

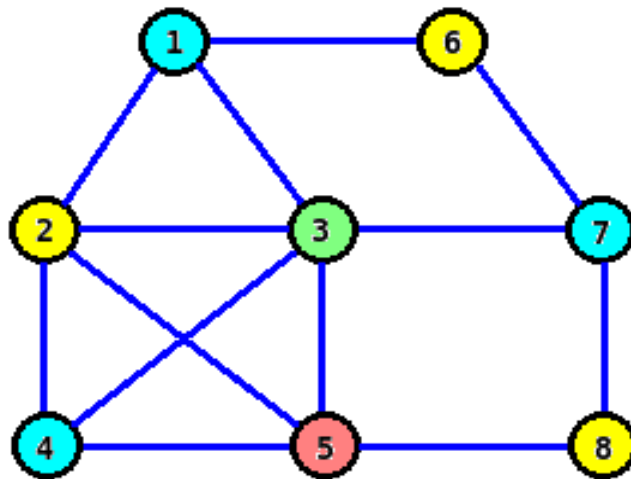
Tarea 5 Metaheurísticas. Metaheurística de Trayectoria para el FAP

Introducción.

En esta practica se implementa una Metaheurística de trayectoria para el problema de Fixed-Spectrum Frequency Assignment. Se realizan 30 ejecuciones de 1 hora para cada una de las instancias GSM2, reportando el mejor resultado cada 2 minutos, además se realiza un estudio de parámetros para determinar la robustez del esquema que hemos implementado. La Metaheurística seleccionada para resolver el problema fue Guided Local Search (GLS), utilizando Stochastic Hill Climbing como método de búsqueda local.

Planteamiento.

El AFP puede ser atacado como un problema de coloración de grafos. Se tiene un grafo $G=(V,E,D,P)$ donde V es un conjunto de vértices y E el conjunto de aristas que los conectan, de tal manera que los bucles (vértices conectándose a si mismos) y aristas múltiples entre vértices no se permiten; D representa una relación de “pesos” por aristas, los cuales definen las restricciones de separación entre las frecuencias y P representa las penalizaciones por interferencias que se producen al no cumplir con la correspondiente restricción. Por tanto, una coloración de G es un mapeo $c:V(G)\rightarrow F$ donde F es un conjunto de “colores” (enteros positivos) de tamaño definido, y para cada arista $e_{ij}\in E$ se incurre en una penalización $p_{ij}\in P$ si $|c_i - c_j| \leq d_{ij} \in D$ (si se viola la restricción de separación). Definimos el costo de c como la interferencia total generada por cada una de las penalizaciones incurridas.



Grafo de 8 vértices coloreado (2 vértices vecinos con diferente color)

Stochastic Hill Climbing.

Es una técnica de optimización matemática que pertenece a la familia de los algoritmos de búsqueda local. Es un algoritmo iterativo que comienza con una solución arbitraria a un problema, luego intenta encontrar una mejor solución variando incrementalmente un único elemento de esta. Si el cambio produce una mejor solución, otro cambio incremental se le realiza a la nueva solución, repitiendo este proceso hasta que no se puedan encontrar mejoras.

```

function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                     neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] < VALUE[current] then return STATE[current]
    current ← neighbor
  end

```

Hill Climbing es bueno para encontrar un óptimo local (una solución que no puede ser mejorada considerando una configuración de la vecindad) pero no garantiza encontrar la mejor solución posible (el óptimo global) de todas las posibles soluciones (el espacio de búsqueda). La característica de que sólo el óptimo local puede ser garantizado puede ser remediada utilizando reinicios (búsqueda local repetida), o esquemas más complejos basados en iteraciones, como búsqueda local iterada, en memoria, como optimización de búsqueda reactiva y búsqueda tabú, o modificaciones estocásticas, como **simulated annealing**.

Guided Local Search (GLS)

En este método se modifica la función objetivo añadiendo penalizaciones con el fin de que la búsqueda “escape” de óptimos locales; para esto es necesario definirse ciertas características que dependen del problema, por ejemplo, la presencia de cierta interferencia entre dos transmisores. A cada característica se le asocia un costo y una penalización. La idea es “guiar” la búsqueda en base a información nueva no incorporada originalmente a la función objetivo, así, cuando lleguemos a un óptimo local penalizaremos alguna de sus características para hacer esta solución menos “deseable” en el futuro.

Dada una función objetivo $G(s)$ que mapea cada solución candidata s a un valor numérico, GLS define una función $H(s)$ que será usada en la búsqueda local en lugar de $G(s)$:

$$H(s) = G(s) + \lambda \sum [p_i I_i(s)]$$

donde s es una solución candidata, λ es un parámetro de escalamiento para GLS, i es el índice de las características, p_i es la penalización de la característica i (todas las p_i se inicializan en cero) e I_i toma un valor de 1 o 0 dependiendo si la característica i está presente o no en la solución.

