Metaheurísticas(2019-2020)
Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada

Propuesta de metaheurística personalizada para el Problema de la Máxima Diversidad (MDP)

Alberto Jesús Durán López
DNI: 54142189-M
albduranlopez@gmail.com

Grupo Jueves, 17:30 - 19:30

20 de junio de 2020

Índice

1. Introducción

Durante el transcurso de la asignatura se ha trabajado con el problema de la máxima diversidad *(Max Diversity Problem)*.
En particular, en esta práctica estudiaremos técnicas de búsqueda basadas en poblaciones para la resolución del problema en cuestión.
Comentaremos todos los pasos y problemas encontrados, detallando minuciosamente todos los detalles y soluciones a los mismos.

Además, se incorporarán tablas para mostrar los resultados de todas las ejecuciones y gráficas para contrastar los modelos (optimización, costes y desviación).

2. Problema de la máxima Diversidad

El problema de la máxima diversidad (*maximum diversity problem*, MDP) es un problema de optimización combinatoria consistente en seleccionar un subconjunto de m elementos ($|M| = m$) de un conjunto inicial N de n elementos (con $n > m$) de forma que se maximice la diversidad entre los elementos escogidos.

El **MDP** se puede formular como:

$$\text{Maximizar: } z_{MS}(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j$$

$$\text{Sujeto a: } \sum_{i=1}^{n} x_i = m \quad \text{con } x_i = \{0,1\},\ i = 1, ..., n \quad \text{donde:}$$

- x es una solución al problema que consiste en un vector binario que indica los m elementos seleccionados.
- d_{ij} es la distancia existente entre los elementos i y j

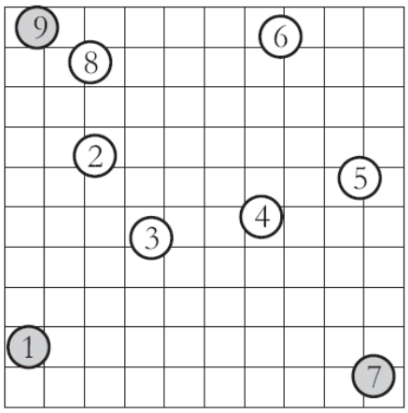


Figura 2.1: MDP-Maximum Diversity Problem

3. Datos y casos considerados

Las ejecuciones se han realizado en un ordenador `Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz, 16GB RAM, 512 SSD`.
Se utilizarán **30 casos** seleccionados de varios de los conjuntos de instancias disponible en la MDPLIB, 10 pertenecientes al grupo `GKD` con distancias Euclideas, n=500 y m=50 *(GKD-c_11_n500_m50 a GKD- c_20_n500_m50)*, 10 del grupo `MDG` con distancias reales en [0,1000], n=500 y m=50 *(MDG-b_1_n500_m50 a MDG-b_10_n500_m50)*; y otras 10 del grupo `MDG` con distancias enteras en 0,10, n=2000 y m=200 *(MDG-a_31_n2000_m200 a MDG- a_40_n2000_m200)*.

Para la realización de la práctica usaremos el lenguaje de programación `C++` ya que se deben probar muchos ejemplos y la ejecución es más rápida al ser un lenguaje de programación compilado.
Todos los archivos se han compilado con la opción de optimización `-O2` y, además, la semilla usada para todas las ejecuciones ha sido 54142189, pudiéndose ésta indicar en la ejecución del archivo en cuestión.

4. Metaheurística original propuesta

Los juegos clásicos que todos conocemos como el buscaminas o el pong, entre otros, se han llevado a cabo en un tablero finito de dimensiones NXM. Sin embargo, existen otros juegos como el snake cuyo tablero no es un plano finito, es decir, se juega sobre un toro llano. SUper ficie no única ,se puede dotar a estos tableros de identificaciones para modificar el tablero.
Antes de empezar, distinguimos los dos tipos de superficies que existen:

- Una superficie es no orientable si existe al menos una curva cerrada simple contenida que es homeomorfa a una banda de Möbius. Ejemplos: La propia banda de Möbius, plano proyectivo, botella de Klein..etc
- En caso contrario, la superficie será orientable. Ejemplos: Toro llano, suma conexa de k toros,

5. Esquema de representación de soluciones

Para todos los algoritmos menos para la *Búsqueda Local*, se ha usado una codificación basada en números binarios. Sin embargo, dejaremos la explicación para las siguientes secciones, donde se explicará con detalle en cada algoritmo.

