COLOREO DE MAPAS
ALGORITMO DE ELIMINACIÓN DE POSIBILIDADES

Luciano García Bes
lucianogarciabes@hotmail.com

Resumen

El problema del coloreo de mapas planos tiene sus orígenes en la década de 1850. Muchos matemáticos han dedicado parte de sus vidas en demostrar cuál es la menor cantidad de colores que se deben utilizar para pintar un mapa plano sin que dos regiones limítrofes estén pintadas con el mismo color. Pero fue recién en 1976, y gracias a la ayuda de las computadoras, se pudo demostrar que la menor cantidad de colores que podían utilizarse era 4.

En el presente trabajo se trata de explicar con la mayor claridad posible el camino seguido para desarrollar cuatro algoritmos para el coloreo de mapas planos con tan solo 4 colores.

El primer algoritmo trabaja de una forma intuitiva tratando de detectar cuales nodos son los que tienen mayor posibilidad de generar un error y pintándolos en las primeras instancias.

En el segundo se investiga el motivo por el cuál el primer algoritmo falla al incrementar la complejidad del mapa y se modifica el algoritmo para que pueda pintar sin problemas el nuevo mapa.

Pero éste nuevo algoritmo presenta problemas con un mapa de una complejidad aún mayor. Por esto se desarrolla un nuevo algoritmo basado en el algoritmo anterior que, haciendo uso de la recursividad, es capaz de colorear el mapa mas complejo.

Y el algoritmo final es una optimización del algoritmo recursivo que baja el tiempo de ejecución para los mapas mas complejos y no provoca cambios considerables en el tiempo de ejecución para los mapas mas sencillos.

Todos los algoritmos han sido probados con los mapas de Argentina, África y España para ver sus rendimientos. En el punto 6 se muestra una tabla con los resultados obtenidos en dichas pruebas.

Introducción[1]

El problema de colorear un mapa plano con la menor cantidad posible de colores de manera que no hubiesen dos regiones limítrofes con el mismo color ha sido un problema de interés desde hace mucho tiempo. Pero el teorema de los cuatro colores fue planteado por Francis Guthrie[2] a principios de la década de 1850.

Muchos matemáticos intentaron demostrar el teorema de Guthrie (que hasta ese entonces no era mas que una conjetura) y el 13 de Junio de 1878 Arthur Cayley envió una carta a la Sociedad Matemática de Londres preguntando si la conjetura de los cuatro colores ya había sido resuelta.

El 17 de Julio de 1879, Alfred Bray Kempe[3] anunció que tenía una demostración de la Conjetura de los Cuatro Colores. El Teorema y su demostración fueron publicados en el *Periódico Americano de Matemáticas* en 1879.

En 1890 un conferencista de Durham, Inglaterra, llamado Percy John Heawood publicó el ensayo *El Teorema del Coloreo de Mapas* donde mostraba que la prueba de Kempe estaba era errónea.

Fue en 1976 cuando, finalmente, se resolvió la Conjetura de los Cuatro Colores. La prueba fue obtenida por Appel y Haken, quienes trabajaron sobre los métodos que había utilizado Kempe y otros métodos creados en los últimos años. Appel y Haken utilizaron 1200 horas de computadora para trabajar en la prueba final.

El Teorema de los Cuatro Colores fue le primer gran teorema en ser demostrado usando una computadora, teniendo una demostración que pudo ser verificada directamente por los matemáticos. Los detalles de la demostración aparecieron en dos publicaciones en 1977. Trabajos recientes han llevado a mejoras en el algoritmo.

En el presente trabajo se muestra el desarrollo de tres algoritmos para el coloreo de mapas y la optimización del último de ellos. Los algoritmos fueron programados en PHP y una versión interactiva del último algoritmo optimizado fue realizada en Macromedia Flash[4].

