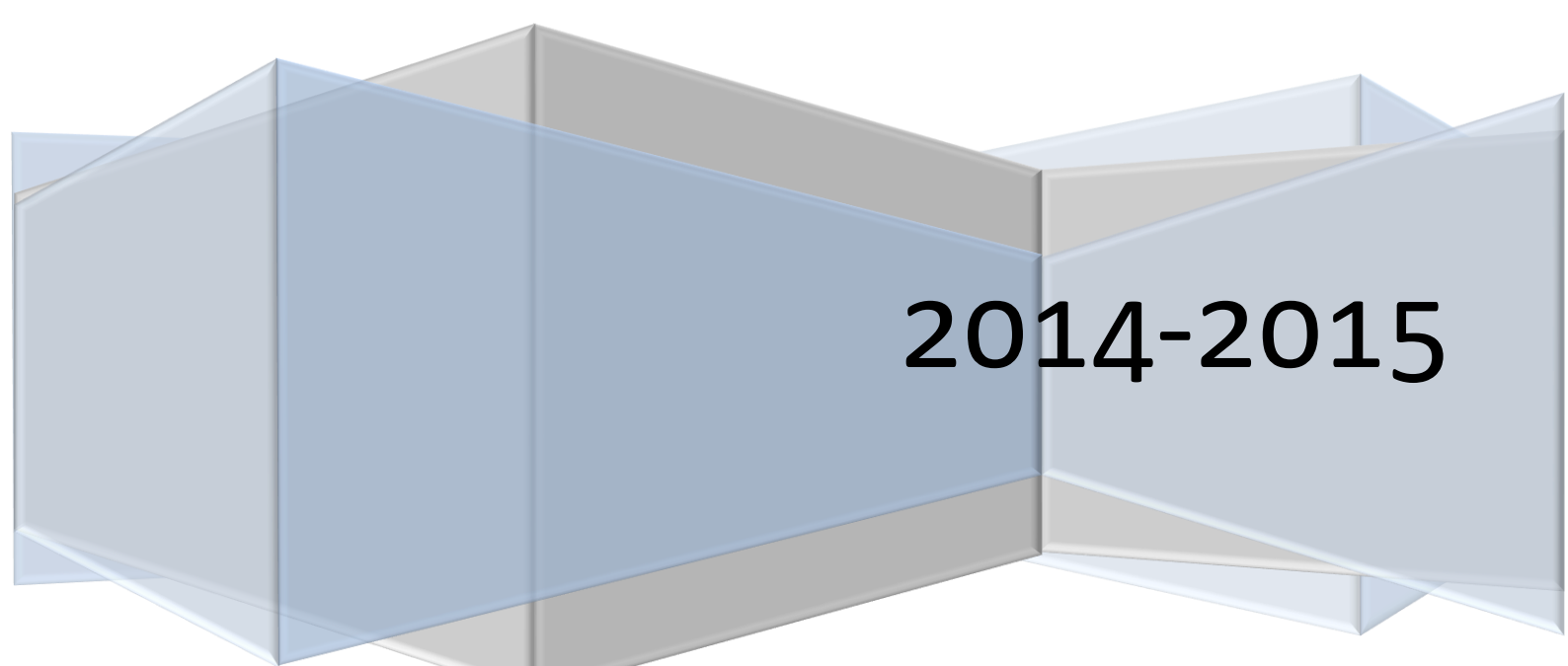


**Universidad de Córdoba – Escuela Politécnica Superior**  
**Grado en Ingeniería Informática**

# **Práctica 2 -**

# **Metaheurísticas**

**Metaheurísticas basadas en una solución**



**2014-2015**

# Índice

---

1. Descripción de los problemas.....	1
1.1. Problemas	
1.1.1. MMDP	
1.1.2. TSP	
1.2. Algoritmos	
1.2.1. Búsqueda Local	
1.2.2. Enfriamiento Simulado	
2. Descripción del esquema de representación y pseudocódigos.....	3
2.1. MMDP	
2.2. TSP	
2.3. Generación de vecino	
2.3.1. MMDP	
2.3.2. TS	
3. Descripción de la estructura del método de búsqueda.....	5
3.1. Búsqueda Local <i>bestImprovement</i>	
3.1.1. MMDP	
3.1.2. TSP	
3.2. Búsqueda Local <i>firstImprovement</i>	
3.2.1. MMDP	
3.2.2. TSP	
3.3. Enfriamiento Simulado	
3.3.1. MMDP	
3.3.2. TSP	
3.4. Generación de lista de candidatos CV	
3.5. Método de exploración en Búsqueda Local	
4. Experimentos y análisis de resultados.....	7
4.1. Descripción de las Instancias	
4.1.1. MMDP	
4.1.2. TSP	
4.2. Resultados MMDP	
4.3. Resultados TSP	

# 1. Descripción de los problemas

## 1.1. Problemas

### 1.1.1. MMDP

Denominado MaxMin Diversity Problem, o en castellano máxima mínima diversidad, consiste en seleccionar un determinado número de elementos de un conjunto de  $n$  elementos de forma que la menor de las distancias entre todos los elementos sea máxima. Es decir, se coge la menor distancia que haya entre los elementos del problema de una de las soluciones escogidas aleatoriamente, y de entre todas ellas se elige la más grande.

### 1.1.2. TSP

Travelling Salesman Problem o el problema del viajante de comercio, consiste en encontrar el camino más corto que une un conjunto de  $m$  ciudades, de forma que ninguna ciudad de visita dos veces y que el recorrido empiece y termina en la misma ciudad. Esto se consigue a través de permutaciones, de forma que si ordenamos de forma aleatoria el vector solución sólo habría que sumar la distancia de uno al siguiente y finalmente del último al primero.

## 1.2. Algoritmos

### 1.2.1. Búsqueda local

Es un proceso iterativo de búsqueda en el que, dada una solución actual, se selecciona una solución de su entorno para continuar la búsqueda. Esta solución se selecciona si mejora a la actual, si no se desecha. En este caso se generará una solución aleatoria y a partir de ésta se irán generando vecinos y cambiando la solución si ésta mejora.

