UNIVERSIDAD AUTÓNOMA METROPOLITANA – UNIDAD AZCAPOTZALCO
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA
INGENIERÍA EN COMPUTACIÓN

Algoritmo evolutivo para el problema de coloración difusa

Proyecto terminal para obtener el grado de Ingeniero en Computación presentado por:

Alberto Alejandro Vázquez Cortés
bajo la direción del
Dr. Pedro Lara Velázquez

16/12/2009

Agradecimientos

Este logro tan importante se lo dedico a mis abuelitos Leonida y Jocundo, las personas más buenas y maravillosas que he conocido; en especial a mi abuelito, ya que el siempre me inspiro un ejemplo a seguir, un deseo por aprender, un ejemplo de vida y las ganas de dedicarme de lleno a mis convicciones.

En especial quiero agradecer a mis padres por forjar la persona que soy; para ellos mi educación siempre fue lo más importante a pesar de la adversidad, sin su apoyo y amor hubiera sido imposible alcanzar mis metas. Gracias por siempre creer en mí, así como mis hermanos a quienes también les dedico este triunfo.

Quiero agradecer el apoyo de mi familia, el que siempre estuvieran pendiente de este proceso y apoyándome incondicionalmente; por ser esa familia unida que demuestra que tenemos un gran legado y estoy seguro que “ellos” están muy orgullosos de todos nosotros.

A mis compañeros de escuela y amigos, que sin su compañerismo, lealtad y sobre todo el espíritu mutuo de ser siempre los mejores, este recorrido de vida hubiera sido más difícil.

De la misma forma quiero reiterar mi lealtad y orgullo a la casa de estudios que me forjó, la Universidad Autónoma Metropolitana. Esta casa abierta al tiempo que fue mi segundo hogar durante cuatro años, en donde no solo estudie y me forje como Ingeniero, sino también comí, reí, dormí, crecí y me dejo la mejor experiencia de mi vida. Gracias a todas las personas que hacen grande esta institución, pero en especial a su cuerpo académico, maestros comprometidos con el alumnado y con la educación, de que quienes siempre recibí tanto la mejor educación como experiencias de vida que estoy seguro me marcarán como profesionista.

Entre el cuerpo académico quiero destacar al Dr. Pedro Lara Velázquez, que mas que ser mi profesor de 2 cursos y ser mi asesor de Proyecto Terminal, fue un amigo y creo que el realizar este trabajo fue una experiencia enriquecedora y divertida. De igual forma quiero agradecer a la Dra. Silvia B. González Brambila todo lo que hace por los alumnos y la carrera; porque siempre se preocupo que tuviéramos lo mejor y que al egresar estuviéramos muy preparados. Podría mencionar a todos los profesores que con su cátedra cada día aprendí lo mejor de ellos, pero me llevaría otro Proyecto Terminal, así que solo destacare la contribución del Dr. Francisco Zaragoza Martínez, gran profesor y sin duda, uno de los mejores de la universidad.

Resumen

El objetivo de este documento es presentar un algoritmo heurístico – evolutivo que resuelva el problema de coloración difusa, que representa un problema de optimización combinatoria.

En el problema de coloración difusa se considera un grafo que en lugar de tener asociada una matriz de adyacencia con aristas bien definidas (0,1), se definen de una forma difusa (la matriz tendrá números fraccionarios entre 0 y 1 (0.1416, 0.9326); conservando la misma propiedad de simetría. Gracias a esto, los valores de muchos parámetros, por ejemplo densidad y número cromático, dejan de ser números y se convierten en variables aleatorias.

Dados un grafo $G= (V, E)$ y el conjunto finito de colores {1,2,…,c}, en una coloración difusa cada color k , $k \in \{1,2,\ldots,c\}$ está presente en la coloración de un vértice $i \in V$ con cierta intensidad. Por ejemplo, si c = 2, considerando los colores *0* (blanco) y *1* (negro) se obtienen las tonalidades de grises.

El algoritmo que se describe fue dividió en tres bloques principales: Generación de instancias aleatorias, Difuminado del grafo y el de Soluciones nítidas parciales; siendo el conjunto soluciones nítidas asociado al corte dado por el difuminado del grafo el Número cromático difuso.

Para encontrar la coloración mínima de un los grafos nítidos, se partió del algoritmo de Recocido Simulado; además se hace todo un análisis del proceso de Recocido Simulado y se explica las decisiones que se toman para que encontrar una buena solución.

Índice

1.Grafos.....5

1.2 Grafos no dirigidos.....6

1.3 El problema de coloración de grafos.....7

2. Enfoque difuso del problema de coloración de grafos.....8

2.1 Introducción.....8

2.2 Coloración difusa de un grafo nítido.....8

2.3 Problema de coloración mínima difusa.....9

2.4 Número cromático difuso.....11

3. Método para resolver el problema.....15

3.1 Introducción.....15

3.2 Introducción al Recocido Simulado.....16

3.3 El proceso de Recocido Simulado.....16

4. El algoritmo del problema de coloración difusa.....19

4.1 Diagrama de bloques.....19

4.2 Generación de instancias aleatorias.....20

4.3 Difuminado del grafo.....21

4.4 Soluciones nítidas parciales.....22

4.4.1 Algoritmo de Recocido Simulado.....22

4.4.2 Algoritmo glotón.....25

4.5 Número cromático difuso.....26

4.6 Presentación de Resultados.....27

1. Grafos

En matemáticas y ciencias de la computación, un grafo (del griego *grafos*: dibujo, imagen) es el principal objeto de estudio de la teoría de grafos.