---

[1] Información tomada del artículo *The four colour theorem* de J.J. O'Connor y E.F. Robertson publicado en http://www-gap.dcs.st-and.ac.uk/~history/HistTopics/The_four_colour_theorem.html

[2]Francis Guthrie (1831 – 1899): Profesor de matemático y abogado ingles. Estudió matemáticas en la universidad de Londres y luego de recibirse estudió derecho. En 1861viajó a Sudáfrica como profesor de matemáticas. Publicó algunos ensayos matemáticos y se interesó en la botánica. Un aparato de calefacción lleva su nombre.

[3] Kempe fue estudiante de matemáticas de Cayley en la Universidad de Cambridge y dedicó gran parte de su vida a distintos estudios matemáticos.

[4] El funcionamiento del último algoritmo optimizado, tanto en PHP como la versión de Flash, puede verse en http://www.angel-gris.com.ar/coloreo. El rendimiento de cada uno de los algoritmos se muestra en el apéndice A del presente trabajo.

1. Problema de coloreo de mapas planos con 4 colores

El problema del coloreo de mapas planos con 4 colores es, básicamente, el problema de pintar un mapa plano con 4 colores de manera que no se presenten 2 regiones limítrofes (o adyacentes) pintadas con el mismo color.

1.1 Grafos Planos

Se puede representar un mapa en forma de grafo si se considera cada región del mapa como un nodo del grafo, y las aristas del grafo representan los límites entre las distintas regiones. Un grafo es plano si sus nodos pueden ser conectados de manera que sus aristas no se crucen.

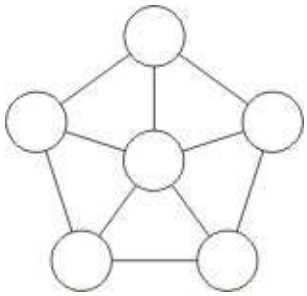

Fig. 1.1 – Grafo Plano

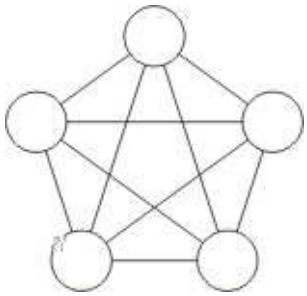

Fig. 1.2 – Grafo no Plano

1.2 Nodos que presentan problemas

Al intentar pintar un grafo, los nodos que presentan problemas son aquellos que cuando van a ser pintados ya tienen los nodos adyacentes pintados con los 4 colores disponibles de manera que no puede ser pintado con ningún color.

Fig. 1.3 – Nodos de Grado 1

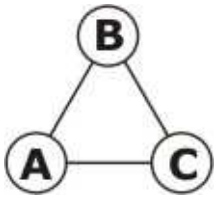

Fig. 1.4 – Nodos de Grado 2

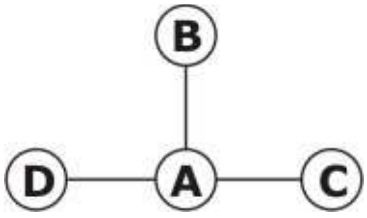

Fig. 1.5 – El nodo A es de grado 3 y los demás son de grado 1

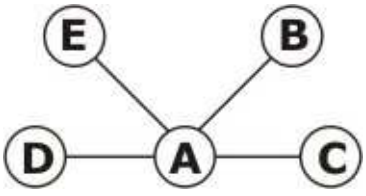

Fig. 1.6 – El nodo A es de grado 4 y los demás son de grado 1

Observando la Fig. 1.3 podemos ver que sin importar el color que tome el nodo B siempre quedarán 3 colores disponibles para pintar A, con lo que vemos que los nodos de grado 1 no presentan problemas. Haciendo análogo estudio sobre la Fig.

1.4, vemos que si el nodo B es pintado con el color 1 y el nodo C con el color 2, el nodo A podrá ser pintado con el color 3 ó con el color 4. Con lo que concluimos que los nodos de grado 2 tampoco presentan problemas.