Otro factor que se toma en cuenta es el valor actual de la penalización de cierta característica, la utilidad de penalizar la característica i , u_i bajo un mínimo local s' , se define de la siguiente manera:

$$u_i(s') = I_i(s') \frac{c_i}{1 + p_i}$$

donde c_i es el costo y p_i es el valor actual de penalización de la característica i . Mientras más alto sea el costo de esta característica, mayor será la utilidad de penalizarla, además, mientras más se penaliza una

característica, menor es la utilidad de volver a penalizarla. Una vez obtenido un mínimo local, la característica con la utilidad mas alta sera penalizada, incrementando p_i en 1. La escala de penalización se ajusta con λ .

Algorithm 2.14 Template of the guided local search algorithm.

Input: S-metaheuristic LS , λ , Features I , Costs c .

$s = s_0$ /* Generation of the initial solution */

$p_i = 0$ /* Penalties initialization */

Repeat

 Apply a S-metaheuristic LS ; /* Let s^* the final solution obtained */

For each feature i of s^* **Do**

$u_i = \frac{c_i}{1+p_i}$; /* Compute its utility */

$u_j = \max_{i=1,..m}(u_i)$; /* Compute the maximum utilities */

$p_j = p_j + 1$; /* Change the objective function by penalizing the feature j */

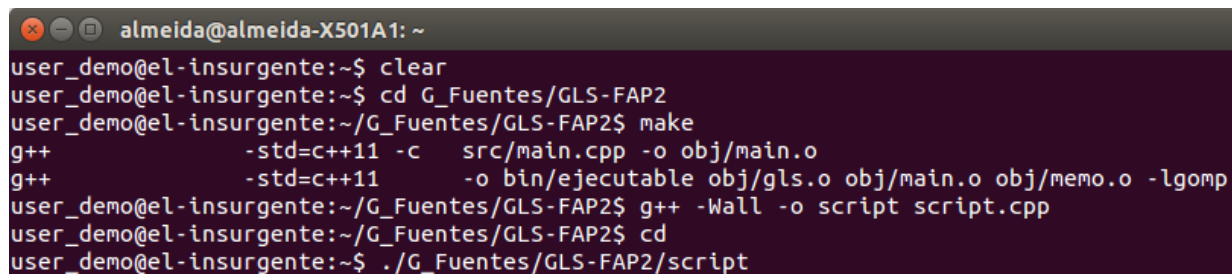
Until Stopping criteria /* e.g. max number of iterations or time limit */

Output: Best solution found.

Implementación.

El programa que se implemento, lee los archivos de las instancias GSM2, y ejecuta el algoritmo de Búsqueda Local Guiada (GLS) durante 1 hora, guardando el mejor resultado obtenido cada 2 minutos en un archivo de resultados. Se genera un archivo Makefile para compilar el programa y sus librerías, así como de un script de ejecuciones que utiliza un archivo de tares y un archivo de maquinas para ejecutar las 30 repeticiones de cada configuración del FAP.

El algoritmo ejecuta 10000 iteraciones de búsqueda local, y penaliza el mínimo local obtenido de acuerdo al método mostrado anteriormente, si por 100 iteraciones no se obtiene ninguna solución que mejore la mejor solución actual, el algoritmo se reinicia con una solución aleatoria y se repite el procedimiento.