6. Descripción de la Función Objetivo

Todos los algoritmos implementados hacen uso de la misma función objetivo, la comentada en el punto (2) anterior y cuyo pseudocódigo mostramos a continuación. Bien es cierto que en el caso de la *Búsqueda Local* se tiene que usar codificación basada en números enteros, sin embargo, dicha función se ve mínimamente modificada.

Datos: ContribucionIndep(*integer ind, vector sel, matriz distancias)*
inicio
  $suma \leftarrow 0$;
  para j *in* sel **hacer**
    $suma \leftarrow sel[j] \cdot (distancias[ind][j] + suma)$;
  fin
  devolver $suma$
fin

Datos: CosteEstimado(*vector sel, matriz distancias)*
inicio
  $suma \leftarrow 0$;
  para i *in* $|sel|$ **hacer**
    si $sel[i] = true$ **entonces**
      $suma \leftarrow ContribucionIndep(i, sel, distancias) + suma$;
    fin
  fin
  devolver $suma/2$
fin

Algoritmo 1: Evaluar solución

7. Explotación Multi-Entorno

```
Datos: MundoSuperficie
inicio
    Superficies ← {Botella de Klein, Plano proyectivo,
                   Toro llano } ;

     para s ∈ Superficies hacer
        s ← SolucionInicial(n,m) ;
    fin
    mientras eval < n_eval hacer
        para s ∈ Superficies hacer
            movimiento ;
            nutrirse ;
            mudar_piel ;
        fin
        si se dan las condiciones para viajar entonces
            para s, s' ∈ Superficies hacer
                s viaja a s';
            fin
        fin
    fin
fin
```

Algoritmo 2: Mundo Superficie

8. Hibridación con Búsqueda Local

El algoritmo de *Búsqueda Local* usado para nuestro algoritmo se corresponde con el implementado en la primera práctica, por ello, conviene recordar como funciona:

- Se ha implementado una función de intercambio, Int(sel,i,j). El parámetro `i` indica el índice del elemento que se eliminará de la solución y el j por cual se sustituirá. Por tanto, las opciones para `i` serán `m` (los m elementos seleccionados) y las opciones para `j` serán `n-m` (los elementos disponibles en *no seleccionados*), por tanto, el entorno estará formado por `m·(n-m)` elementos.

Datos: EvaluaVecinos
inicio
  $mejora \leftarrow True$;
  $eval \leftarrow 0$;

  mientras *eval<100.000 and mejora* **hacer**
    . . .

    $salir \leftarrow$ False, $c \leftarrow 0$;
    mientras $c < n - m$ *and not salir* **hacer**
      Genera elemento válido j ;
      si $Contrib(s_j) > Contrib(s_i)$ **entonces**
        Int(sel, i, j) ;
        $salir \leftarrow$ True;
        $coste \leftarrow coste + contribNueva - contribAntigua$
      fin
      $c \leftarrow c + 1$;
      $eval \leftarrow eval + 1$;
    fin
  fin
fin

Algoritmo 3: EvaluaVecinos

- En cada iteración, podemos calcular la diferencia de costes entre las dos soluciones recalculando todas las distancias de la función objetivo, sin embargo, esto no es necesario ya que como únicamente añadimos y quitamos 1 elemento, basta con sumar y restar la distancia del nuevo y del viejo elemento al resto de elementos seleccionados, respectivamente. Además, combinamos la factorización del coste con el cálculo de la contribución de los elementos para mejorar aún más la eficiencia.

