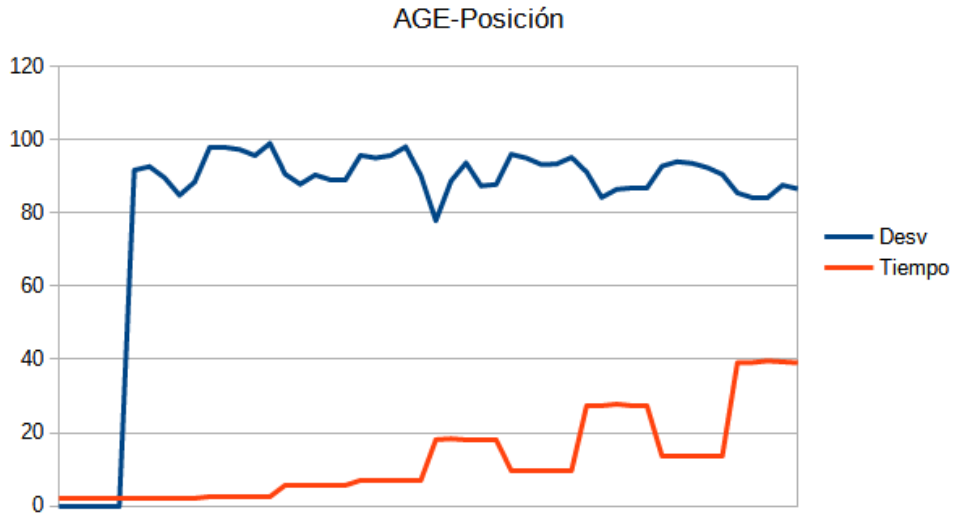
5.2.4 AGG-Posición



5.2.5 AGE-Uniforme



5.2.6 AGE-Posición



5.3 Análisis

Por desgracia, como mis resultados obtenidos son **pésimos**, *haré el análisis solamente en base a mis resultados y a mis implementaciones. Me gustaría sacar una conclusión real, pero solo puedo limitarme a hacer comparaciones y mencionar posibles errores.*

El algoritmo Greedy debería de haber dado una desviación mayor que la BL. Los AGE en general deberían ser mejores que los AGG al ser constantes en cuanto a mejorar, además, sus tiempos son prácticamente idénticos. Podría decir que si puede notarse la diferencia entre el cruce uniforme y el de posición, pero poco.

¿Cómo puede ser que haya implementado peor el AGE-Posición que la BL?

En los AGG, a partir del caso 25, los tiempos empiezan a dispararse, llegando a superar el minuto por caso (esto se nota mucho cuando m es grande)

Si dieran buenos resultados tendría un pase, pero a ojo, la desviación media de todos los algoritmos es ≈ 80, que es demasiado. Esto es igual solamente para los genéticos. El que mejor lo hace de todos es el AGG-Uniforme (en un caso, puede que el 35, bajó del 40 de desviación)

Lo único que hacen bien todos los algoritmos es en hacer bien los primeros 5 casos.