Práctica 3.a:
Búsquedas por Trayectorias para el Problema de la Mínima Dispersión Diferencial (MDD)

Juan Antonio Martínez Sánchez

Curso 2021/22

?contentsname?

1 Introducción

El objetivo de esta práctica es estudiar el funcionamiento de las Técnicas de Búsqueda basadas en Trayectorias (tanto simples como múltiples) en la resolución del problema de la mínima dispersión diferencial (MDD). Para ello, se veremos las siguientes técnicas metaheurísticas:

- Enfriamiento Simulado (ES).
- Búsqueda Multiarranque Básica (BMB).
- Búsqueda Local Reiterada (ILS).
- Hibridación de ILS y ES (ILS-ES).

Se utilizarán 50 casos seleccionados de varios de los conjuntos de instancias todas disponibles en la MDPLIB, pertenecientes al grupo GDK-b con distancias aleatorias reales con, n entre $\{25, 50, 75, 100, 125, 150\}$, y m entre 2 y 45 (GDK-bGKD-b_1_n25_m2.txt a GKD-b_50_n150_m45.txt)

La función objetivo para el poblema vendrá dada por la función $CalculaDispersion()$ implementada en el archivo utilidades.cpp. Esta función en base a un vector con la suma de las distancias entre todos los puntos de la solución calculará la diferencia entre el máximo y mínimo que será la dispersión generada entre todos los puntos seleccionados. El vector de distancias comentado anteriormente se calculará con la función $SumaDistancias()$ de utilidades.cpp.

2 Aplicación de los Algoritmos

2.1 Enfriamiento Simulado (ES)

Este algoritmo es un modo de evitar caer en óptimos locales como ocurre en Búsqueda Local y una solución es permitir que algunos movimientos sean hacia soluciones peores pero de manera controlada para no desviarnos demasiado de un buen camino hacia una solución. En el caso del Enfriamiento Simulado (ES), esto se realiza controlando la frecuencia de los movimientos de escape mediante una función de probabilidad que hará disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda.
La implementación consistirá en primeramente en generar una solución válida aleatoria de m puntos seleccionados, calcular su coste e inicializar la temperatura inicial y otras variables necesarias como el máximo de vecinos a explorar o el máximo de éxitos por edad de enfriamiento y M que sera el número de enfriamientos. Seguidamente mientras no se cumpla el criterio de parada de que la solución no mejor en toda la edad, que la temperatura actual sea mayor que la final o que se superen 100000 evaluaciones haremos lo siguiente. En cada edad de enfriamiento (cada iteracion) Generaremos un vecino aleatorio de la solución actual la cual será intercambiando un elemento seleccionado en la solución por otro no seleccionado. A este vecino generado le calcularemos su coste y haremos la diferencia con el coste de la mejor solución y si esta la mejora, actualizaremos la mejor solucion y el mejor coste. Sin embargo, también tendremos otro criterio de aceptación en el que aceptaremos si se cumple $U(0,1) \leq e^{-\Delta_{costes}/T_k}$. Una vez alcanzado el máximo de vecinos a explorar o el máximo de éxitos, procederemos a enfriar la temperatura actual, comprobaremos si hay alguna mejora y volveremos a repetir el procedimiento.

Pseudocódigo ES:

```
ES(n, m, D) {
    T_final = 0.001
    GenerarSolucionAleatoria( solucion )
    coste = CalcularCoste( solucion )
    Inicializar T_inicial
    T_actual = T_inicial

    while T_inicial < T_final AND evaluaciones < 100000 AND mejora do:
        while vecinos < max_vecinos AND exitos < max_exitos do:
            solucion_siguiente = GenerarVecinoAleatorio( solucion )
            vecinos++

            coste_siguiente = CalcularCoste( solucion_siguiente )
            diferencia = coste_siguiente - coste

            if diferencia < 0 OR U(0,1) <= exp(-diferencia/T_actual) :
                coste = coste_siguiente
                solucion = solucion_siguiente
                exitos++
        end
        Enfriar( T_actual )

        if exitos == 0 :
            !mejora

        vecinos = 0
        exitos = 0
    end

    return solucion
}
```

Para calcular la temperatura inicial utilizaremos la siguiente fórmula:

$T_0 = \frac{\mu \cdot C\{S_0\}}{-ln\{\phi\}}$
Donde μ y ϕ valdrán 0.3 y $C\{S_0\}$ representa el coste de la solución inicial.

Para el esquema de enfriamiento emplearemos el esdquema de Cauchy modificado:

$T_{k+1} = \frac{T_k}{1+\beta \cdot T_k}$ donde $\beta = \frac{T_0 - T_f}{M \cdot T_0 \cdot T_f}$
En este caso M es el número de enfriamientos (iteraciones) a realizar, T_0 es la temperatura inicial y T_f la temperatura final que tendrá un valor cercano a cero. T_k es la temperatura actual en cada iteración.
Este esquema estará implementado en la función $Enfriar()$.

2.2 Búsqueda Local (Práctica 1)

Como algoritmo de BL para el MDD consideraremos el esquema del primer mejor. La representación será en forma de un conjunto de elementos seleccionados. Para ser una solución candidata válida, tiene que satisfacer las restricciones (ser un conjunto de tamaño m):

- No puede tener elementos repetidos.
- Ha de contener exactamente m elementos.
- El orden de los elementos no es relevante.