  Si el intercambio es favorable, es decir, si el coste de la nueva solución (sumando las distancias del nuevo elemento y restando las del elemento anterior) es mayor que la anterior, aceptamos el intercambio. En caso contrario, rechazamos.

- Repetiremos este proceso hasta que se realicen N evaluaciones de la función objetivo o cuando no encuentre mejora en el entorno.

9. Búsqueda Tabú

La Búsqueda Tabú es una metaheurística que utiliza un procedimiento de búsqueda heurística local para explorar el espacio de la solución que pertenece a un óptimo local. Uno de los principales características de este método es el uso de memoria adaptativa, que hace que el comportamiento de búsqueda sea más flexible.
Usaremos dos listas:

- `freq`: se almacena el número de veces que el elemento s_i ha sido seleccionado en construcciones previas donde max_freq representa el máximo valor de $freq[i]$ $\forall i$, $i \in 0, ..., n$
- `quality`: En esta lista se almacena la media de las soluciones previas en las que el elemento s_i ha sido seleccionado. Por otro lado, max_q hace referencia al máximo valor de $quality[i]$ $\forall i$, $i \in 0, ..., n$

Then, we modify the evaluation of the attractiveness of each non- selected element in the current construction according to these quantities to favor the selection of elements with low frequency and high quality values.
Mostramos a continuación el pseudocódigo asociado a la función principal de nuestro algoritmo donde S representa el número total de índices, Sel los índices seleccionados y S-Sel los no seleccionados.

Datos: BusquedaTabu
inicio
  $freq[i] = quality[i]$ $\forall s_i \in S$;
  para $eval < n\_eval$ **hacer**
    $sel \leftarrow \emptyset$;
    $s_c = s\_center(S)$;
    mientras $|sel| < m$ **hacer**
      para $s_i \in S - sel$ **hacer**
        $d'(s_i, s_c) \leftarrow d(s_i, s_c) - \beta d(s_i, s_c)\frac{freq[i]}{max\_freq} + \delta d(s_i, s_c)\frac{quality[i]}{max\_q}$;
      fin
      Tomar i^* *tal que* $d'(s_{i^*}, s_c) = max\{d'(s_i, s_c)\}$;
      $freq[i^*] \leftarrow freq[i^*] + 1$;
      $sel \leftarrow sel \cup s_{i^*}$;
      $S \leftarrow S - s_{i^*}$;
      $s_c \leftarrow s\_center(sel)$;
    fin
    $z \leftarrow CosteSolucion(sel)$;
    $quality[i] \leftarrow \frac{quality[i](freq[i]-1)+z}{freq[i]}$ $\forall s_i \in sel$;
  fin
fin