Existen dos variantes de este algoritmo:

- FirstImprovement: En este caso el algoritmo para de buscar en el momento en el que encuentra una solución que mejore a la que ya se tiene.
- BestImprovement: El algoritmo explorará todas las posibles soluciones vecinas y seleccionará la mejor de entre todas ellas.

### 1.2.2. Enfriamiento Simulado

Es un algoritmo de búsqueda por entornos con un criterio probabilístico de aceptación de soluciones basado en termodinámica. Permite que algunos movimientos sean hacia soluciones peores, para evitar que se quede en óptimos locales. Hay que controlar estos movimientos, ya que nos pueden desviar del óptimo global. Para ello siempre se guarda la mejor solución hasta el momento. Conforme va avanzando la búsqueda va disminuyendo la probabilidad de que salte a una solución peor que la actual.

## 2. Descripción del esquema de representación y pseudocódigos

### 2.1. MMDP

Para este problema se han utilizado dos estructuras, una para almacenar la instancia a utilizar y otra para almacenar la solución.

- La estructura de la instancia se compone por un entero que almacenará el tamaño de la matriz de distancias, el número de elementos que se van a seleccionar y de la matriz de distancias.
- La estructura de la solución se compone de un vector booleano que indica qué elementos se han seleccionado.

#### Función objetivo

```
Para (desde inicio (i) hasta N)
  Si (solucion[i]==1)
    Para (desde i+1(j) hasta N)
      Si (solucion[j]==1)
        Si (distancias[i][j] < minimo)
          minimo=distancias[i][j]
        finSi
      finPara
    finSi
  finPara
devuelve minimo;
```

### 2.2. TSP

Para este problema se han utilizado dos estructuras, una para almacenar la instancia a utilizar y otra para almacenar la solución.

- La estructura de la instancia se compone por un entero que almacenará el tamaño de la matriz de distancias y de la matriz de distancias.
- La estructura de la solución se compone de un vector donde se almacena el orden de los nodos para la suma de las distancias entre ellos.