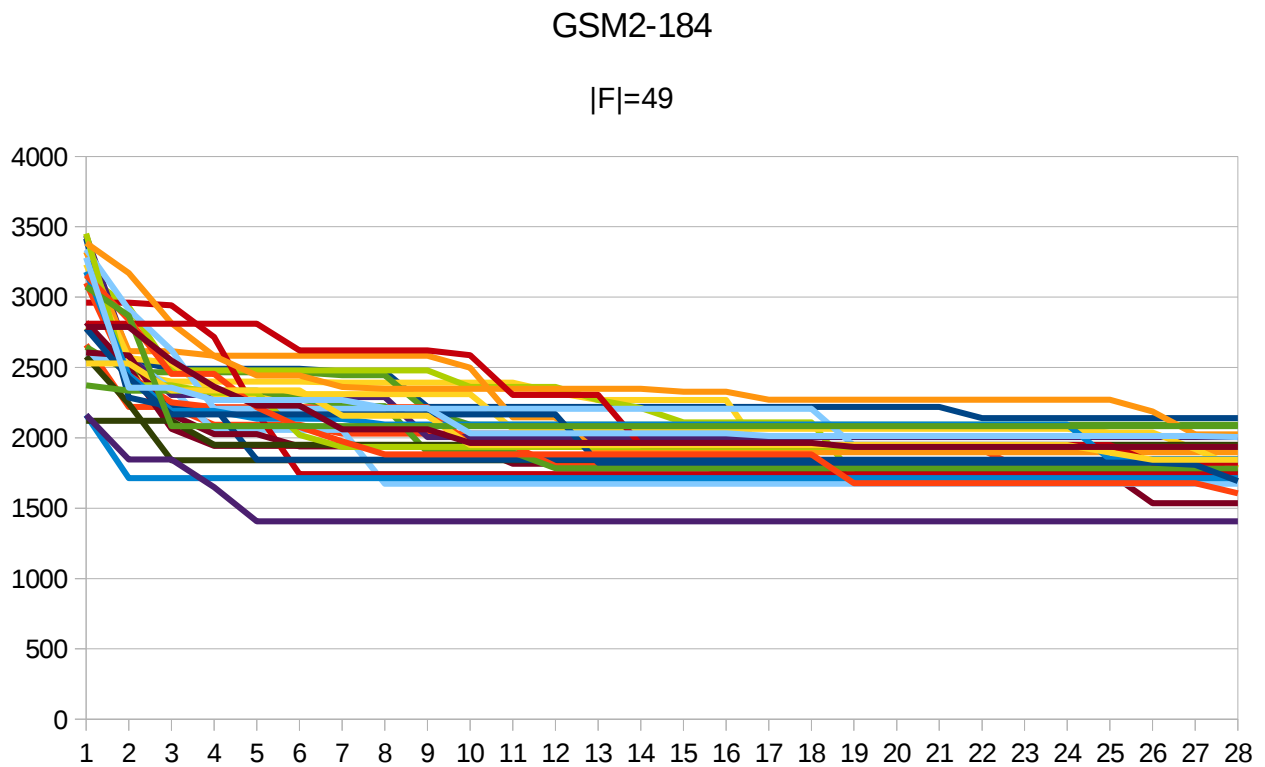
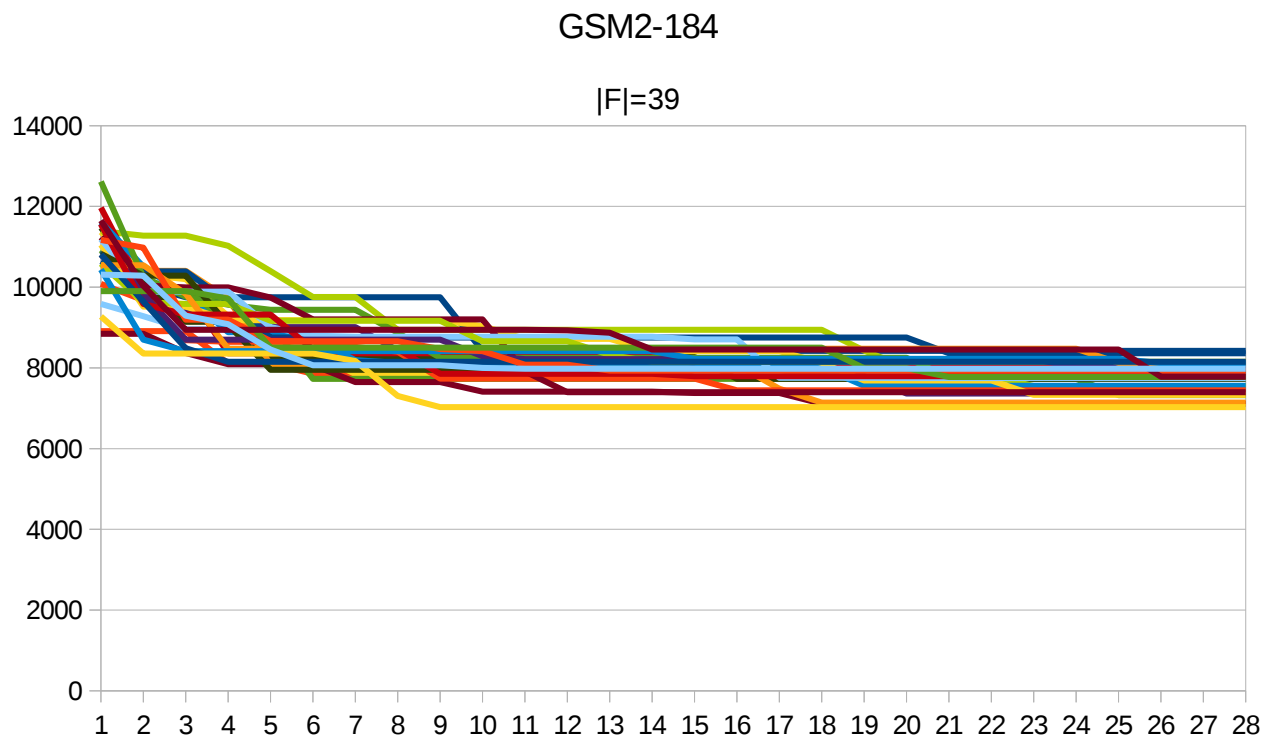


```
almeida@almeida-X501A1: ~
user_demo@el-insurgente:~$ clear
user_demo@el-insurgente:~$ cd G_Fuentes/GLS-FAP2
user_demo@el-insurgente:~/G_Fuentes/GLS-FAP2$ make
g++          -std=c++11 -c    src/main.cpp -o obj/main.o
g++          -std=c++11      -o bin/ejecutable obj/gls.o obj/main.o obj/memo.o -lgomp
user_demo@el-insurgente:~/G_Fuentes/GLS-FAP2$ g++ -Wall -o script script.cpp
user_demo@el-insurgente:~/G_Fuentes/GLS-FAP2$ cd
user_demo@el-insurgente:~$ ./G_Fuentes/GLS-FAP2/script
```

Compilación del programa, y compilación y ejecución del script.