El entorno de una solución Sel está formado por las soluciones accesibles desde ella a través de un movimiento de intercambio.
Se utilizará una función de itercambio de puntos $IntercambiarPunto((i,j), Sel, d_{Sel}, MatrizDistancias)$ donde se intercambiará el punto $i \in Sel$, por el punto $j \notin Sel$, y al mismo tiempo se actualizarán las distancias entre los puntos seleccionados d restando las distancias con i y añadiendo la nueva distancia con j a través de la $MatrizDistancias$. A esto último lo denominamos *factorización* de la función objetivo en todos los casos. Una vez realizado el movimiento, se actualiza la solución actual y los valores de contribución de los elementos seleccionados al coste de dicha solución, y se comienza a explorar el nuevo entorno. La exploración del entorno se realizará en orden aleatorio.

En cada ejecución de la BL, se partirá de una solución inicial aleatoria y se detendrá la ejecución cuando no se encuentre mejora en todo el entorno o cuando se hayan realizado 100000 evaluaciones de la función objetivo, es decir, en cuanto se cumpla alguna de las dos condiciones.

La BL del Mejor explora todo el vecindario, las soluciones resultantes de los $m(n-m)$ intercambios posibles. Una vez generado el vecindario, aleatoriamente se irán seleccionando las soluciones resultantes y se comprobará si la dispersión mejora y si es así actualizaremos la solución actual por la nueva obtenida y volveremos a generar el nuevo vecindario y repetir el proceso. En cambio, si no mejora en ningún caso pararemos el proceso y nos quedaremos con la última solución obtenida.

Pseudocódigo BL:

ALGORITMO BL:

void $BL(Sel,\ n,\ m,\ MatrizDistancias,\ dispersion)$
1 : $restantes \leftarrow$ *todo* $p \notin Sel$
2 : $distancias \leftarrow SumaDistancias(Sel,\ m,\ M)$
3 : $dispersion \leftarrow CalculaDispersion(distancias)$
4 : $Sel_{aux} \leftarrow Sel$
5 : $distancias_{aux} \leftarrow distancias$
6 : **while** $evaluaciones < 100000$ && $!mejora$ **do**
7 : **for** *para cada punto* $u \in Sel$
8 : **for** *para cada punto* $v \in restantes$
9 : $Vecindario \leftarrow$ *tupla* (u, v)
10 : $MezclarVecindario()$
11 : **for** *para cada tuplat* $(u, v) \in Vecindario$
12 : $IntercambiarPunto(t_i,\ Sel_{aux},\ distancias_{aux},\ MatrizDistancias)$
13 : $dispersion_{candidata} \leftarrow CalculaDispersion(distancias_{aux})$
14 : **if** $dispersion_{candidata} > dispersion$
15 : $Sel \leftarrow Sel_{aux}$
16 : $distancias \leftarrow distancias_{aux}$
17 : $dispersion \leftarrow dispersion_{candidata}$
18 : $restantes \leftarrow restantes \bigcup \{u\}$
19 : $restantes \leftarrow restantes \setminus \{v\}$
20 : **else**
21 : $Sel_{aux} \leftarrow Sel$
22 : $distancias_{aux} \leftarrow distancias$
23 : $evaluaciones++$
24 : **end while**

Pseudocódigo Función IntercambiaPunto:

IntercambiaPunto:

void $IntercambiaPunto(tupla(u, v),\ Sel,\ distancias,\ M)$
1 : **for** $i < Sel.size()$
2 : **if** $Sel[i] == u$
3 : $d_u \leftarrow distancias[i]$
4 : **else**
5 : $distancias[i] \leftarrow distancias[i] - d_{Sel[i],u}$
6 : $distancias[i] \leftarrow distancias[i] + d_{Sel[i],v}$
7 : $dtotal_v \leftarrow dtotal_v + d_{Sel[i],v}$
8 : $Sel \leftarrow Sel \backslash \{u\}$
9 : $Sel \leftarrow Sel \bigcup \{v\}$
10 : $distancias \leftarrow Sel \backslash \{d_u\}$
11 : $distancias \leftarrow Sel \bigcup \{dtotal_v\}$

La función consiste en dada una tupla (u, v) siendo $u \in Sel$ y $v \notin Sel$, un vector de distancias, un vector de seleccionados (Sel) y la matriz de distancias que intercambie u por v y actualice las distancias utilizando al factorización, es decir, que a la distancia total de cada punto con el resto se le vaya restando la distancia con el punto u y se le sume la correspondiente con v a través de la matriz de distancias M.

Entendiendo esta función podremos explicar el funcionamiento de la función de Búsqueda Local. Al comienzo generaremos el conjunto de puntos no seleccionados iniciales llamado *restantes*, seguidamente calcularemos la suma de las distancias de todos los puntos de Sel con todos mediante la función $SumaDistancias()$ que veremos más tarde, y calcularemos la dispersión con la función $CalculaDispersion()$.

Mientras no lleguemos a 100000 evaluaciones o no mejoremos la dispersión repetiremos el siguiente proceso:

1. Crearemos el *Vecindario* con todas las posibles combinaciones de intercambios a través de

un vector de tuplas (u, v) ($u \in Sel$ y $v \notin Sel$), donde u será intercambiado en el conjunto de seleccionados (Sel) por v.