Algoritmo 4: Búsqueda Tabú

10. Resultados Obtenidos

Instancia	Mejor Coste	ES	Desv	Time	BMB	Desv	Time
GKD-c_11_n500_m50	19587,13	19581,2	0,03	0,03	19586	0,01	0,03
GKD-c_12_n500_m50	19360,24	19357	0,02	0,04	19360,2	0,00	0,05
GKD-c_13_n500_m50	19366,70	19366,7	0,00	0,03	19366,7	0,00	0,04
GKD-c_14_n500_m50	19458,56	19453,1	0,03	0,04	19458,6	0,00	0,04
GKD-c_15_n500_m50	19422,15	19418,6	0,02	0,03	19421,6	0,00	0,03
GKD-c_16_n500_m50	19680,21	19665,2	0,08	0,02	19680,2	0,00	0,04
GKD-c_17_n500_m50	19331,39	19327,3	0,02	0,03	19331,4	0,00	0,03
GKD-c_18_n500_m50	19461,39	19459,8	0,01	0,03	19461,4	0,00	0,03
GKD-c_19_n500_m50	19477,39	19471,6	0,03	0,04	19477,3	0,00	0,04
GKD-c_20_n500_m50	19604,84	19601,8	0,02	0,05	19604,8	0,00	0,04
MDG-b_1_n500_m50	778030,62	762432	2,00	0,07	768472	1,23	0,04
MDG-b_2_n500_m50	779963,69	764288	2,01	0,05	771856	1,04	0,04
MDG-b_3_n500_m50	776768,44	763946	1,65	0,05	767181	1,23	0,04
MDG-b_4_n500_m50	775394,62	766680	1,12	0,06	765863	1,23	0,04
MDG-b_5_n500_m50	775611,06	753849	2,81	0,02	770415	0,67	0,04
MDG-b_6_n500_m50	775153,69	759891	1,97	0,05	765810	1,21	0,04
MDG-b_7_n500_m50	777232,87	762001	1,96	0,04	766665	1,36	0,04
MDG-b_8_n500_m50	779168,75	764350	1,90	0,05	773351	0,75	0,04
MDG-b_9_n500_m50	774802,19	756551	2,36	0,03	766528	1,07	0,03
MDG-b_10_n500_m50	774961,31	759213	2,03	0,05	771090	0,50	0,04
MDG-a_31_n2000_m200	114139	112544	1,40	0,41	113028	0,97	0,67
MDG-a_32_n2000_m200	114092	112382	1,50	0,50	112651	1,26	0,65
MDG-a_33_n2000_m200	114124	112426	1,49	0,49	112944	1,03	0,65
MDG-a_34_n2000_m200	114203	112573	1,43	0,43	112835	1,20	0,63
MDG-a_35_n2000_m200	114180	112820	1,19	0,44	112944	1,08	0,64
MDG-a_36_n2000_m200	114252	112386	1,63	0,47	112987	1,11	0,64
MDG-a_37_n2000_m200	114213	112244	1,72	0,43	112874	1,17	0,65
MDG-a_38_n2000_m200	114378	112921	1,27	0,40	112991	1,21	0,65
MDG-a_39_n2000_m200	114201	112209	1,74	0,41	112908	1,13	0,64
MDG-a_40_n2000_m200	114191	112819	1,20	0,40	112834	1,19	0,65

Tabla 10.1: Resultados ES/BMB

Instancia	Mejor Coste	ILS	Desv	Time	ILS-ES	Desv	Time
GKD-c_11_n500_m50	19587,13	19587,1	0,00	0,03	19584,6	0,01	0,31
GKD-c_12_n500_m50	19360,24	19360,2	0,00	0,04	19360,2	0,00	0,32
GKD-c_13_n500_m50	19366,70	19366,7	0,00	0,05	19366,7	0,00	0,32
GKD-c_14_n500_m50	19458,56	19458,6	0,00	0,03	19458,6	0,00	0,29
GKD-c_15_n500_m50	19422,15	19421,6	0,00	0,03	19422,1	0,00	0,33
GKD-c_16_n500_m50	19680,21	19680,2	0,00	0,03	19680,2	0,00	0,32
GKD-c_17_n500_m50	19331,39	19331,4	0,00	0,04	19329,4	0,01	0,29
GKD-c_18_n500_m50	19461,39	19461,4	0,00	0,04	19461,4	0,00	0,35
GKD-c_19_n500_m50	19477,39	19477,3	0,00	0,03	19477,3	0,00	0,32
GKD-c_20_n500_m50	19604,84	19604,8	0,00	0,04	19604,8	0,00	0,34
MDG-b_1_n500_m50	778030,62	768893	1,17	0,04	770455	0,97	0,59
MDG-b_2_n500_m50	779963,69	776145	0,49	0,04	768793	1,43	0,38
MDG-b_3_n500_m50	776768,44	769205	0,97	0,05	768283	1,09	0,65
MDG-b_4_n500_m50	775394,62	769641	0,74	0,04	767491	1,02	0,45
MDG-b_5_n500_m50	775611,06	769000	0,85	0,03	765396	1,32	0,38
MDG-b_6_n500_m50	775153,69	759684	2,00	0,04	768544	0,85	0,47
MDG-b_7_n500_m50	777232,87	765951	1,45	0,06	769574	0,99	0,50
MDG-b_8_n500_m50	779168,75	775018	0,53	0,04	771452	0,99	0,54
MDG-b_9_n500_m50	774802,19	770633	0,54	0,05	769227	0,72	0,50
MDG-b_10_n500_m50	774961,31	766173	1,13	0,05	769320	0,73	0,45
MDG-a_31_n2000_m200	114139	113825	0,28	0,41	113008	0,99	4,21
MDG-a_32_n2000_m200	114092	113268	0,72	0,46	112858	1,08	4,23
MDG-a_33_n2000_m200	114124	113223	0,79	0,43	113062	0,93	4,70
MDG-a_34_n2000_m200	114203	113306	0,79	0,38	112887	1,15	4,55
MDG-a_35_n2000_m200	114180	113838	0,30	0,37	112966	1,06	4,87
MDG-a_36_n2000_m200	114252	113220	0,90	0,39	113002	1,09	4,81
MDG-a_37_n2000_m200	114213	113447	0,67	0,38	112946	1,11	4,39
MDG-a_38_n2000_m200	114378	113427	0,83	0,37	113016	1,19	4,62
MDG-a_39_n2000_m200	114201	113589	0,54	0,38	112761	1,26	4,34
MDG-a_40_n2000_m200	114191	113398	0,69	0,39	112959	1,08	4,61

