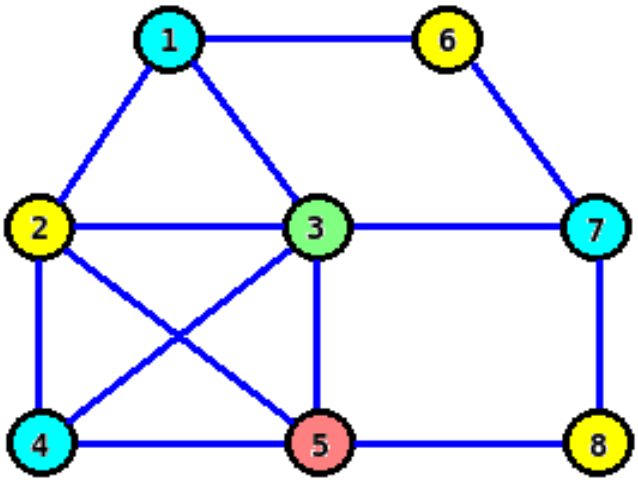
Tarea 2 Metaheurísticas. Heurísticas constructivas para el AFP

MÉTODO 1

La primera heurística constructiva utilizada para resolver el problema de asignación de frecuencias fue el método DSATUR, tomado del artículo “Frequency Allocation for WLANs Using Graph Colouring Techniques”, de Janne Riihijarvi, Marina Petrova y Petri Mahonen, con la modificación para el caso Fixed Sprectrum descrita en “Adaptive Memory Programming: An Empirical Study with the Bandwith Coloring Problem”, de Rafael Marti, Francisco Gortazar y Abraham Duarte.

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)

Heurística constructiva: DSATUR

Es un método que consiste en ir asignando “colores” a cada vértice del grafo, dependiendo de una variable llamada grado de saturación s_i, la cual, en su forma básica se define para cada vértice como el numero de vecinos de diferente color, o de forma equivalente como el numero de colores no admisibles para un vértice. En nuestro caso estamos manejando un “peso” para cada arista, por lo cual definimos el grado saturación como el numero de colores no admisibles para cada vértice en cada momento de la coloración, tomando en cuenta la restricción de cada arista

$$|c_i-c_j|\notin T_{ij} \qquad T_{ij} = \textit{colores no permitidos}$$

Algoritmo:

1. Inicializar $s_i = 0 \; \forall \, i \in V$
2. Elegir $i_{max} = arg\,max_{i \in V}\{s_i\}$
si hay empate, elegir $i_{max} = arg\,max_{i \in V}\{N_i\}$
3. $Vi_{max} \leftarrow$ *frecuencia mínima admisible* $f_{min} \in F$
si no hay ninguna $f_{min} \in F$ *admisible:*
elegir $f_{min} = arg\,min_{f \in F}\{p_{j,i_{max}}\}$ *con* $j \in U$
(U=conjunto de vértices asignados)
4. $V \leftarrow V - Vi_{max}, U \leftarrow U + Vi_{max}$
5. Actualizar frecuencias admisibles y vecindades N_i
6. Actualizar grados de saturación $s_i \; \forall \, i \in V$
s_i *= numero de frecuencias no admisibles para cada vértice*
7. Repetir 2-6 hasta $V = \emptyset$

METODO 2

El segundo método utilizado consiste en una modificación del algoritmo DSATUR, con el cual se consiguieron algunas mejoras en los resultados, a cuenta de un incremento del tiempo de ejecución:

- Se omite el calculo del grado de saturación como se había propuesto, en su lugar se calcula de acuerdo al articulo " An Empirical Study with the Bandwith Coloring Problem":

$$s_i = \sum_{h=1}^{k} max\{d_{ij} : j \in N_i, c_j = h\}$$

donde N_i representa el conjunto de vértices adyacentes a *i* y *k* es el color máximo utilizado. Por tanto el grado de satura queda definido como la suma de las distancias máximas entre $v_i \in V$ y cada color adyacente.