#### Función objetivo

```
suma=0.0

Para (desde inicio hasta N)
  suma+=distancias[solucion[i]][solucion[(i+1)modulo(N)]]
finPara

devuelve suma;
```

### 2.3. Generación de vecino

Para este algoritmo se han implementado dos formas diferentes, una por cada tipo de problema. En ambas la única estructura que se mantiene es la de la solución, que en cada caso es la descrita para cada problema en el punto anterior.

#### 2.3.1. MMPD

En este caso se utiliza un contador, que servirá para saber qué elemento a 1 hay que intercambiar (el primero, el segundo, el tercero, etc.). Se usa también un elemento “p” que nos indica a partir de qué elemento del vector de solución se tiene que ir mirando para intercambiarlo por el elemento que está a 1.

```
contador=0
Para (desde inicio (i) hasta N)
  Si (solucion[i]==1)
    contador++
    Si (contador==cont)
      Para (desde p (j) hasta N)
        Si (solsucion[j]==0)
          solucion[i]=0
          solucion[j]=1
          devuelve solucion
        finSi
      finPara
    finSi
  finSi
finPara
```

#### 2.3.2. TSP

Ahora hay que controlar si el algoritmo a utilizar es BLf o ES, puesto que el tamaño del vector de candidatos es la mitad que el que se utiliza en BLb. Se utiliza una variable “aux” para efectuar el cambio, una variable “candidato” para saber qué elemento de la lista se va a intercambiar y una variable “cambio” que señala el valor que lo va a sustituir.

```
totalBeta=data.n*0.5;

/*p=0 BLf; p=1 BLb; p=2 ES*/

Si (p==0 or p==2)
  candidato=cont modulo(totalBeta)
  aux=solucion[(cv[candidato]+1)modulo(totalBeta)]
  solucion[(cv[candidato]+1)%totalBeta]=solucion[cambio]
  solucion[cambio]=aux

Sino si (p==1)
  candidato=cont%(cv.size())
  aux=solucion[(cv[candidato]+1)modulo(N)]
  solucion[(cv[candidato]+1)modulo(N)]=solucion[cambio]
  solucion[cambio]=aux
finSi

devuelve solucion;
```

### 3. Descripción de la estructura de los métodos de búsqueda

#### 3.1. Búsqueda Local *firstImprovement*

##### 3.1.1. MMDP

```
Para (desde inicio hasta (N*50))
  Si ((i)modulo(N) ==0)
    cont++ //Elige el 1 a cambiar
    sol2=generarVecinos(sol, (i)modulo(N), (cont)modulo(M));
    Si (aceptacion(solución, sol2))
      devuelve solucion2
finPara
```

##### 3.1.2. TSP

```
cv=listaCandidatos(sol, p=1)
Para (desde inicio (i) hasta 1000*M)
  Para (desde inicio (j) hasta N)
    Si (j!=(anterior al candidato) && j!=(siguiente al candidato) && j!=candidato)
      introduce (j) en cambios
  finPara
  numerosPorElegir=cambios
  Para (desde inicio (t) hasta N-3)
    cambio=elegirAleatorioSinRepeticion(numerosPorElegir, N-3)
    sol2 = generarVecinos1(sol, p, cv[(i)modulo(totalBeta)], cv, cambio);
    Si (aceptacion(sol, sol2))
      sol=sol2
  finPara
finPara
devuelve sol;
```

#### 3.2. Búsqueda Local *bestImprovement*

##### 3.2.1. MMDP

```
Para (desde inicio hasta (N*50))
  Si ((i)modulo(N) ==0)
    cont++ //Elige el 1 a cambiar
    sol2=generarVecinos(sol, (i)modulo(N), (cont)modulo(M));
    Si (aceptacion(solución, sol2))
      sol=solucion2
finPara
devuelve sol
```