2. Mezclaremos las tuplas con la función $random\_shuffle$ de la libreria *algorithm* para que en el siguiente paso al seleccionar tuplas haya aleatoriedad.

3. Seleccionaremos una tupla hasta que tras llamar a $IntercambiarPunto$ y seguidamente a $CalculaDispersion$ comprobemos si la dispersión mejora.

    - En el caso de que mejore actualizaremos la dispersión actual a la última calculada, el conjunto Sel al nuevo con el intercambio de la tupla y el vector nuevo de distancias. Cambiamos mejora a $True$ para seguir generando el nuevo *Vecindario* y paremos de examinar el actual. Finalmente a *restantes* añadiremos v y eliminaremos u.
    - En el caso de que no mejore restableceremos los cambios realizados en las variables auxiliares Sel_{aux} y $distancias_{aux}$ con la última solución.

4. Por cada tupla que examinamos le aplicamos la función de evaluación, es decir, hemos realizado una evaluación por tanto incrementamos su variable en 1.

2.3 Búsqueda Multiarranque Básica (BMB)

El algoritmo BMB consistirá simplemente en generar un determinado número de soluciones aleatorias iniciales y optimizar cada una de ellas con el algoritmo de BL considerado. Se devolverá la mejor solución encontrada en todo el proceso.
La implementación consiste en generar una solución aleatoria con la función *GenerarSolucionAleatoria*(), aplicarle por Búsqueda Local y ver si mejor a la mejor solución. Este proceso lo repetiremos un total de 10 veces con un máximo de 10000 iteraciones en la BL.

Pseudocódigo BMB:

```
BMB(n, m, M) {

    GenerarSolucionAleatoria( solucion )
    coste = CalcularCoste( solucion )

    MEJOR SOLUCION = solucion
    MEJOR COSTE = coste

    while iteraciones < 10 do:
        BL( solucion, coste )

        if coste < MEJOR COSTE :
            Actualizar MEJOR SOLUCION

        GenerarSolucionAleatoria( solucion )
    end

    return MEJOR SOLUCION
}
```

2.4 Búsqueda Local Reiterada (ILS)

El algoritmo ILS consistirá en generar una solución inicial aleatoria y aplicar el algoritmo de BL sobre ella. Una vez obtenida la solución optimizada, se estudiará si es mejor que la mejor solución encontrada hasta el momento y se realizará una mutación sobre la mejor de estas dos, volviendo a aplicar el algoritmo de BL sobre esta solución mutada. Este proceso se repetirá 10 veces, devolviéndose la mejor solución encontrada en toda la ejecución. Por tanto, se seguirá el criterio del mejor como criterio de aceptación de la ILS.
El algoritmo está implementado de manera que generaremos una solución aleatoria inicial a la cual le aplicaremos BL y la guardaremos como la mejor solución. Esto último lo consideraremos la primera iteración ya que es la primera llamada a la BL, por tanto procederemos a mutar la mejor solucion, a aplicarle BL y a comprobar si esta solución mutada mejora a la mejor y todo esto lo repetiremos 10 veces.

Pseudocódigo ILS:

```
ILS(n, m, M) {

    GenerarSolucionAleatoria( solucion )

    BL(solucion, coste)

    MEJOR SOLUCION = solucion
    MEJOR COSTE = coste

    while iteraciones < 9 do:

        solucion_mutada = MEJOR SOLUCION
        Mutar( solucion_mutada )

        BL( sol_mutada, coste_mutacion )

        if coste_mutacion < MEJOR COSTE :
            Actualizar MEJOR SOLUCION por solucion_mutada

        iteraciones++
    end

    return MEJOR SOLUCION
}
```

El operador de mutación de ILS estará basado en un operador de vecino para representación de orden que provoque un cambio más brusco en la solución actual que el considerado en la BL. Para ello, usaremos una modificación aleatoria de tamaño fijo t. Este proceso consiste en generar aleatoriamente t posiciones de actual solución y cambiarlos por t valores aleatorios que no estén en la actual solución.
En esta práctica t valdrá $0.3m$. Mi implementación consiste en generar números aleatorios entre 0 y n hasta que tengamos $0.3m$ números que no estén en la solución. Haremos lo mismo para obtener $0.3m$ posiciones diferentes entre 0 y m y finalmente intercambiaremos en estas posiciones los valores anteriormente generados.

Pseudocódigo Operador de Mutacion del ILS:

```
Mutar(solucion, n, m) {

    while cambios.size() < 0.3*m do:
        aleatorio = rand()%n
        if aleatorio NO SELECCIONADO
            cambios.push_back( aleatorio )
    end

    while posiciones.size() < 0.3*m do:
        aleatorio = rand()%n
        if aleatorio NO SE REPITE en posiciones
            posiciones.push_back( aleatorio )
    end

    for i=0 hasta 0.3*m do:
        solucion[posiciones[i]] = cambios[i]
    end

    return solucion
}
```

2.4.1 Función GenerarSolucionAleatoria()

Para generar soluciones válidas aleatorias he diseñado la función *GenerarSolucionAleatoria*() que consiste en generar m numeros aleatorios diferentes entre 0 y n.