Informalmente, un grafo es un conjunto de objetos llamados vértices unidos por enlaces llamados aristas, que permiten representar relaciones binarias entre elementos de un conjunto.

Desde un punto de vista práctico, los grafos permiten estudiar las interrelaciones entre unidades que interactúan unas con otras. Prácticamente cualquier problema puede representarse mediante un grafo, y su estudio trasciende a las diversas áreas de las ciencias exactas y las ciencias sociales.

Por ejemplo, en 1736 el primer artículo relativo a grafos escrito por el matemático Leonhard Euler demostró en base al grafo asociado al esquema de puentes de Königsberg (representado en la figura 1.1) que no es posible regresar al vértice de partida sin pasar por alguna arista dos veces. Este es solo un ejemplo de una abstracción de un problema del mundo real a la teoría de grafos.

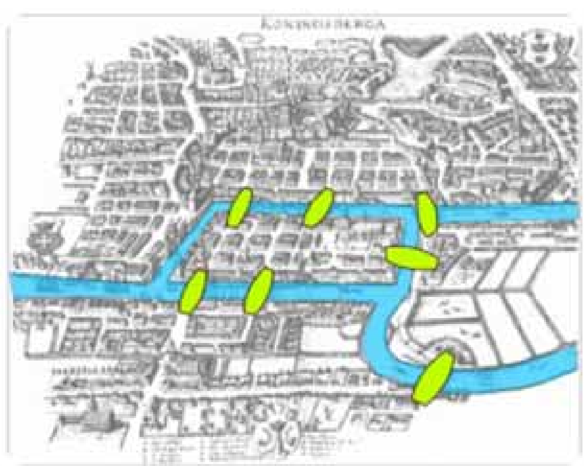


Figura 1.1. Los siete puentes de Königsberg

1.1 Grafos no dirigidos

En el presente proyecto se trabajó con grafos no dirigidos debido a que las interrelaciones entre los vértices corresponden solamente a un estado de compatibilidad y no tiene sentido un comportamiento dirigido. En la figura 1.2 podemos notar la diferencia de una manera gráfica.

Figura 1.2. Diferencia entre grafo dirigido y no dirigido

Un **grafo no dirigido** podemos definirlo como una pareja ordenada *G= (V, E)* donde *V* es un conjunto finito de vértices y *E* es un conjunto finito de aristas. Una arista puede definirse como un par no ordenado de vértices *E= {u, v}* donde *u*, *v* ∈ *V*. Se dice que dos vértices *u*, *v* son vecinos si el par *{u, v}* ∈ *E*. Se define como el grado de un vértice al número de vecinos que este tenga en el grafo.

A continuación se explicará de una manera más intuitiva el concepto de grafo y se introduce el concepto de matriz de adyacencia para representar cuando un vértice es vecino de otro, es decir tienen asociada una arista.

Dado el presente grafo:

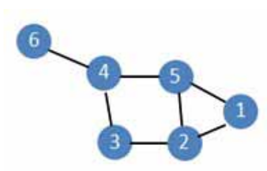


Figura 1.3. Grafo

|V| = 6
V = {1, 2, 3, 4, 5, 6}
|E| = 7
E = { {1,2}, {1,5}, {2,3}, {2,5}, {3,4}, {4,5}, {4,6} }

La **matriz de adyacencia** es una matriz cuadrada que se utiliza como una forma de representar relaciones binarias. Cada grafo tiene asociado su matriz de adyacencia.
En este caso el número uno significa que hay una arista uniendo dos vértices, en otras palabras que son vecinos.
La representación de matriz de adyacencia del grafo de la figura 1.3 es la siguiente:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 1.4. Matriz de Adyacencia

1.2 El problema de coloración de grafos

Este problema consiste en asignar un color a cada vértice de forma que ninguna arista tenga el mismo color en sus dos extremos. Se define como una *k*-coloración a la coloración de un grafo que utiliza a lo mucho *k*-colores. Generalmente, se busca obtener la coloración mínima de un grafo, es decir, **minimizar** el número de colores utilizados.

Algunos problemas de planificación del tiempo o de recursos se pueden plantear como problemas coloración de los vértices de un grafo; los vértices representan los elementos a programar y hay una arista entre dos vértices cuando los elementos representados por éstos son incompatibles; una coloración válida identifica una asignación de recursos compatibles. Si el objetivo es usar el mínimo número de colores se tiene planteado el problema de coloración mínima.

Por ejemplo, el grafo de la figura 1.5, tiene una *4 – coloración*, en donde también coincide que es una coloración mínima.