##### 3.2.2. TSP

```
cv=listaCandidatos(sol, p=0)
Para (desde inicio (i) hasta 1000*M)
  Para (desde inicio (j) hasta N)
    Si (j!=(anterior al candidato) && j!=(siguiente al candidato) && j!=candidato)
      introduce (j) en cambios
  finPara
  Para (desde inicio (j) hasta N-3)
    sol2 = generarVecinos1(sol, p, cv[(i)modulo(N*0,5)], cv, cambios[j]);
    if (aceptacion(sol, sol2))
      devuelve sol2
  finPara
finPara
```

### 3.3. Enfriamiento Simulado

#### 3.3.1. MMDP

```
nMejoras=data.n*(0.1), valorActual=evaluarSolucion(sol), tActual=10*valorActual/(-
log(0.3)), mejorSol=sol, iterRelaTamano=50*N
Hacer
    Hacer
        Si ((i)modulo(N)==0)
            cont++
            sol2=generarVecinos(solActual, (i)modulo(N), (cont)modulo(M))
            valorActual=evaluarSolucion(solActual)
            soluciones++
            incremento=valorActual-evaluarSolucion(sol2)
            Si (incremento<0 or (U(0,1) <= exp(-
incremento*evaluarSolucion(sol)/tActual))
                solActual=sol2
                Si (evaluarSolucion(sol2)>evaluarSolucion(mejorSol)){
                    mejorSol=sol2
                    mejoras++
                }
            finSi
            Si ((i)modulo(N)==0 && cont==M)
                acabado=1
            i++;
        mientras (i<N && mejoras<nMejoras && acabado==0 &&
soluciones<iterRelaTamano)
            tActual=tActual*gamma
        mientras (tActual<tFinal && soluciones<iterRelaTamano);
    devuelve mejorSol;
```

#### 3.3.2. TSP

```
sol2, mejorSol=sol, solActual=sol, cv=listaCandidatos(sol, p),
valorActual=evaluarSolucion(sol);, tActual=0.001*valorActual/(-log(0.3));

Hacer
    Hacer
        cv=listaCandidatos(solActual, p)
        i=rand()modulo(nCandidatos)
        Para (desde inicio (j) hasta N)
            Si (j!=(anterior al candidato) && j!=(siguiente al candidato) &&
j!=candidato)
                introduce (j) en cambios
        finPara
        c=rand()modulo(N-3)
        sol2=generarVecinos1(solActual, p, cv[i], cv, cambios[c])
        soluciones++
        incremento=valorActual-evaluarSolucion(sol2)
        Si (incremento<0 || U(0,1) <= exp(-
incremento*evaluarSolucion(sol)/tActual))
            solActual=sol2
            Si (evaluarSolucion(sol2)>evaluarSolucion(mejorSol)){
                mejorSol=sol2
                mejoras++
            }
        finSi
        mientras (soluciones<N*5 && mejoras<nMejoras &&
soluciones<iterRelaTamano)
            tActual=tActual*gamma
        mientras (tActual<tFinal && soluciones<iterRelaTamano && mejorSol != solActual)
    devuelve mejorSol
```

### 3.4. Generación de lista de candidatos CV

```
d, d2, cv //vectores
totalBeta=0.5*data.n

Para (desde inicio hasta N)
    Introducimos distancias en d
finPara
d2=d
ordenamos d en orden descendente
Si (p==0 || p==2) //p=0 BLf; p=1 BLb; p=2 ES
    Para (desde inicio (i) hasta totalBeta)
        Para (desde inicio (j) hasta N)
            Si(d[i]==d2[j])
                Introducimos (j) en cv (Introducimos las aristas en el orden de
                                     la solución inicial, no ordenadas por
                                     tamaño, así equivalen al nodo que hay
                                     que cambiar en la solución)
        finPara
    finPara
sino
    Igual que para BLf y ES pero el primer Para es hasta N

devuelve cv;
```

### 3.5. Método de exploración en Búsqueda Local

El método de exploración está explícito en el algoritmo de búsqueda local, ya que se recorren todas las opciones de la forma descrita en el guión a la vez que se van generando con su posterior evaluación.