Pseudocódigo Operador de Generar Solución:

```
GenerarSolucionAleatoria(n, m) {

    while solucion.size() < m do:
        aleatorio = rand()%n
        if aleatorio NO SE REPITE en solucion
        solucion.push_back( aleatorio )
    end

    return solucion
}
```

2.5 Algoritmo Híbrido ILS-ES

Este algoritmo es idéntico al ILS anteriormente comentado, con el único cambio de que en vez de aplicar BL aplicamos el algoritmo ES implementado en esta práctica.

3 Experimentos y análisis de resultados

Después de experimentar con los 50 casos de datos ejecutados con los diferentes algoritmos descritos anteriormente, ya podemos hacer un análisis y llegar a una conclusión.
Si observamos la tabla de resultados 4 podemos ver que todos los algoritmos de esta práctica obtienen mejores resultados que la Búsqueda Local y Greedy de la práctica 1, aunque las medias de tiempos nos dicen que generalmente son más lentos.
Centrándonos ya en analizar los resultados de esta práctica podemos destacar el algoritmo de Búsqueda Multiarranque Básica (BMB) como el que menos desviación tiene, por tanto sería la mejor opción a escoger si pretendemos tener un algoritmo que nos minimice la desviación. Si observamos los tiempos medios de ejecución el más rápido es el Enfriamiento Simulado (ES) que en comparación a los otros requiere un tiempo menor y sería el mejor si necesitamos un algoritmo óptimo y rápido.
Una vez constrastados los mejores algoritmos en base al mejor minimizando la desviación y al más rápido, vamos a comparar la Búsqueda Local con el Enfriamiento Simulado ya que ambos se basan en técnicas de trayectorias simples. Analizando la Búsqueda Local sabemos que puede caer en óptimos locales y no poder optar a mejores resultados, en cambio el Enfriamiento Simulado si consigue evitarlos. Esto se debe a la incorporación de un nuevo criterio de aceptación que consiste en escoger soluciones peores para poder optar a mejores caminos hacía mejores óptimos locales por lo que podremos mejorar las soluciones de la Búsqueda Local, este procedimiento funcionará en base a la temperatura actual que se tendrá en cada iteración en la que contra más alta sea más probabilidad habrá de que la solución sea aceptada sin que esta mejore a la mejor del momento. La probabilidad de aceptación se irá reduciendo conforme avancemos etapas de enfriamiento. Gracias a esto mejoramos notablemente la desviación como podemos observar en la tabla 4.
Ahora vamos a analizar los resultados de la Búsqueda Multiarranque Básica y la Búsqueda Local Reiterada donde ambas se basan en técnicas de trayectorias múltiples, es decir, en cada iteración se aplicará la Búsqueda Local (en este caso) y nos quedaremos con la mejor solución entre todas las generadas. Ambas tienen la misma filosofía de aplicar la BL en cada iteración pero ILS implementa un operador de mutación para poder explorar aleatoriamente otros vecinos cercanos a la mejor solución, a diferencia de BMB que genera en cada iteración una solución completamente aleatoria. En mi opinión ILS al explorar vecinos cercanos a la mejor solución tras realizar la mutación, implica reducir el área de búsqueda de la solución más óptima, por tanto exploraremos en un área menor. En constraste, la BMB dependerá de la aleatoriedad porque de esta depende obtener una solución dentro de un área del vecindario prometedora donde podemos alcanzar un buen óptimo local o al revés y obtener una solución que se aleja de la mejor. Esto último en los dos casos lo solucionamos repitiendo la idea varias veces y así quedarnos con la mejor.

A continuación voy a insertar imágenes con las tablas resultantes de aplicar cada algoritmo.