El nodo de grado 3 de la Fig. 1.5 tampoco presenta problemas, ya que cada uno de los nodos que son adyacentes a éste podrían estar pintados de un color distinto y aún así quedaría un color disponible para pintar el nodo en cuestión. Pero cuando se presenta un nodo de grado 4, como es el caso del nodo A de la Fig. 1.6, se puede presentar un problema si cada uno de sus nodos adyacentes está pintado con un color distinto agotando todos los colores disponibles.

Tras éste análisis podemos concluir que los nodos que presentan problemas son aquellos de grado mayor o igual que 4, y que cuanto mayor sea su grado mas posibilidades tiene de presentar un problema.

2. Algoritmo de eliminación de posibilidades (primer intento)

Teniendo en cuento lo dicho sobre los nodos que presentan problemas este algoritmo de eliminación consiste, en un primer paso, en seleccionar del mapa sin pintar el nodo de mayor grado y pintarlo con cualquiera de los 4 colores disponibles. De esta manera, en los pasos siguientes, estoy limitando a los nodos adyacentes de los ya coloreados a la posibilidad de tomar uno de los colores restantes. O lo que es igual, *elimino* de sus listas de posibles colores candidatos el/los color/es con el que pinté el/los nodo/s adyacente/s.

2.1 El mapa de Argentina

Para ilustrar el algoritmo utilizaremos el mapa de la República Argentina, considerando a la Capital Federal como una provincia mas, con lo que el total de provincias (nodos) serían de 24.

Fig. 2.1 – Mapa de Argentina con la Capital Federal considerada como una provincia

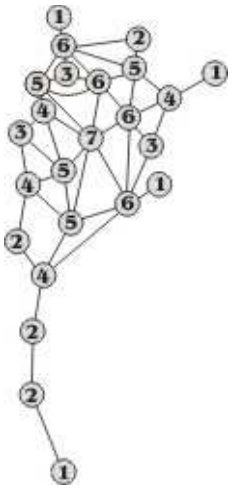

Fig. 2.2 – Grafo asociado al mapa de Argentina. Cada nodo indica su grado

Como se puede observar en el grafo asociado al mapa de Argentina (Fig. 2.2), hay 11 nodos que tienen grado inferior a 4, por lo que no presentan problemas. Y el

nodo de mayor grado tiene grado 7 y es el correspondiente a la provincia de Córdoba.

2.2 Explicación del algoritmo

Como dijimos anteriormente, comenzaremos pintando los nodos que tengan mayor grado. Por esto, el primer nodo con el que trabajaremos será el que corresponde a Córdoba.

Fig. 2.3 – Córdoba y sus límites

Utilizando cuatro colores numerados del 1 al 4 de la siguiente manera: 1 – Amarillo, 2 – Rojo, 3 – Azul y 4 – Verde. (teniendo en cuenta este orden de prioridad de elección). Podríamos utilizar cualquiera de éstos colores para pintar a Córdoba ya que ninguna de los nodos adyacentes está pintado, pero teniendo en cuenta el orden de prioridad de elección del color, lo pintaremos con el color 1.

Fig. 2.3 – Córdoba pintada de Amarillo

Ahora debemos seleccionar otro nodo que tenga el mayor grado posible y que no esté pintado. Córdoba es el único nodo de grado 7 y ya está pintado, por lo que procederemos con los nodos de grado 6.

En la Fig. 2.2 vemos que hay 4 nodos de grado 6. Salta, Santiago del Estero, Santa Fe y Buenos Aires. Tomaremos uno al azar; Santiago del Estero.

Fig. 2.4 – Santiago del Estero y sus límites

Para éste caso vemos que uno de los límites (Córdoba) ya está pintado de amarillo, por lo que Santiago del estero sólo podrá ser pintado con uno de los tres colores restantes. La pintaremos de rojo.