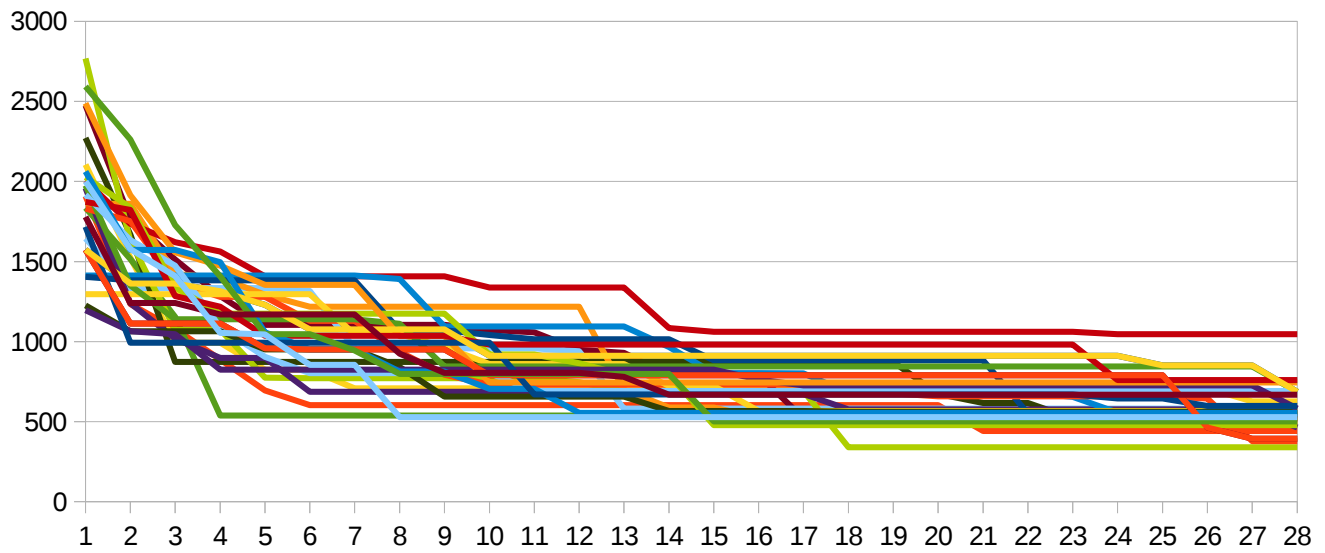
Resultados.

Gráficas de Evolución del Fitness para cada una de las instancias ejecutadas:



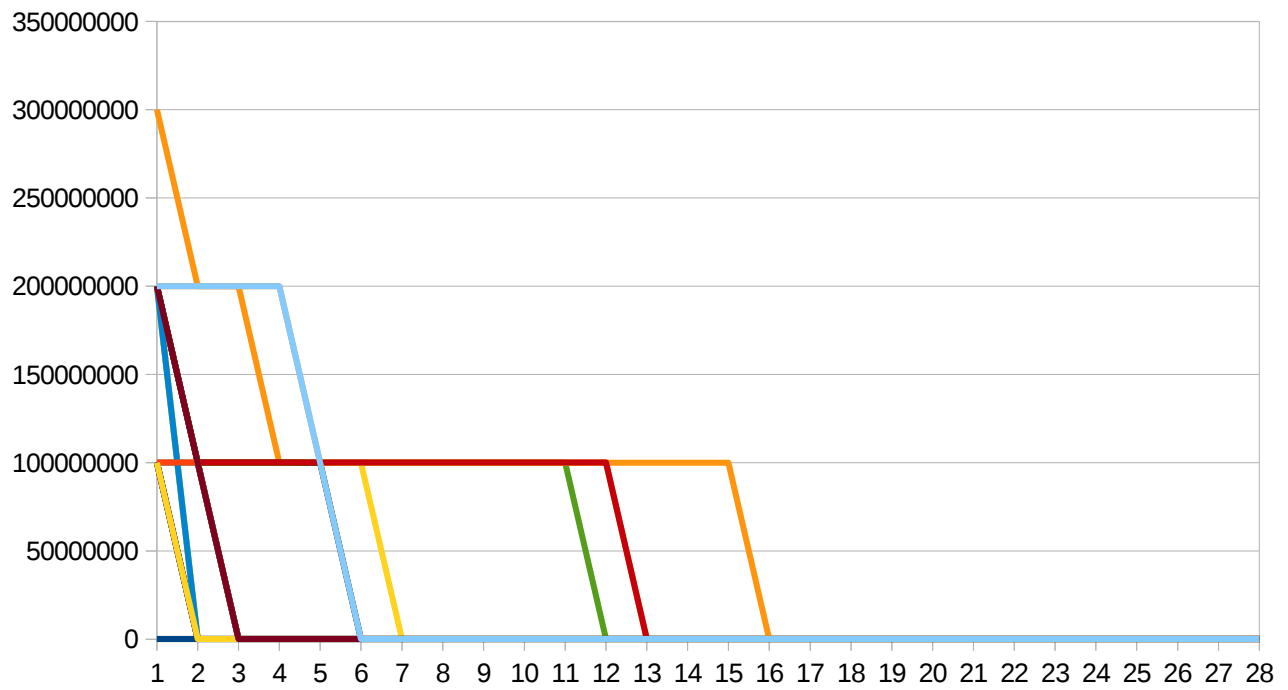
GSM2-184

$|F|=52$



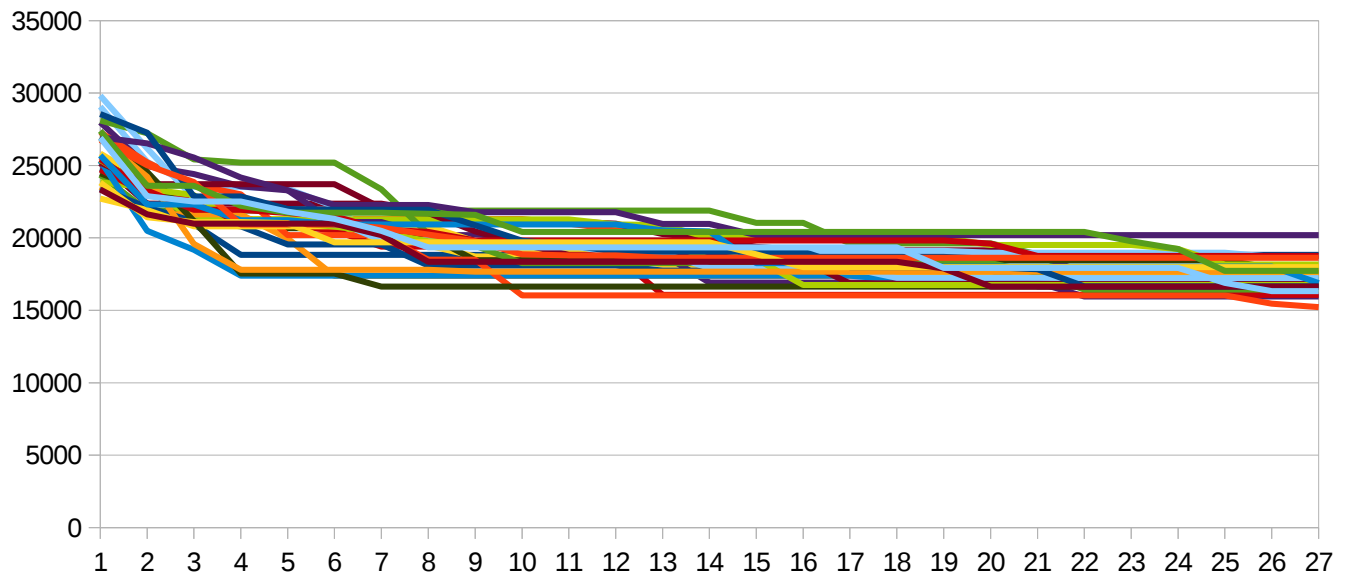
GSM2-227

$|F|=29$



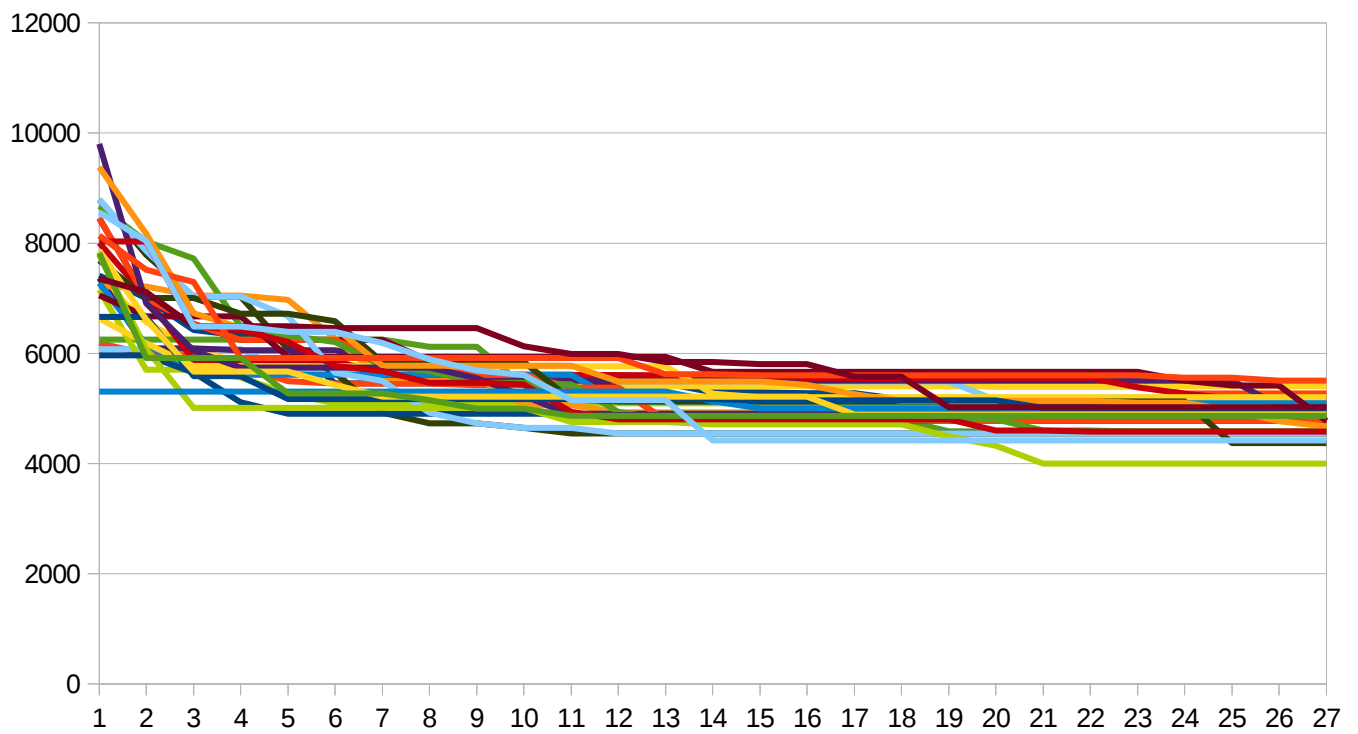
GSM2-227