Algoritmo BL			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	3,92E-05
GKD-b_2_n25_m2	0,0000	0,00	4,78E-05
GKD-b_3_n25_m2	0,0000	0,00	4,31E-05
GKD-b_4_n25_m2	0,0000	0,00	4,22E-05
GKD-b_5_n25_m2	0,0000	0,00	4,44E-05
GKD-b_6_n25_m7	39,2021	67,56	4,23E-04
GKD-b_7_n25_m7	27,9700	49,59	3,45E-04
GKD-b_8_n25_m7	38,5928	56,57	3,38E-04
GKD-b_9_n25_m7	39,8396	57,16	3,24E-04
GKD-b_10_n25_m7	38,1981	39,09	5,02E-04
GKD-b_11_n50_m5	30,2544	93,63	6,44E-04
GKD-b_12_n50_m5	14,6676	85,54	4,27E-04
GKD-b_13_n50_m5	16,8537	85,98	4,51E-04
GKD-b_14_n50_m5	16,1220	89,68	4,62E-04
GKD-b_15_n50_m5	31,6256	90,98	4,61E-04
GKD-b_16_n50_m15	121,1840	64,73	3,13E-03
GKD-b_17_n50_m15	101,3580	52,54	3,49E-03
GKD-b_18_n50_m15	101,1020	57,27	3,17E-03
GKD-b_19_n50_m15	120,8550	61,60	4,52E-03
GKD-b_20_n50_m15	110,7570	56,92	3,36E-03
GKD-b_21_n100_m10	40,4112	65,77	3,48E-03
GKD-b_22_n100_m10	31,8462	57,09	4,04E-03
GKD-b_23_n100_m10	39,2590	60,91	3,77E-03
GKD-b_24_n100_m10	40,0255	78,41	2,46E-03
GKD-b_25_n100_m10	44,3333	61,20	2,82E-03
GKD-b_26_n100_m30	396,5660	57,45	2,44E-02
GKD-b_27_n100_m30	363,1230	65,00	2,43E-02
GKD-b_28_n100_m30	370,4030	71,28	2,30E-02
GKD-b_29_n100_m30	316,0250	56,51	2,96E-02
GKD-b_30_n100_m30	300,1200	57,52	1,98E-02
GKD-b_31_n125_m12	36,1022	67,47	1,28E-02
GKD-b_32_n125_m12	43,7893	57,09	7,16E-03
GKD-b_33_n125_m12	60,8093	69,53	5,92E-03
GKD-b_34_n125_m12	44,7439	56,44	7,22E-03
GKD-b_35_n125_m12	67,8362	73,30	6,77E-03
GKD-b_36_n125_m37	330,4210	52,96	6,11E-02
GKD-b_37_n125_m37	480,3190	58,59	4,41E-02
GKD-b_38_n125_m37	542,6080	65,36	4,06E-02
GKD-b_39_n125_m37	316,5290	46,74	5,84E-02
GKD-b_40_n125_m37	419,8180	57,55	7,46E-02
GKD-b_41_n150_m15	67,3402	65,33	1,32E-02
GKD-b_42_n150_m15	78,0787	65,69	1,73E-02
GKD-b_43_n150_m15	70,8865	62,26	1,20E-02
GKD-b_44_n150_m15	73,5100	64,72	1,71E-02
GKD-b_45_n150_m15	64,3543	56,84	1,23E-02
GKD-b_46_n150_m45	570,1420	60,05	9,49E-02
GKD-b_47_n150_m45	498,8750	54,18	8,79E-02
GKD-b_48_n150_m45	505,2170	55,12	6,22E-02
GKD-b_49_n150_m45	519,4510	56,41	9,21E-02
GKD-b_50_n150_m45	496,8310	49,91	8,97E-02

Algoritmo Greedy			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	8,02E-07
GKD-b_2_n25_m2	0,0000	0,00	8,97E-04
GKD-b_3_n25_m2	0,0000	0,00	9,90E-04
GKD-b_4_n25_m2	0,0000	0,00	6,93E-04
GKD-b_5_n25_m2	0,0000	0,00	6,00E-07
GKD-b_6_n25_m7	49,2781	74,19	9,59E-05
GKD-b_7_n25_m7	70,0438	79,87	8,20E-05
GKD-b_8_n25_m7	42,8261	60,86	8,19E-05
GKD-b_9_n25_m7	78,3335	78,21	8,14E-05
GKD-b_10_n25_m7	80,0830	70,95	7,71E-05
GKD-b_11_n50_m5	26,7260	92,79	8,42E-05
GKD-b_12_n50_m5	29,4615	92,80	8,66E-05
GKD-b_13_n50_m5	36,7743	93,58	1,00E-04
GKD-b_14_n50_m5	24,6666	93,26	8,13E-05
GKD-b_15_n50_m5	28,2984	89,92	8,24E-05
GKD-b_16_n50_m15	181,7840	76,49	4,50E-04
GKD-b_17_n50_m15	187,2190	74,30	4,82E-04
GKD-b_18_n50_m15	168,9950	74,44	4,86E-04
GKD-b_19_n50_m15	193,3450	76,00	6,10E-04
GKD-b_20_n50_m15	173,2890	72,47	4,51E-04
GKD-b_21_n100_m10	75,3892	81,65	4,89E-04
GKD-b_22_n100_m10	71,4620	80,88	5,04E-04
GKD-b_23_n100_m10	74,1665	79,31	5,43E-04
GKD-b_24_n100_m10	69,8125	87,62	5,60E-04
GKD-b_25_n100_m10	82,0937	79,05	5,84E-04
GKD-b_26_n100_m30	439,7110	61,63	2,96E-03
GKD-b_27_n100_m30	431,9330	70,57	3,83E-03
GKD-b_28_n100_m30	457,5030	76,75	2,72E-03
GKD-b_29_n100_m30	325,5970	57,78	3,14E-03
GKD-b_30_n100_m30	455,2380	72,00	2,68E-03
GKD-b_31_n125_m12	107,1910	89,04	1,25E-03
GKD-b_32_n125_m12	69,6034	73,01	1,21E-03
GKD-b_33_n125_m12	103,7390	82,14	9,14E-04
GKD-b_34_n125_m12	63,9608	69,53	8,60E-04
GKD-b_35_n125_m12	84,8231	78,65	9,63E-04
GKD-b_36_n125_m37	458,5780	66,11	5,30E-03
GKD-b_37_n125_m37	463,8730	57,12	5,73E-03
GKD-b_38_n125_m37	545,3780	65,53	5,20E-03
GKD-b_39_n125_m37	447,0590	62,29	5,58E-03
GKD-b_40_n125_m37	415,8630	57,15	7,40E-03
GKD-b_41_n150_m15	109,1780	78,62	1,39E-03
GKD-b_42_n150_m15	142,6770	81,22	2,04E-03
GKD-b_43_n150_m15	103,7920	74,22	1,78E-03
GKD-b_44_n150_m15	137,1490	81,09	1,80E-03
GKD-b_45_n150_m15	112,6590	75,35	2,03E-03
GKD-b_46_n150_m45	538,0100	57,67	1,12E-02
GKD-b_47_n150_m45	578,9720	60,52	8,91E-03
GKD-b_48_n150_m45	448,7150	49,47	8,22E-03
GKD-b_49_n150_m45	559,0420	59,50	8,48E-03
GKD-b_50_n150_m45	643,3480	61,32	1,05E-02