Fig. 2.5 - Santiago del Estero pintada de rojo

Si ahora queremos pintar a Santa Fe, veremos que los dos nodos pintados anteriormente son adyacentes a éste. Por lo tanto no podremos pintarla con el color de Córdoba (Amarillo) ni con el color de Santiago del Estero (Rojo), así que la pintaremos con el tercer color de la lista, el Azul.

Veamos qué sucede si ahora deseo pintar a Buenos Aires.

Fig. 2.6 – Buenos Aires y sus límites

En éste caso podríamos pintar a Buenos Aires de Rojo o de Verde, ya que sus nodos adyacentes ocupan el Amarillo y el Azul. Pero lo pintaremos de Rojo, de manera de guardar el color Verde solamente para casos en los que los nodos adyacentes al seleccionado utilicen los tres colores restantes.

El último nodo de grado 6 que falta pintar es Salta. Como el único nodo adyacente a Salta que está pintado es Santiago del Estero y utiliza el color Rojo, Salta podrá ser pintado con cualquiera de los colores restantes. Pero siguiendo el precepto de utilizar siempre el color de mayor prioridad (o lo que es lo mismo, el color de menor valor numérico asociado) que esté disponible[5], pintaremos a Salta de Amarillo.

[5] Los valores de los colores fueron enunciados al principio del apartado 2.2 y son los siguientes:

1 – Amarillo; 2 – Rojo; 3 – Azul; 4 – Verde.

Se plantea ir coloreando los nodos en forma sucesiva con el color de menor valor que esté disponible. Pero el algoritmo también funciona coloreándolos tomando de forma aleatoria los colores disponibles sin importar su valor, aunque de ésta manera el algoritmo pierda eficiencia.

Fig. 2.7 – Buenos Aires pintada de Rojo

y Salta de Amarillo

Una vez pintados todos los nodos de grado 6, se continúa la tarea con los de grado 5, luego con los de 4, los de 3 y finalmente los de grado 2 y los de 1.

Fig. 2.8 – Una solución encontrada utilizando éste algoritmo

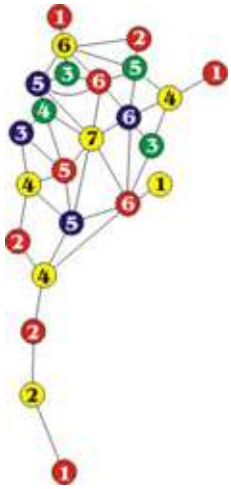

Fig. 2.9 – Grafo asociado a la solución encontrada

Si bien la solución encontrada utilizando éste algoritmo va a variar según el orden en que se tomen los nodos cuando hay mas de uno con el mismo orden, hay algunos datos que se mantienen constantes.

Córdoba siempre va a estar pintada de Amarillo, por ser el único nodo de grado 7. Los nodos de grado 1 sólo podrán ser pintados de Amarillo (si su nodo adyacente es de color Rojo) o Rojo (si su adyacente es Amarillo). Y los únicos nodos que podrían ser pintados de Verde son aquellos que tienen grado 3 o superior.

2.3 El algoritmo

```
Coloreo (G)   {        (G es un grafo plano)
        Si hay algún nodo sin pintar {
                Sea n el nodo sin pintar de mayor grado
                Pintar (n)
                Coloreo (G)
        }
```

}

Donde *Pintar (n)* pinta al nodo *n* con el color de menor valor que no está asignado a ninguno de los nodos adyacentes a *n*.

Éste algoritmo se corrió con el mapa de Argentina mas de 20.000 veces sin presentar problemas.

3. Algoritmo 2 de eliminación de posibilidades (Incorporación de "grado de error")

Al intentar pintar el mapa de Argentina con el algoritmo dado no se presentan problemas. Pero hay que tener en cuenta que el mapa de Argentina es relativamente muy sencillo. Tiene pocos nodos y los grados de los mismos no son muy altos. Pero el utilizar éste algoritmo con un mapa un poco mas complicado, como es el mapa de África (tiene el doble de la cantidad de nodos y de mayor grado) se presentan algunos problemas[6].