## 4. Experimentos y análisis de resultados

### 4.1. Descripción de las instancias

#### 4.1.1. MMDP

El nombre de las instancias está compuesto por un código alfanumérico, en el que se encuentra un “nXX” y un “mXX”, donde las XX indican el número de nodos del grafo y el número de elementos a elegir, respectivamente.

En la instancia encontramos en la primera fila el número de nodos del grafo seguido del número de nodos a elegir para calcular la distancia. Seguidamente encontramos por filas el número de un nodo, el número de un nodo diferente y la distancia que hay entre estos dos nodos.

#### 4.1.2. TSP

Estas instancias contienen en la primera fila el número de ciudades comprendidas en el grafo seguidas de las distancias entre las ciudades. En cada fila están las distancias de la ciudad que corresponde a esa posición (si es la primera fila, la primera ciudad) con las demás ciudades, separadas por un espacio. Los nombres de las instancias están compuestos por unas letras y un número, que indica el número de nodos del grafo, excepto la instancia “p01.txt”, que tiene 15 ciudades.



## 4.2. Resultados obtenidos

	Algoritmo BL b		Algoritmo BL f		Algoritmo ES	
Fichero	Desv	Tiempo	Desv	Tiempo	Desv	Tiempo
GKD-la_11_n10_m4.txt	-1,75	0,010	-12,53	0,006	-20,78	0,005
GKD-la_12_n10_m4.txt	2,90	0,008	-14,29	0,007	-14,29	0,005
GKD-la_1_n10_m2.txt	-6,63	0,009	-20,63	0,009	-20,63	0,007
GKD-la_21_n10_m8.txt	-1,94	0,010	-3,52	0,007	-16,01	0,005
GKD-la_22_n10_m8.txt	0,33	0,009	-9,47	0,006	0,00	0,004
GKD-la_2_n10_m2.txt	-6,08	0,009	-19,68	0,008	-16,64	0,008
GKD-la_31_n15_m4.txt	-19,30	0,012	-44,39	0,004	-46,56	0,005
GKD-la_32_n15_m4.txt	2,72	0,010	-13,08	0,005	-19,37	0,005
GKD-la_41_n15_m9.txt	9,11	0,014	-1,60	0,005	-1,60	0,005
GKD-la_42_n15_m9.txt	7,70	0,014	-9,39	0,005	-14,13	0,011
GKD-la_46_n15_m12.txt	0,28	0,013	-9,78	0,005	-9,78	0,005
GKD-la_47_n15_m12.txt	0,00	0,013	-3,13	0,006	-3,13	0,005
GKD-la_56_n30_m9.txt	-0,66	0,034	-20,47	0,006	-20,47	0,007
GKD-la_57_n30_m9.txt	-9,14	0,035	-23,23	0,006	-14,09	0,005
GKD-la_61_n30_m12.txt	-5,01	0,035	-25,53	0,005	-24,57	0,005
GKD-la_62_n30_m12.txt	-12,42	0,044	-38,52	0,008	-22,53	0,005
GKD-la_66_n30_m18.txt	-2,87	0,048	-20,33	0,012	-20,69	0,005
GKD-la_67_n30_m18.txt	-3,65	0,048	-23,71	0,010	-23,71	0,005
GKD-la_71_n30_m24.txt	-7,64	0,051	-10,21	0,009	-15,34	0,006
GKD-la_72_n30_m24.txt	0,00	0,058	-8,51	0,016	-19,68	0,006
GKD-lc_10_n500_m50.txt	-40,64	30,706	-49,15	0,058	-36,18	0,625
GKD-lc_11_n500_m50.txt	-41,69	31,393	-41,69	14,635	-44,32	0,606
GKD-lc_12_n500_m50.txt	-48,89	30,248	-48,89	6,684	-38,28	0,545
GKD-lc_13_n500_m50.txt	-46,04	29,063	-46,04	29,975	-35,28	0,593
GKD-lc_14_n500_m50.txt	-52,33	31,472	-52,33	31,319	-40,79	0,593
GKD-lc_15_n500_m50.txt	-51,69	29,461	-51,69	6,166	-39,61	0,585
GKD-lc_16_n500_m50.txt	-44,97	30,664	-44,97	11,443	-45,46	0,696
GKD-lc_17_n500_m50.txt	-51,95	29,747	-51,95	30,108	-38,80	0,553
GKD-lc_18_n500_m50.txt	-60,61	30,531	-60,61	30,240	-42,68	0,626
GKD-lc_19_n500_m50.txt	-51,76	30,310	-51,76	12,860	-42,96	0,600
GKD-lc_1_n500_m50.txt	-56,39	30,263	-56,39	30,340	-42,32	0,604
GKD-lc_20_n500_m50.txt	-42,81	29,829	-42,81	22,616	-41,55	0,518
GKD-lc_2_n500_m50.txt	-56,31	31,970	-56,31	31,275	-40,57	0,545
GKD-lc_3_n500_m50.txt	-45,58	29,828	-45,58	19,017	-41,08	0,503
GKD-lc_4_n500_m50.txt	-58,33	30,104	-58,33	29,484	-41,65	0,556
GKD-lc_5_n500_m50.txt	-48,31	29,826	-48,31	12,500	-38,15	0,579
GKD-lc_6_n500_m50.txt	-48,02	31,080	-48,02	31,376	-41,07	0,551
GKD-lc_7_n500_m50.txt	-51,13	31,713	-51,13	31,223	-37,69	0,593
GKD-lc_8_n500_m50.txt	-41,88	29,836	-41,88	16,687	-36,09	0,591
GKD-lc_9_n500_m50.txt	-49,71	30,217	-49,71	30,445	-40,78	0,576
Media	-26,08	15,22	-33,24	10,71	-28,73	0,29
Desviación Típica	24,44	15,40	18,92	13,07	13,71	0,29
Máximo	9,11	31,97	-1,60	31,38	0,00	0,70
Mínimo	-60,61	0,01	-60,61	0,00	-46,56	0,00