Tabla 10.2: Resultados ILS

Algoritmo	*Desv*	*Tiempo*
Greedy	$1,36$	$0,46$
BL	$1,07$	0,49
ES	$1,15$	$0,17$
BMB	$0,72$	$0,24$
ILS	$0,55$	$0,16$
ILS-ES	$0,70$	$1,78$

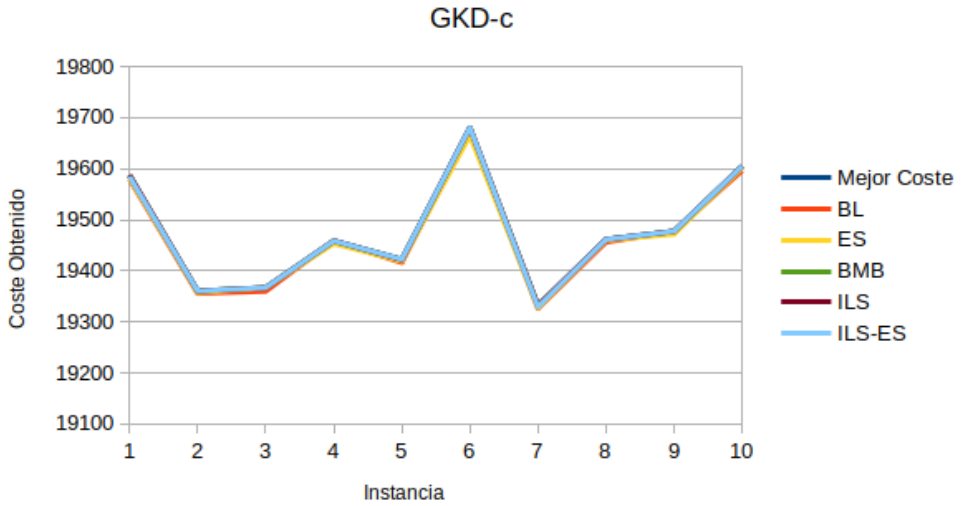
11. Análisis de Resultados

- Observando la tabla de resultados globales 10 anterior, el algoritmo *Greedy* es con el que peor resultados se obtienen. El resto de algoritmos considerados tienen una desviación que se encuentra en un rango de valores muy óptimo y ajustado.
- Por otro lado, todos los algoritmos son muy eficientes, con un tiempo de ejecución de media por debajo de 1 segundo, a excepción de *ILS-ES* que casi dobla este valor.
- Dejando el *Greedy* de lado, obtenemos dos grupos de algoritmos, los que están basados en trayectorias simples (*BL y ES*) y los que están basados en trayectorias múltiples (*BMB, ILS e ILS-ES*), caracterizados por ejecutar las técnicas de trayectorias simples varias veces.