?figurename? 1: BL y Greedy

Algoritmo ES			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	8,40E-02
GKD-b_2_n25_m2	0,0000	0,00	1,45E-02
GKD-b_3_n25_m2	0,0000	0,00	6,54E-02
GKD-b_4_n25_m2	0,0000	0,00	5,28E-02
GKD-b_5_n25_m2	0,0000	0,00	8,59E-02
GKD-b_6_n25_m7	36,7216	65,37	4,12E-03
GKD-b_7_n25_m7	32,8797	57,12	4,59E-03
GKD-b_8_n25_m7	28,5340	41,26	1,69E-02
GKD-b_9_n25_m7	25,4040	32,81	2,78E-03
GKD-b_10_n25_m7	42,3428	45,06	7,79E-03
GKD-b_11_n50_m5	12,2717	84,30	9,12E-03
GKD-b_12_n50_m5	12,4580	82,97	1,01E-02
GKD-b_13_n50_m5	10,7965	78,12	6,30E-03
GKD-b_14_n50_m5	2,9222	43,08	4,94E-03
GKD-b_15_n50_m5	17,5515	83,74	7,26E-03
GKD-b_16_n50_m15	69,8031	38,76	3,50E-02
GKD-b_17_n50_m15	61,6053	21,91	8,12E-02
GKD-b_18_n50_m15	55,3975	22,03	2,87E-02
GKD-b_19_n50_m15	122,7370	62,19	3,44E-02
GKD-b_20_n50_m15	72,2550	33,96	2,19E-02
GKD-b_21_n100_m10	43,3871	68,12	1,74E-02
GKD-b_22_n100_m10	32,7717	58,30	1,76E-02
GKD-b_23_n100_m10	37,8864	59,50	3,82E-02
GKD-b_24_n100_m10	23,4963	63,23	3,16E-02
GKD-b_25_n100_m10	12,1529	-29,35	2,31E-02
GKD-b_26_n100_m30	247,6940	31,88	1,64E-01
GKD-b_27_n100_m30	177,3260	28,33	1,50E-01
GKD-b_28_n100_m30	208,7000	49,03	1,33E-01
GKD-b_29_n100_m30	226,8780	39,42	5,65E-02
GKD-b_30_n100_m30	194,3690	34,41	1,16E-01
GKD-b_31_n125_m12	42,5033	72,37	8,18E-02
GKD-b_32_n125_m12	48,9725	61,63	3,78E-02
GKD-b_33_n125_m12	33,8659	45,28	2,47E-02
GKD-b_34_n125_m12	49,4737	60,61	2,96E-02
GKD-b_35_n125_m12	18,5817	2,53	3,79E-02
GKD-b_36_n125_m37	298,2660	47,89	2,13E-01
GKD-b_37_n125_m37	236,3300	15,84	7,84E-02
GKD-b_38_n125_m37	303,9630	38,16	1,44E-01
GKD-b_39_n125_m37	239,4440	29,59	1,48E-01
GKD-b_40_n125_m37	372,2760	52,13	1,19E-01
GKD-b_41_n150_m15	53,6625	56,49	3,14E-02
GKD-b_42_n150_m15	36,7950	27,19	3,89E-02
GKD-b_43_n150_m15	41,3984	35,37	4,56E-02
GKD-b_44_n150_m15	48,9275	46,99	6,14E-02
GKD-b_45_n150_m15	46,0309	39,66	5,61E-02
GKD-b_46_n150_m45	416,4510	45,31	2,62E-01
GKD-b_47_n150_m45	322,1300	29,03	3,46E-01
GKD-b_48_n150_m45	306,3560	25,99	3,52E-01
GKD-b_49_n150_m45	513,6810	55,92	1,50E-01
GKD-b_50_n150_m45	372,3470	33,17	2,10E-01