3.1 Grado de error

En el caso del mapa de África, la efectividad del algoritmo descrito al ejecutarlo 20000 veces es cercana al 80%. Y se puede ver que los errores, frecuentemente, se dan en los nodos 9, 16 y 29 (Libia, Chad y Nigeria, respectivamente).

Al realizar pruebas de escritorio sobre los caminos que seguía el algoritmo para llegar a esos errores se descubrió que el problema estaba dado por los nodos adyacentes a los que generaban errores que eran pintados antes que éstos.

Tomemos como ejemplo el nodo 29, que es de grado 4. Sus nodos adyacentes son el 12 (de grado 7), el 16 (de grado 6), el 19 (de grado 4), y el 33 (de grado 6). Los nodos 12, 16 y 33 serán, inevitablemente, pintados antes que el 29 por tener mayor grado. Pero el nodo 19, al ser del mismo grado, podrá ser pintado o no antes que el nodo 29, dependiendo del orden en que los tome el algoritmo. Cuando el algoritmo toma primero al nodo 19 se genera el error, cuando pinta primero al 29 no se genera el error.

Análogos resultados se obtienen al estudiar los nodos 9 y 16.

Para evitar éste error habría que hacer que el nodo 29 sea pintado antes que el 19 sin depender del orden en que los tome el algoritmo. Así, planteo este nuevo algoritmo definiendo "el grado de error de un nodo" como el grado del nodo mas la cantidad de nodos adyacentes que tenga grado igual o superior a él.

Sabiendo que los nodos de grado inferior a 4 no pueden generar errores[7] , podemos definir un algoritmo para calcular el grado de error de un nodo de la siguiente manera:

Grado de error (G, n) {(G es un grafo y n un nodo del grafo G)

[6] Para más información sobre la estructura de datos utilizada y los grafos asociados a cada mapa dirigirse al apéndice A.

[7] Por lo visto en el apartado 1.2.

Si (Grado(n) < 4) devolver Grado (n)
Si no {
Sean k_i los nodos adyacentes a n
Sea x = Grado (n)
para (i = 1..Grado(n)) {
si (Grado (k_i) >= Grado (n)) x = x + 1
}
devolver x
}
}

Donde *Grado(n)* devuelve el grado del nodo *n*.

De ésta forma el grado de error del nodo 29 es 8 y el del nodo 19 es 7, con lo que si modificamos el algoritmo para que tome en primer lugar a los nodos que tienen mayor grado de error en lugar de los que tienen mayor grado resolvemos el problema que se nos había presentado al trabajar con el mapa de África.

3.2 El algoritmo

De esta manera, calculando el grado de error de cada nodo del grafo, planteamos el siguiente algoritmo:

Coloreo (G) { (G es un grafo plano)
Si hay algún nodo sin pintar {
Sea n el nodo sin pintar de mayor grado de error
Pintar (n)
Coloreo (G)
}
}

Éste algoritmo tiene una efectividad del 100% tanto en el mapa de Argentina como en el de África tras haber corrido mas de 2000 veces en cada uno.

4. Algoritmo 3 de eliminación de posibilidades (incrementar el grado de error)

Una vez mas intentamos aplicar éste algoritmo en un mapa de mayor complejidad. El mapa elegido en éste caso es el de España que cuenta con 51 nodos[8].

[8] El mapa de España tiene 37 nodos de grado 4 o superior, mientras que el de África tiene 27. El promedio de grado por nodo en el mapa de España es de 4,549 y en el de África

Al aplicar el algoritmo presentado en el apartado 3.2 sobre el mapa de España su efectividad se ve reducida al 38%, aproximadamente.

La modificación en éste nuevo intento consiste en aumentar el grado de error de un nodo cuando éste nodo genera un error. Para ello, se suma una unidad al grado de error del nodo que generó el conflicto, se despintan los nodos ya pintados y se comienza nuevamente el proceso.