$|F|=39$



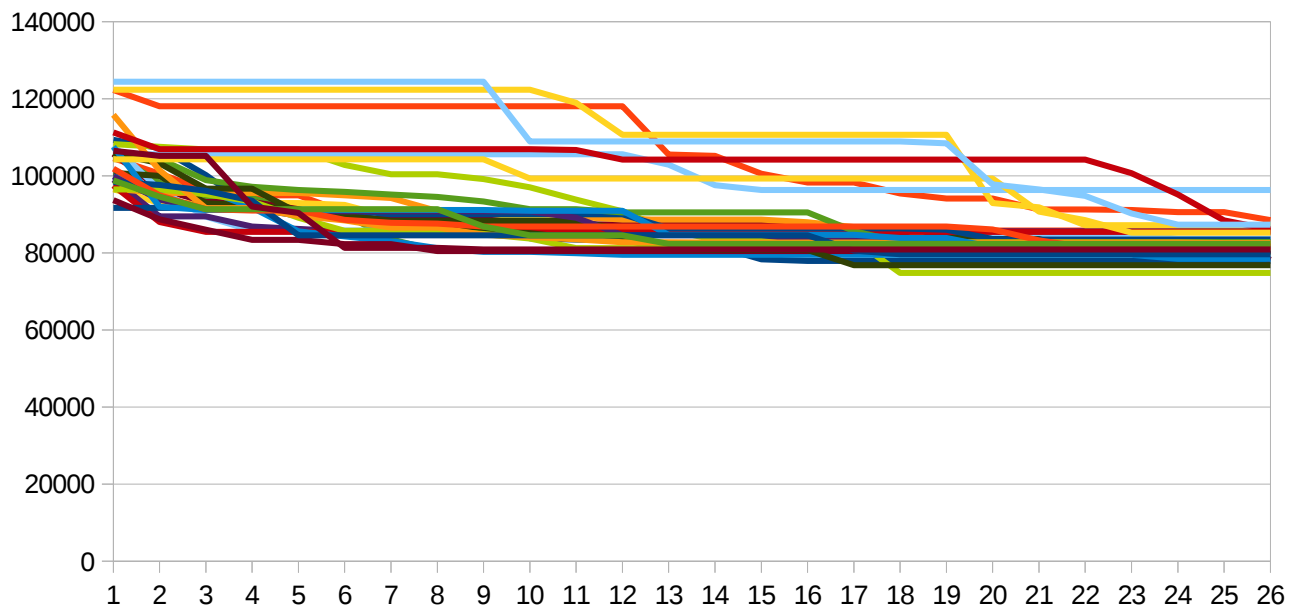
GSM2-227

$|F|=49$



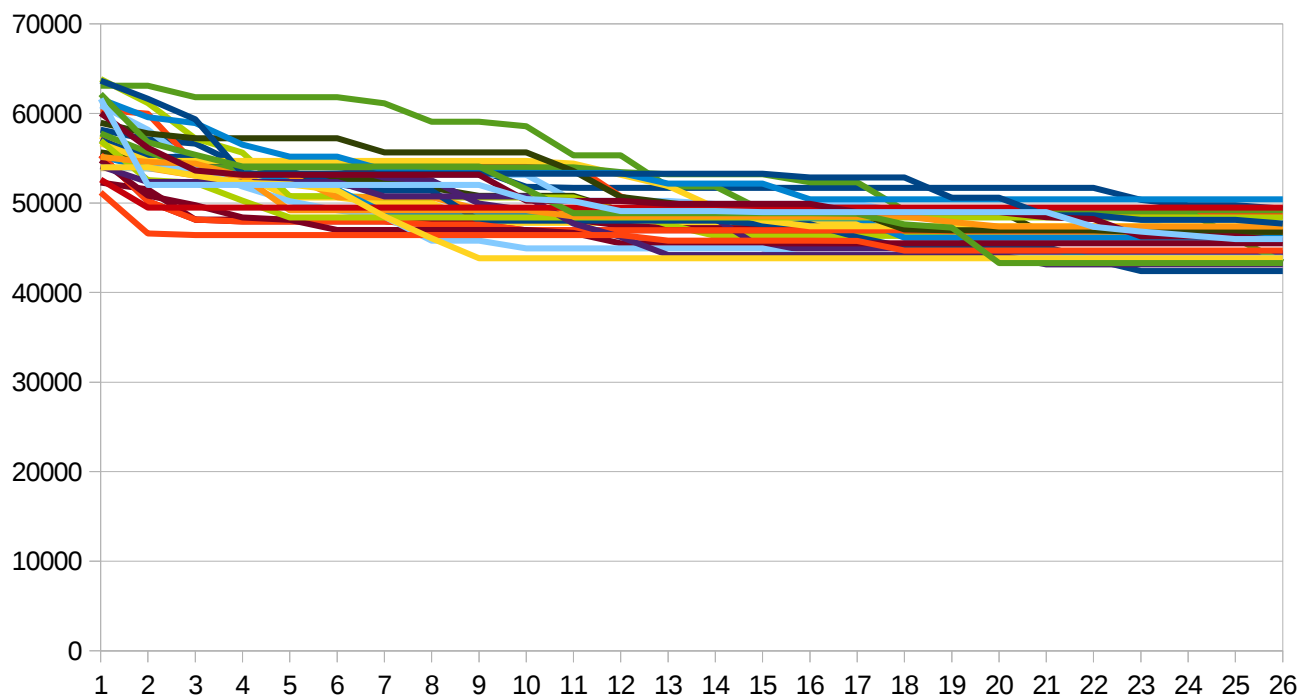
GSM2-272

$|F|=34$



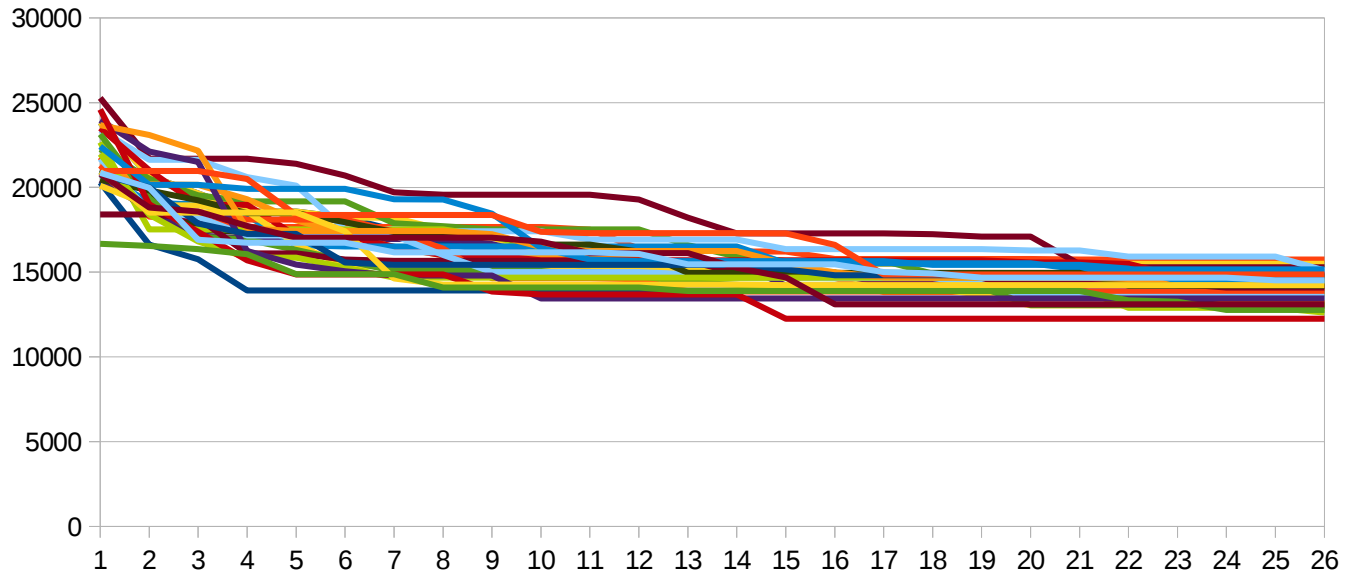
GSM2-272

$|F|=39$



GSM2-272

$|F|=49$



A continuación se muestran los resultados obtenidos de las 30 ejecuciones, se registra el mejor valor obtenido en cada ejecución:

GSM2-184			GSM2-227			GSM2-272		
39	49	52	29	39	49	34	39	49
8423	2006	501	102989	17382	4916	74782	46218	13180
7791	1897	464	102513	15966	5061	80133	44379	14202
7848	1731	339	104033	18158	5049	83638	46259	12621
7324	1674	582	90233	18698	4546	78181	46316	14245
7116	1747	519	89201	17592	4777	80496	46525	15094
7871	1536	519	94201	17305	5039	81280	44039	15286
7729	2092	538	91151	17566	4585	85778	44230	13746
7552	1725	627	105037	18018	5405	81642	49172	15500
7364	1773	441	103075	17781	5271	85610	48565	15738
7139	2139	687	89201	18823	4790	77914	49380	15500
7558	1936	478	93069	16758	4000	81256	46761	14671
8148	1951	562	109507	17221	4372	76794	48385	14201
7952	1938	691	108260	16639	4546	96334	43389	13545
7748	1945	394	119265	16861	4775	80449	43884	14093
7397	1783	687	101779	16440	5203	87170	45505	13051
7764	1784	566	104082	17667	4582	82831	47409	13043
7335	1798	378	100706	15226	4777	88472	46938	13857
7444	1844	551	101843	16660	5128	83433	42379	13008
8360	1713	556	101624	17127	4907	78468	46109	14754
8110	1741	1046	97280	16077	5181	85447	46938	15160
7981	2005	668	94253	16329	4421	80920	45962	14513
7775	1935	501	92554	16638	5023	87345	45782	13103
7784	1844	599	106206	17732	4892	82372	43286	12770
7025	2084	555	96294	17975	4867	85259	43828	14227
7929	1605	760	100321	16668	5508	81493	44685	14868
8149	1693	529	106169	18637	5011	86049	47709	14813
8121	1949	745	110374	16876	4924	79639	50404	14503
7791	1848	581	96805	18764	4582	81196	47362	12245
8125	2024	619	98592	17671	4671	82533	49479	14598
7967	1406	716	104741	20195	4882	81816	43151	13455

En la siguiente tabla se muestra una comparativa entre los resultados que obtuvimos en esta practica y los resultados obtenidos en el articulo de Path Relinking:

Instance	F	GLS		rPR		mrPR	
		C_prom	C_best	C_prom	C_best	C_prom	C_best
GSM2-184	39	7754	7025	5258.1	5250	5266.1	5258
GSM2-184	49	1838	1406	874	874	874	874
GSM2-184	52	580	339	162	162	162	162
GSM2-227	29	100512	89201	59405.8	57731	59573.5	56955
GSM2-227	39	17382	15226	9121.6	8772	9201.1	8809
GSM2-227	49	4856	4000	2004.7	1998	2015.5	1998
GSM2-272	34	82624	74782	54570.2	53080	54795.3	53688
GSM2-272	39	46148	42379	27475	26237	28135.1	26453
GSM2-272	49	14120	12245	7128.5	6997	7208	7056

Estudio Paramétrico.

A continuación realizaremos un estudio de robustez en función del parámetro de escalamiento λ . Por la estructura del método que estamos utilizando, son pocos los parámetros que podemos regular para ajustar nuestro algoritmo, en nuestra implementación, se ajustaron también el número de iteraciones de nuestra búsqueda local y cada cuando hacemos un reinicio total. Se presenta una tabla de donde se muestra el promedio de los mejores resultados obtenidos para diferentes valores del parámetro para dos instancias del FAP, en general se observa que para valores entre 1×10^4 y 1×10^6 no se aprecia que el algoritmo sea sensible a λ , pero debido a que las penalizaciones son muy grandes, un valor por debajo de estos no afecta demasiado en la penalización total.

Parámetro	GSM2-184-39	GSM2-227-39
1000000	7804	15962
100000	7906	17333
10000	7773	15367
1000	8788	18931
100	8854	23829
10	10726	24945

Conclusiones.

Aunque no se pudieron igualar los resultados del artículo, se mejoraron mucho respecto a la práctica anterior donde se implementó una heurística constructiva para este mismo problema. Un aspecto que pudo haberse modificado para posiblemente obtener mejores resultados sería nuestra definición de vecindad, pues en nuestra implementación, una solución vecina es solo un cambio aleatorio en el valor de frecuencia de una de los transmisores, por lo que una vecindad que permita una mayor diversidad podría mejorar el resultado.