Algoritmo BMB			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	6,77E-04
GKD-b_2_n25_m2	0,0000	0,00	9,15E-04
GKD-b_3_n25_m2	0,0000	0,00	7,62E-04
GKD-b_4_n25_m2	0,0000	0,00	6,76E-04
GKD-b_5_n25_m2	0,0000	0,00	6,76E-04
GKD-b_6_n25_m7	12,7180	0,00	6,82E-03
GKD-b_7_n25_m7	20,9563	32,72	7,81E-03
GKD-b_8_n25_m7	16,7612	0,00	7,90E-03
GKD-b_9_n25_m7	19,9569	14,47	6,08E-03
GKD-b_10_n25_m7	32,3825	28,15	5,26E-03
GKD-b_11_n50_m5	8,2062	76,53	6,52E-03
GKD-b_12_n50_m5	8,2329	74,24	5,99E-03
GKD-b_13_n50_m5	11,1055	78,73	5,82E-03
GKD-b_14_n50_m5	9,7243	82,90	6,55E-03
GKD-b_15_n50_m5	9,8645	71,08	5,35E-03
GKD-b_16_n50_m15	71,4936	40,21	4,03E-02
GKD-b_17_n50_m15	84,6775	43,19	3,40E-02
GKD-b_18_n50_m15	59,0739	26,88	2,81E-02
GKD-b_19_n50_m15	73,9217	37,21	3,04E-02
GKD-b_20_n50_m15	47,7151	0,00	3,17E-02
GKD-b_21_n100_m10	17,6392	21,58	3,89E-02
GKD-b_22_n100_m10	29,4475	53,60	3,41E-02
GKD-b_23_n100_m10	27,6731	44,55	3,32E-02
GKD-b_24_n100_m10	28,0150	69,16	3,43E-02
GKD-b_25_n100_m10	26,5326	35,17	3,96E-02
GKD-b_26_n100_m30	251,4400	32,89	2,83E-01
GKD-b_27_n100_m30	213,2290	40,39	2,23E-01
GKD-b_28_n100_m30	199,6980	46,73	2,95E-01
GKD-b_29_n100_m30	264,2700	47,99	2,91E-01
GKD-b_30_n100_m30	216,8940	41,22	3,81E-01
GKD-b_31_n125_m12	20,9761	44,01	1,31E-01
GKD-b_32_n125_m12	36,9977	49,22	9,10E-02
GKD-b_33_n125_m12	32,7848	43,48	8,89E-02
GKD-b_34_n125_m12	41,7969	53,37	8,24E-02
GKD-b_35_n125_m12	26,1339	30,69	8,21E-02
GKD-b_36_n125_m37	275,7060	43,62	3,89E-01
GKD-b_37_n125_m37	339,4410	41,41	4,42E-01
GKD-b_38_n125_m37	364,0610	48,37	4,36E-01
GKD-b_39_n125_m37	252,0130	33,10	4,02E-01
GKD-b_40_n125_m37	294,8270	39,56	4,14E-01
GKD-b_41_n150_m15	41,5348	43,79	1,23E-01
GKD-b_42_n150_m15	58,7032	54,36	1,21E-01
GKD-b_43_n150_m15	44,5965	40,01	1,17E-01
GKD-b_44_n150_m15	42,6617	39,21	1,48E-01
GKD-b_45_n150_m15	29,3445	5,36	1,54E-01
GKD-b_46_n150_m45	406,9080	44,03	7,13E-01
GKD-b_47_n150_m45	316,0860	27,68	6,80E-01
GKD-b_48_n150_m45	367,0460	38,22	6,22E-01
GKD-b_49_n150_m45	430,4760	47,40	6,24E-01
GKD-b_50_n150_m45	384,6700	35,31	7,73E-01

?figurename? 2: ES y BMB

Algoritmo ILS			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	6,35E-04
GKD-b_2_n25_m2	0,0000	0,00	8,17E-04
GKD-b_3_n25_m2	0,0000	0,00	8,03E-04
GKD-b_4_n25_m2	0,0000	0,00	6,17E-04
GKD-b_5_n25_m2	0,0000	0,00	6,15E-04
GKD-b_6_n25_m7	26,1224	51,31	6,75E-03
GKD-b_7_n25_m7	26,2168	46,22	5,48E-03
GKD-b_8_n25_m7	23,3432	28,20	6,68E-03
GKD-b_9_n25_m7	24,5974	30,61	6,35E-03
GKD-b_10_n25_m7	30,2121	22,99	5,65E-03
GKD-b_11_n50_m5	3,5615	45,92	7,76E-03
GKD-b_12_n50_m5	7,4370	71,48	6,28E-03
GKD-b_13_n50_m5	10,9761	78,48	5,86E-03
GKD-b_14_n50_m5	10,7547	84,54	6,22E-03
GKD-b_15_n50_m5	9,8505	71,04	6,03E-03
GKD-b_16_n50_m15	74,7359	42,80	3,11E-02
GKD-b_17_n50_m15	81,1622	40,73	3,61E-02
GKD-b_18_n50_m15	59,0739	26,88	2,58E-02
GKD-b_19_n50_m15	65,4382	29,07	3,05E-02
GKD-b_20_n50_m15	63,5880	24,96	2,96E-02
GKD-b_21_n100_m10	22,6137	38,83	3,31E-02
GKD-b_22_n100_m10	20,9110	34,65	3,34E-02
GKD-b_23_n100_m10	28,0298	45,25	3,97E-02
GKD-b_24_n100_m10	26,1000	66,89	3,46E-02
GKD-b_25_n100_m10	28,5724	39,80	3,87E-02
GKD-b_26_n100_m30	272,2970	38,03	2,17E-01
GKD-b_27_n100_m30	202,9980	37,39	2,14E-01
GKD-b_28_n100_m30	222,5540	52,20	2,12E-01
GKD-b_29_n100_m30	229,3250	40,06	2,96E-01
GKD-b_30_n100_m30	190,9030	33,22	2,37E-01
GKD-b_31_n125_m12	20,9761	44,01	9,88E-02
GKD-b_32_n125_m12	35,3080	46,79	5,42E-02
GKD-b_33_n125_m12	37,9415	51,16	5,62E-02
GKD-b_34_n125_m12	31,4504	38,03	7,92E-02
GKD-b_35_n125_m12	31,2772	42,09	6,39E-02
GKD-b_36_n125_m37	240,3260	35,32	4,98E-01
GKD-b_37_n125_m37	374,8520	46,94	4,90E-01
GKD-b_38_n125_m37	286,5720	34,41	4,53E-01
GKD-b_39_n125_m37	259,8080	35,11	3,55E-01
GKD-b_40_n125_m37	353,2010	49,55	3,76E-01
GKD-b_41_n150_m15	46,4561	49,75	1,77E-01
GKD-b_42_n150_m15	46,7188	42,66	1,31E-01
GKD-b_43_n150_m15	40,4785	33,90	1,28E-01
GKD-b_44_n150_m15	45,0720	42,46	1,09E-01
GKD-b_45_n150_m15	44,5920	37,72	1,25E-01
GKD-b_46_n150_m45	373,6440	39,05	6,88E-01
GKD-b_47_n150_m45	382,3180	40,21	6,03E-01
GKD-b_48_n150_m45	346,0850	34,48	6,57E-01
GKD-b_49_n150_m45	480,4960	52,88	5,96E-01
GKD-b_50_n150_m45	412,6600	39,69	5,36E-01