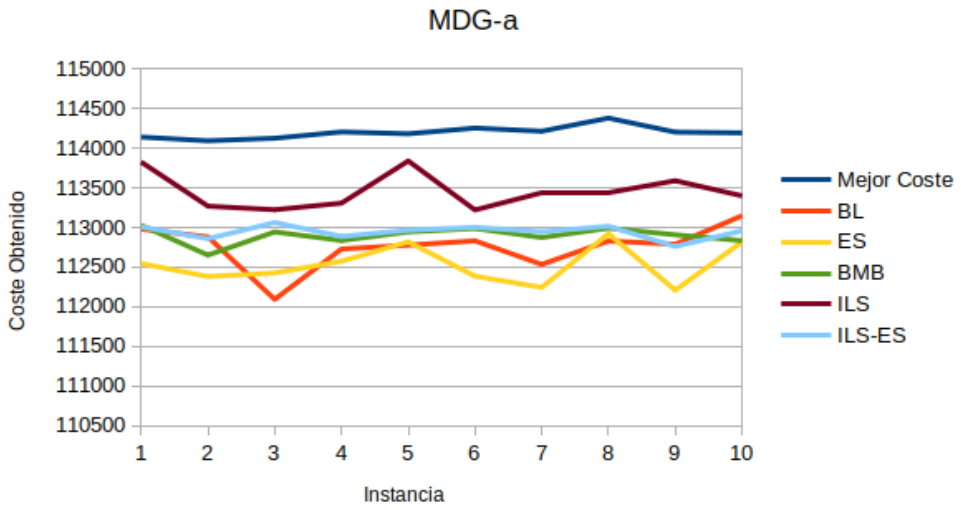
  Teniendo en cuenta este punto, los resultados obtenidos cuadran con el razonamiento teórico, ya que *BMB, ILS e ILS-ES* reducen en torno al 40 % la desviación obtenida de las técnicas basadas en trayectorias simples.

Recordemos los casos seleccionados para la ejecución de los algoritmos: Mostramos los resultados obtenidos en gráficas diferentes según el grupo al que pertenezca cada instancia para reflejar mejor las diferencias:

- `GKD-c`: Conjunto de datos, referentes al grupo de distancias Euclideas con $n = 500$, $m = 50$, en el que mejor resultados se han obtenido, tanto los algoritmos basados en técnicas de trayectorias simples como múltiples llegan al mejor coste. Hay convergencia con todos algoritmos en todas las instancias, de hecho, se aprecia que todas las líneas están superpuestas, obteniendo prácticamente una desviación nula.