Algoritmo:

1. Inicializar $s_i = 0 \; \forall \, i \in V$
2. Elegir $i_{max} = arg\,max_{i \in V}\{s_i\}$
si hay empate, elegir $i_{max} = arg\,max_{i \in V}\{N_i\}$
3. $Vi_{max} \leftarrow$ *frecuencia mínima admisible* $f_{min} \in F$
si no hay ninguna $f_{min} \in F$ *admisible:*
elegir $f_{min} = arg\,min_{f \in F}\{p_{j,i_{max}}\}$ *con* $j \in U$
(U=conjunto de vértices asignados)
4. $V \leftarrow V - Vi_{max}, U \leftarrow U + Vi_{max}$
5. Actualizar frecuencias admisibles
6. Actualizar grados de saturación $s_i \; \forall \, i \in V$
$s_i = \sum_{h=1}^{k} max\{d_{ij} : j \in N_i, c_j = h\}$
7. Repetir 2-6 hasta $V = \emptyset$

Resultados. A continuación se muestra una tabla comparativa de los dos métodos implementados (versión original y nueva) asimismo se incluyen los resultados obtenidos en el articulo “Path relinking for the fixed spectrum frequency assignment problem”:

	DSATUR		DSATUR modified		Path Relinking (rPR)			Path Relinking (mrPR)		
Instances	Cost	Time	Cost	Time	Best Cost	Avg Cost	Time	Best Cost	Avg Cost	Time
GSM2-184 |F|=39	18194	0.035	14108	1.022	5250	5258.1	1529	5258	5266.1	1578
GSM2-184 |F|=49	3990	0.015	4004	0.813	874	874	672	874	874	898
GSM2-184 |F|=52	2285	0.066	2131	0.859	162	162	295	162	162	425
GSM2-227 |F|=29	700138924	0.183	800074213	1.075	57731	59405.8	3131	56955	59573.5	3091
GSM2-227 |F|=39	31148	0.081	26113	1.246	8772	9121.6	2611	8809	9201.1	2251
GSM2-227 |F|=49	5730	0.052	11374	1.553	1998	2004.7	2230	1998	2015.5	2396
GSM2-272 |F|=34	100141973	0.154	117480	1.92	53080	54570.2	5513	53688	54795.3	5344
GSM2-272 |F|=39	80458	0.077	67208	2.141	26237	27475	6278	24453	28135.1	5368
GSM2-272 |F|=49	28336	0.143	25129	2.751	6997	7128.5	5798	7056	7208	4480

Como se puede apreciar, el algoritmo modificado es en general mejor que el algoritmo original, con alguna excepciones, también se aprecia que no se pudo obtener un resultado favorable para el caso de 227 transmisores con 29 frecuencias, posiblemente se hubiera podido hacer una mejora generando un sub-grafo con los vértices mas penalizados para después resolverlo de forma greedy y acoplarlo al grafo original, posteriormente se trataran de mejorar las heurísticas para este problema.

Compilación/Ejecución. El programa compila con `gcc`. Archivo Makefile incluido, el comando `make` compila los ficheros `main.c` y la librería `dsatur.h` contenidas en la carpeta “src”, y genera los objetos en la carpeta “obj” y el ejecutable en “bin”; `make run` ejecuta el programa, se utiliza la directiva `args` para pasar los argumentos:

```
almeida@almeida-X501A1: ~/DSATUR
almeida@almeida-X501A1:~$ cd /home/almeida/DSATUR
almeida@almeida-X501A1:~/DSATUR$ make
gcc               -std=c99       -c       src/dsatur.c -o obj/dsatur.o
gcc               -std=c99       -c       src/main.c -o obj/main.o
gcc               -std=c99       -c       src/memo.c -o obj/memo.o
gcc               -std=c99        -o bin/ejecutable obj/dsatur.o obj/main.o obj/m
emo.o -lm
almeida@almeida-X501A1:~/DSATUR$ make run args="GSM2-184.ctr 184 39"
./bin/ejecutable GSM2-184.ctr 184 39
Solucion:
39 36 33 30 15 13 10 7 4 1 20 17 10 7 1 14 11 8 5 2 13 10 7 4 1 20 17 38 35 32 2
5 22 15 12 3 37 34 31 28 16 12 9 6 3 35 32 5 16 24 21 18 15 12 14 27 24 20 17 39
 19 11 8 2 33 14 11 8 29 25 20 17 14 24 16 13 4 27 24 21 18 14 11 8 5 24 21 18 1
5 15 37 34 31 28 25 22 19 2 32 29 26 23 4 38 35 32 29 26 12 9 6 3 5 37 34 31 29
26 23 20 17 10 7 4 1 2 28 25 22 19 26 23 38 35 32 14 11 8 2 28 25 22 19 5 37 34
31 16 16 12 9 6 3 25 22 16 13 6 37 29 26 23 28 25 22 19 30 27 21 18 21 18 12 9 3
 6 38 35 32 9 39 36 30 15 2


Costo minimo:
18194
almeida@almeida-X501A1:~/DSATUR$
```