4.1 El algoritmo

```
Coloreo (G)   {        (G es un grafo plano)
    Si hay algún nodo sin pintar {
        Sea n el nodo sin pintar de mayor grado de error
        Si hay algún color disponible para n Pintar (n)
        Si no {
            Incrementar el grado de error de n en 1
            Despintar (G)
        }
        Coloreo (G)
    }
}
```

Donde *Despintar (G)* despinta todos los nodos del grafo G.

5. Optimización del algoritmo

Al ver el último algoritmo planteado cabe hacer una pregunta. ¿Es realmente necesario despintar todos los nodos?. La verdad que el despintar todos los nodos implica que luego tendré que volver a pintarlos y ésta tarea implicaría trabajo extra.

Al analizar el funcionamiento del algoritmo podríamos concluir que la modificación del grado de error de un nodo sólo podría afectar a los nodos que le son adyacentes, que en definitiva son los que han utilizado los colores disponibles y, de ésta manera, agotado las posibilidades para el nodo que genera el error.

Por esto, una optimización para el algoritmo sería despintar solamente los nodos que son adyacentes al que generó el error y continuar trabajando.

5.1 El algoritmo[9]

```
Coloreo (G)      {        (G es un grafo plano)
    Si hay algún nodo sin pintar {
```

es de 4,255.

[9] Para ver éste algoritmo en funcionamiento puede dirigirse a http://www.angel-gris.com.ar/coloreo

```
                    Sea n el nodo sin pintar de mayor grado de error
                    Si hay algún color disponible para n Pintar (n)
                    Si no {
                        Incrementar el grado de error de n
                        Despintar los nodos adyacentes a n
                    }
                    Coloreo (G)
                }
            }
```

La eficiencia de este algoritmo se muestra en el punto siguiente.

6. Rendimiento de los algoritmos

Los rendimientos se toman tras haber corrido cada uno de los algoritmos 1500 veces con PHP en una computadora Intel Pentium II, 233 MHz con 128 Mb de memoria RAM.

	1er Algoritmo	2do Algoritmo	3er Algoritmo	3er Algoritmo Opt.
Argentina	**0,0223 seg.**	**0,0261 seg.**	**0,0259 seg.** **0 errores**	**0,0263 seg.** **0 errores**
África	**89,53 %**	**0,0817 seg.**	**0,0821 seg.** **0 errores**	**0,0826 seg.** **0 errores**
España	**3,2 %**	**36,4 %**	**0,1279 seg.** **1,126 errores**	**0,1108 seg.** **1,5053 errores**

En el cuadro se muestran los rendimientos de cada uno de los algoritmos aplicados a los 3 mapas utilizados para su desarrollo. Para los dos primeros algoritmos se especifica el tiempo promedio que demora cada vez que corre si su efectividad es del 100% (es decir, encuentra una solución cada vez que corre) y en caso contrario se muestra su porcentaje de efectividad. Para el tercer algoritmo y su optimización se muestra el tiempo promedio que demora en correr y el promedio de errores por cada solución encontrada.

7. Conclusiones

Tras haber desarrollado los tres algoritmos y optimizado el último de estos, podemos concluir que la complejidad que presenta un mapa a la hora de ser pintado con tan solo 4 colores sin que ninguna de sus regiones adyacentes tengan el mismo color no está dada por el grado de sus nodos, si no mas bien por el *grado de error* de los mismos.

El grafo asociado al mapa de Argentina tiene un promedio de grado de error de sus nodos de 4,88. El grafo asociado a África tiene un promedio de 5,66 y el de España es de 6,73. Con lo que podemos decir que la dificultad que presenta un mapa está dada por el promedio de grado de error por nodo que tiene su grafo asociado.

El problema fue resuelto con un algoritmo recursivo que tiene la facultad de modificar el valor del grado de error de un nodo en caso de ser necesario. Y dicho algoritmo fue optimizado obteniendo los resultados que se muestran en el punto 6 en el que se ve que el mapa de España presenta mayor cantidad de errores en el algoritmo optimizado, pero aún así el tiempo promedio que demora en encontrar una solución es menor comparado con el tiempo que demora el algoritmo sin optimizar.