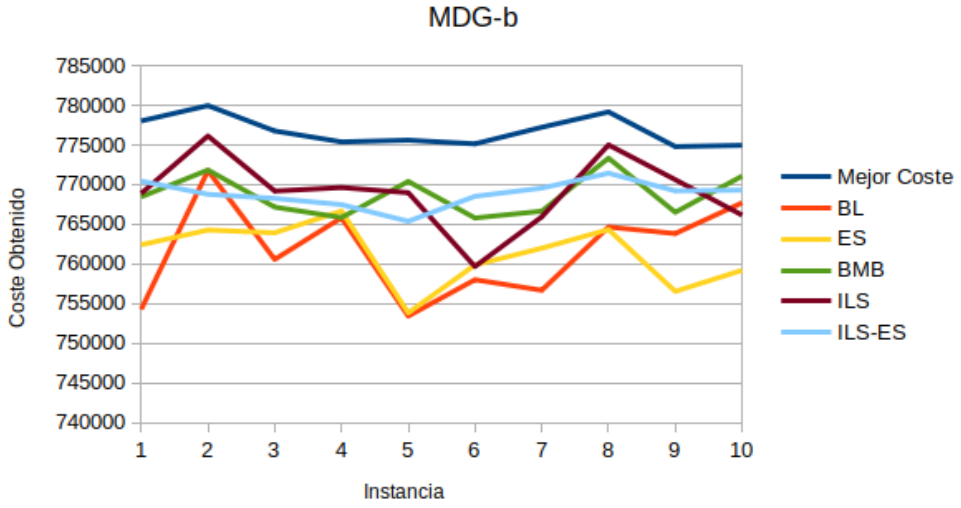


- `MDG-a`: Conjunto de datos correspondiente al grupo de distancias enteras y tamaño de $n = 2000$ y $m = 200$. En todas las instancias los costes obtenidos por los algoritmos basados en técnicas de trayectorias múltiples superan a los que están basados en técnicas de trayectorias simples. Se aprecia como la línea de *ILS* se encuentra por encima del resto, mientras la roja y amarilla, pertenecientes a los algoritmos *BL y ES*, se encuentran por debajo.



- `MDG-b`: Conjunto de datos correspondiente al grupo de distancias reales y tamaño $n = 500$ y $m = 50$. Al igual que en el grupo de instancias anterior, los algoritmos basados en trayectorias simples (BL y ES) se encuentran por debajo del resto.

  Si observamos el trazado por cada uno de los algoritmos, observamos que se presentan altibajos o fluctuaciones en las mismas instancias. Esto es debido a la presencia de óptimos locales, haciendo costoso que nuestros algoritmos puedan salir de estos.

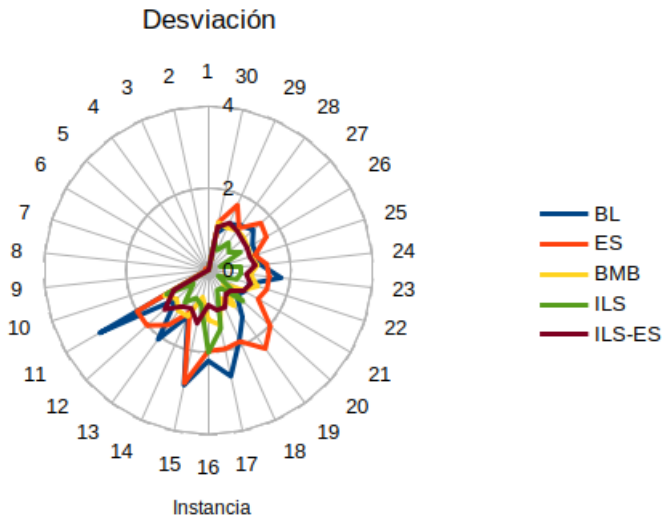


Un aspecto diferente en los algoritmos implementados radica en el número de soluciones usadas en el proceso de optimización. *Enfriamiento Simulado* o *Búsqueda Local* utilizan una sola solución en el proceso de búsqueda.

Por otro lado, técnicas como *Búsqueda Multiarranque Básica* o *Búsqueda Local Reiterada* optimizan un conjunto de soluciones iniciales creadas de forma aleatoria por lo que se obtienen soluciones de más calidad debido a la ventaja que supone explorar diferentes áreas del espacio de búsqueda.
Debido a la posible presencia de óptimos locales, algoritmos como *Búsqueda Local Reiterada* (*ILS*) presentan un factor de mutación brusco, mejorando en gran medida los resultados obtenidos.

11.1. Desviación

Por último, reflejados en el siguiente gráfico en red, mostramos la desviación obtenida de todas las instancia tras la ejecución de cada algoritmo. Como bien sabemos, cuanto menor sea la desviación, mejor será el coste obtenido con nuestro algoritmo. Por tanto, el algoritmo que presente un mayor área bajo su trazado, será peor.



En resumen, podemos considerar que tanto la eficacia como la eficiencia de todos nuestros algoritmos es muy buena por tener una desviación y un tiempo muy pequeños.
Con los que mejor resultados se obtienen son con los algoritmos basados en técnicas de trayectorias múltiples, especialmente el algoritmo de *Búsqueda Local Reiterada* (*ILS*) ya que en media obtiene soluciones más cercanas al mejor valor conocido.

Referencias

-Seminario 1
-Seminario 2
-Seminario 3
-Seminario 4
-Guión de prácticas