Algoritmo ILS-ES			
Caso	Coste medio obtenido	Desv	Tiempo
GKD-b_1_n25_m2	0,0000	0,00	1,29E-02
GKD-b_2_n25_m2	0,0000	0,00	3,83E-02
GKD-b_3_n25_m2	0,0000	0,00	7,22E-03
GKD-b_4_n25_m2	0,0000	0,00	6,69E-02
GKD-b_5_n25_m2	0,0000	0,00	6,38E-03
GKD-b_6_n25_m7	36,7216	65,37	1,34E-02
GKD-b_7_n25_m7	35,4433	60,22	1,76E-02
GKD-b_8_n25_m7	28,5340	41,26	2,21E-02
GKD-b_9_n25_m7	14,2133	-16,73	1,96E-02
GKD-b_10_n25_m7	29,2815	20,55	2,04E-02
GKD-b_11_n50_m5	4,9918	61,41	2,67E-02
GKD-b_12_n50_m5	4,7832	55,66	2,67E-02
GKD-b_13_n50_m5	9,1888	74,29	3,17E-02
GKD-b_14_n50_m5	7,6573	78,28	3,76E-02
GKD-b_15_n50_m5	10,7900	73,56	3,29E-02
GKD-b_16_n50_m15	72,8178	41,30	1,00E-01
GKD-b_17_n50_m15	52,9493	9,14	1,94E-01
GKD-b_18_n50_m15	73,4716	41,21	7,68E-02
GKD-b_19_n50_m15	92,2536	49,69	8,34E-02
GKD-b_20_n50_m15	53,5905	10,96	7,51E-02
GKD-b_21_n100_m10	30,3331	54,40	8,30E-02
GKD-b_22_n100_m10	31,6549	56,83	7,46E-02
GKD-b_23_n100_m10	23,7813	35,47	7,69E-02
GKD-b_24_n100_m10	23,9850	63,97	7,88E-02
GKD-b_25_n100_m10	27,5686	37,61	9,61E-02
GKD-b_26_n100_m30	214,2140	21,23	4,82E-01
GKD-b_27_n100_m30	249,9130	49,14	4,13E-01
GKD-b_28_n100_m30	268,6980	60,41	4,69E-01
GKD-b_29_n100_m30	210,0580	34,56	3,86E-01
GKD-b_30_n100_m30	260,0890	50,99	4,58E-01
GKD-b_31_n125_m12	23,5328	50,09	1,97E-01
GKD-b_32_n125_m12	35,9540	47,74	1,11E-01
GKD-b_33_n125_m12	28,6515	35,32	1,26E-01
GKD-b_34_n125_m12	38,9089	49,91	1,27E-01
GKD-b_35_n125_m12	25,5979	29,24	1,38E-01
GKD-b_36_n125_m37	312,2190	50,22	8,12E-01
GKD-b_37_n125_m37	315,6480	36,99	9,04E-01
GKD-b_38_n125_m37	368,0200	48,92	7,12E-01
GKD-b_39_n125_m37	317,1240	46,84	7,99E-01
GKD-b_40_n125_m37	309,1560	42,36	7,92E-01
GKD-b_41_n150_m15	53,6233	56,46	2,15E-01
GKD-b_42_n150_m15	44,6661	40,02	2,13E-01
GKD-b_43_n150_m15	53,7079	50,19	1,96E-01
GKD-b_44_n150_m15	55,2752	53,08	1,94E-01
GKD-b_45_n150_m15	33,4053	16,86	2,07E-01
GKD-b_46_n150_m45	405,7610	43,87	1,14E+00
GKD-b_47_n150_m45	374,4660	38,95	1,29E+00
GKD-b_48_n150_m45	427,6640	46,98	1,04E+00
GKD-b_49_n150_m45	454,0600	50,14	1,21E+00
GKD-b_50_n150_m45	481,7720	48,35	1,18E+00

?figurename? 3: ILS y ILS-ES

Algoritmo	Desv	Tiempo
Gready	66,53792625876	0,002293429016
BL	56,71072367558	0,019550993792
ES	40,33419921395	0,075851557
BMB	36,83574280366	0,17037027956
ILS	39,15541283116	0,15700306784
ILS-ES	40,26644538952	0,3026681576

?figurename? 4: Tabla de resultados