Apéndice A

A.1 Estructura de datos

Al trabajar con grafos no dirigidos de *n* nodos lo que hacemos es crear una matriz *V* de *n* x *n* en la que el elemento v_{ij} vale 1 si los nodos *i* y *j* son adyacentes y vale 0 en caso contrario.

A.2 Mapa de Argentina

Fig. A.2.1 – Mapa de Argentina

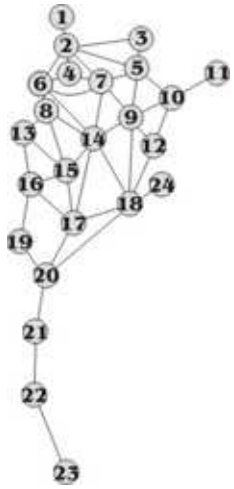

Fig. A.2.2 – Grafo asociado al mapa de Argentina

Nodo	Nombre	Grado	Grado de Error
1	Jujuy	1	1
2	Salta	6	7
3	Formosa	2	2
4	Tucumán	3	3
5	Chaco	5	8
6	Catamarca	5	8
7	Santiago del Estero	6	9
8	La Rioja	4	7
9	Santa Fe	6	9
10	Corrientes	4	6
11	Misiones	1	1
12	Entre Ríos	3	3
13	San Juan	3	3
14	Córdoba	7	7
15	San Luis	5	7
16	Mendoza	4	6
17	La Pampa	5	8
18	Buenos Aires	6	8
19	Neuquen	2	2
20	Río Negro	4	6
21	Chubut	2	2
22	Santa Cruz	2	2
23	Tierra del Fuego	1	1
24	Capital Federal	1	1

A.3 Mapa de África

Fig. A.3.1 – Mapa de África

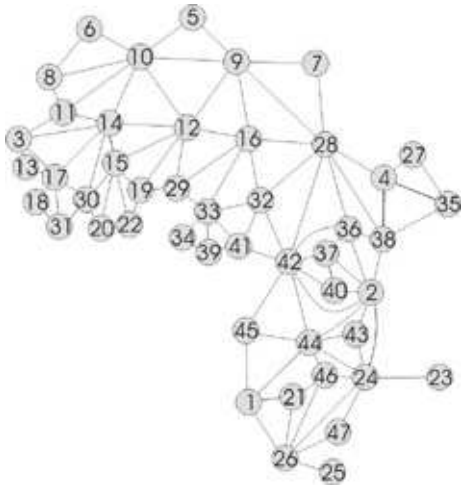

Fig. A.3.2 – Grafo asociado al mapa de África

Nodo	Nombre	Grado	Grado de Error
1	Namibia	4	6
2	Tanzania	8	9
3	Senegal	4	7
4	Etiopía	4	6
5	Túnez	2	2
6	Marruecos	2	2
7	Egipto	2	2
8	Sahara Español	3	3
9	Libia	6	10
10	Argelia	7	9
11	Mauritania	4	7
12	Níger	7	9
13	Puerto Guinea	2	2
14	Malí	7	9
15	Alto Volta	6	8
16	Chad	6	10
17	Guinea	6	7
18	Sierra Leona	2	2
19	Dahomey	4	7
20	Ghana	3	3
21	Botswana	3	3
22	Togo	3	3
23	Madagascar	1	1
24	Mozambique	7	9
25	Lesotho	1	1
26	Rep. Sudafricana	6	7
27	Somalia	2	2
28	Sudán	8	9
29	Nigeria	4	8
30	Costa de Marfil	5	8
31	Liberia	3	3
32	Rep. Centro Africana	5	9
33	Camerún	6	7
34	Guinea Ecuatorial	2	2
35	Rep. Somalí	3	3