Tabla 1: Resultados MMDP

	Algoritmo BL b		Algoritmo BL f		Algoritmo ES	
Fichero	Desv	Tiempo	Desv	Tiempo	Desv	Tiempo
att48.txt	568,89	0,028	1354,96	0,019	1470,98	0,027
berlin52.txt	95,09	0,031	289,45	0,021	320,15	0,020
ch130.txt	189,47	0,143	629,15	0,034	691,16	0,055
ch150.txt	199,53	0,196	712,12	0,040	760,35	0,167
d198.txt	373,14	0,458	1126,72	0,055	1187,93	0,392
dantzig42.txt	108,26	0,026	290,17	0,020	321,38	0,020
eil101.txt	138,77	0,084	441,02	0,027	480,71	0,053
eil51.txt	101,17	0,031	288,50	0,020	320,64	0,025
eil76.txt	124,48	0,050	359,41	0,022	389,06	0,039
fri26.txt	71,72	0,022	178,73	0,020	210,14	0,021
gr17.txt	38,80	0,022	122,85	0,020	151,02	0,021
gr202.txt	166,31	0,513	473,23	0,057	501,67	0,424
gr96.txt	201,85	0,075	565,18	0,026	604,03	0,057
kroA100.txt	188,53	0,082	635,85	0,027	723,23	0,037
kroA200.txt	284,05	0,397	1066,36	0,658	1140,86	0,432
p01.txt	42,04	0,021	123,37	0,020	137,57	0,020
pr76.txt	158,11	0,051	420,69	0,023	463,93	0,038
st70.txt	139,18	0,041	415,80	0,022	459,56	0,026

Media	177,19	0,13	527,42	0,06	574,13	0,10
Desviación Típica	127,56	0,16	349,21	0,15	371,17	0,15
Máximo	568,89	0,51	1354,96	0,66	1470,98	0,43
Mínimo	38,80	0,02	122,85	0,02	137,57	0,02

Tabla 2: Resultados TSP

### 4.3. Análisis de los resultados

#### 4.3.1. MMDP

En la Tabla 1 se encuentran los resultados referentes a la aplicación de los algoritmos al problema MMDP. Estos resultados son la media de la aplicación de los algoritmos con tres semillas diferentes (10, 20 y 30).