36	Uganda	5	9
37	Ruanda	4	7
38	Kenia	5	8
39	Gabón	3	3
40	Burundi	3	3
41	Congo	4	7
42	Zaire	9	9
43	Malawi	3	3
44	Zambia	7	10
45	Angola	3	3
46	Rhodesia	4	7
47	Swazilandia	2	2

A.4 Mapa de España

Fig. A.4.1 – Mapa de España

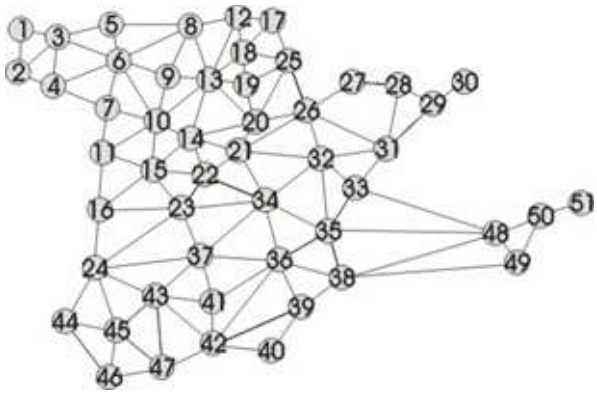

Fig. A.4.2 – Grafo asociado al mapa de España

Nodo	Nombre	Grado	Grado de Error
1	La Coruña	2	2
2	Pontevedra	3	3
3	Lugo	5	6
4	Orense	4	7
5	Asturias	3	3
6	León	7	8
7	Zamora	4	8
8	Cantabria	5	7
9	Palencia	4	8
10	Valladolid	7	9
11	Salamanca	4	8
12	Vizcaya	4	7
13	Burgos	8	8
14	Segovia	6	11
15	Ávila	6	9
16	Cáceres	4	8
17	Guipúzcoa	3	3
18	Álava	5	7
19	La Rioja	4	8
20	Soria	6	10
21	Guadalajara	6	11
22	Madrid	5	10
23	Toledo	6	10
24	Badajoz	6	9
25	Navarra	5	8

26	Zaragoza	6	9
27	Huesca	2	2
28	Lérida	3	3
29	Barcelona	3	3
30	Gerona	1	1
31	Tarragona	5	7
32	Teruel	6	10
33	Castellón	4	8
34	Cuenca	7	8
35	Valencia	6	9
36	Albacete	7	8
37	Ciudad Real	6	11
38	Alicante	5	8
39	Murcia	4	7
40	Almería	2	2
41	Jaén	4	8
42	Granada	6	8
43	Córdoba	6	9
44	Huelva	3	3
45	Sevilla	5	7
46	Cádiz	3	3
47	Málaga	4	7
48	Ibiza	5	7
49	Formentera	3	3
50	Mallorca	3	3
51	Menorca	1	1

Bibliografía

- "Algoritmos, estructura de datos y matemática para informática"- Dr. Thomas Hibbard, Apuntes de cátedra, Universidad Nacional de Salta, 2002.
- "Coloreo del mapa de la República Argentina" - Mónica Flores y Jimena Ríos, trabajo de cátedra, Universidad Nacional de Salta 2003.
- "Algoritmos en C++" – Robert Sedgewick – Edit. Addison-Wesley/Diaz Santos - 1996
- "Estructuras de Matemáticas Discretas para la Computación". Kolman, Busby, Ross - Ed. Prentice Hall. Año 1997.
- "Matemáticas Discretas". Richard Johnsonbaugh - Ed. Prentice Hall. Año 1999.
- "Matemáticas Discretas y Combinatoria". Ralph Grimaldi - Ed. Addison Wesley. Año 1997.
- "Iterative improvement and local search", http://www.cs.toronto.edu/~mitchell/ai-course/s4.html
- "The four colour theorem", J.J. O'Connor y E.F. Robertson, http://www-gap.dcs.st-and.ac.uk/~history/HistTopics/The_four_colour_theorem.html