Como se puede observar, con las instancias de tamaño no superior a 30 nodos, los tiempos de ejecución son ínfimos, mientras que con un tamaño de 500 nodos son del orden de medio minuto o un minuto en los algoritmos de Búsqueda Local. Esto se debe principalmente al tamaño del problema, básicamente porque al tener que tratar con una solución de mayor tamaño, el algoritmo tiene muchos más posibles vecinos que recorrer y comparar, aunque también depende de la máquina y del código. Podría ser una señal clara de que hace falta depurarlo para llegar a una ejecución más rápida.

En BLb se puede observar una tendencia clara, en la que los grafos con tamaño 30 o menor tienen desviaciones muy pequeñas, exceptuando la séptima instancia. Esto se debe a que en estas instancias el algoritmo puede recorrer todas o casi todas las soluciones posibles,

por lo que se quedaría muy próximo al óptimo global, mientras que en las de mayor tamaño sólo recorre hasta un máximo de soluciones vecinas a la original y no sigue avanzando más.

En BLf también encontramos dicha tendencia, aunque menos pronunciada, ya que no recorre todas las posibles soluciones vecinas, si no que va las va recorriendo y se queda con la primera que mejora la solución inicial. La razón de dicha tendencia en este caso es la misma que en el anterior, al tener un número de posibles soluciones al problema mucho mayor, es más complicado quedarse cerca del óptimo global.

En ES ocurre algo muy parecido al BLb, y esto se debe a que tiene muchas más condiciones de parada que el algoritmo BL. Bien porque la temperatura desciende demasiado o bien porque se alcanza el máximo de soluciones preestablecido, el algoritmo parará de buscar. La primera razón puede darse y ser el número de soluciones vistas mucho menor el BLb.

Si nos fijamos en las medias se puede observar que los mejores resultados los obtiene BLb, sobretodo si eliminamos los resultados de las instancias más grandes, en las que ninguno de los algoritmos ha conseguido resultados que sobresalgan sobre el resto. BLb obtiene la mejor media, aunque sus datos son más dispares que los de los otros algoritmos. El tener los mejores resultados tiene un coste temporal más alto, como se puede apreciar en la media de tiempo empleado. Respecto a ES, vemos que en general obtiene una media de desviación aceptable, con unos datos no demasiado dispares y un tiempo de ejecución ínfimo. Esto podría deberse bien a que habría que mejorar la implementación de los otros dos algoritmos o bien a que este algoritmo encuentra óptimos bastante aceptables en muchas menos repeticiones (gracias a la condición de un número máximo de mejoras).

#### **4.3.2. TSP**

En la Tabla 2 se encuentran los resultados referentes a la aplicación de los algoritmos al problema TSP. Estos resultados son la media de la aplicación de los algoritmos con tres semillas diferentes (10, 20 y 30).

Al contrario que en los resultados con MMDP, los tiempos de ejecución en todas las instancias son similares. Esto se debe a que el problema a tratar es diferente, ya que en uno queremos la mínima distancia que conecte a todos los nodos y en el otro caso que la distancia mínima entre dos nodos de un conjunto del total sea máxima. En uno estamos tratando con todo el problema siempre y en el otro sólo con una parte que luego se va modificando.

En este caso el peor de los resultados lo tiene el algoritmo ES, aunque en media similar al BLf, y esto podría deberse a que la temperatura desciende demasiado rápido y no le da tiempo a comprobar suficientes soluciones posibles como haría el BLb y se quedaría en un óptimo local. Con BLf ocurre algo similar respecto a que no evalúa suficientes soluciones, ya que en el momento que una de las soluciones vecinas mejora a la inicial, el algoritmo se detiene y deja de mirar. Esta es la razón de sus malos resultados comparados a BLb, el cuál consigue resultados mucho mejores ya que comprueba todos los vecinos de la mejor solución que tiene en el momento, por lo que éste es la mejor opción para aplicar al TSP.