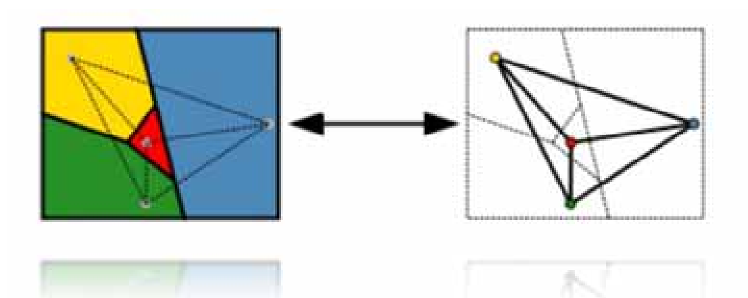


Figura 1.5. Grafo coloreado |V| = 4, k = 4

2. Enfoque difuso del problema de coloración de grafos

2.1 Introducción

A partir del concepto de número difuso, y considerando que una función de coloración asigna números (colores) a los vértices de un grafo, es natural plantearse qué ocurre cuando el color asignado es un número difuso, introduciendo así la **coloración difusa de un grafo nítido**. También se puede uno plantear qué ocurre cuando la función de coloración es nítida pero el grafo es difuso; a partir de esta idea se plantea el problema de **coloración mínima difusa** y se introduce el concepto de **número cromático difuso**.

2.2 Coloración difusa de un grafo nítido (Javier Ramírez Rodríguez, 2000[1])

Una de las formas en que se puede representar un conjunto D es usando el concepto el concepto de **función característica μ_D (v)** cuyo valor uno o cero indica si el elemento v pertenece o no al conjunto D.

Sean los conjuntos $V = \{v_1,v_2,v_3,v_4,v_5\}$ y $D = \{v_1,v_3,v_5\}$; entonces se puede escribir $\mu_D(v_1)=1$, $\mu_D(v_2)=0$, $\mu_D(v_3)=1$, $\mu_D(v_4)=0$, $\mu_D(v_5)=1$, lo que permite describir al conjunto D con los elementos de V acompañados del valor de la función característica:

$$D = \{(v_1,1),(v_2,0),(v_3,1),(v_4,0),(v_5,1)\}$$

Los elementos con grado de pertenencia igual a cero normalmente no se escriben.

Al considerar que la función característica puede tomar cualquier valor en el intervalo [0,1], en 1965 Zadeh introdujo el siguiente concepto de conjunto difuso, que generaliza al concepto clásico de conjunto.

Definición 1. (Zadeh[2]) Sea V un conjunto finito o infinito de elementos denotados genéricamente por v, entonces un subconjunto difuso D de V es el conjunto de parejas $\{v, \mu_D(v)\}$, $\forall v \in V$ donde $\mu_D(v)$ es el grado de pertenencia de v a D. Se puede decir que $\mu_D(v)$ toma sus valores de un conjunto $\boldsymbol{M}$ llamado **conjunto de pertenecía**, con la función $\mu_D : V \rightarrow M$. Esta función se llama función de pertenecía.

Si $M = \{0,1\}$, entonces el conjunto difuso D se vuelve un subconjunto no difuso o un subconjunto común, en este proyecto se considera $M = [0,1]$.

Considerando que la función de coloración de un grafo asigna colores (números) a cada vértice, y que esta coloración implica una determinada asignación, es natural plantearse la cuestión de generalizar esta asignación permitiendo que el color sea un número difuso. La **coloración difusa** $\mathfrak{E}$ así introducida debe asignar a cada vértice del grafo una graduación de cada uno de los colores seleccionados de un conjunto fijado previamente. Formalmente se introduce la siguiente definición:

Definición 2. Dados un grafo *G= (V, E)* y el conjunto finito de colores {1,2,…,c}, una coloración difusa es una aplicación

$$\mathfrak{E}^c : V \rightarrow (\mu_{\mathfrak{E}^c}(1),\dots, \mu_{\mathfrak{E}^c}(c))$$

Es decir, cada color ***k***, $k \in \{1,\dots,c\}$ está presente en la coloración de un vértice $i \in V$ con cierta intensidad $\mu_{\mathfrak{E}^c}(k)$. Por ejemplo, si $c = 2$, considerando los colores *0* (blanco) y *1* (negro) se obtienen las tonalidades de grises.

2.3 Problema de coloración mínima difusa (Javier Ramírez Rodríguez, 2000[1])

Definición 3. (Rosenfeld[3]) Sea $V = \{1,...,n\}$, un grafo $\hat{G} = (V,\sigma,\mu)$ es un grafo difuso, donde

1. $\sigma: V \rightarrow [0, 1]$, que indica el nivel de pertenencia de cada vértice.
2. $\mu: V \times V \rightarrow [0, 1]$, que indica el nivel de pertenencia de cada arista.

Dados $x, y \in V$. Si $\mu(x,y) = 0$ entonces no existe la arista {x,y} y si $\mu(x,y) > 0$ entonces

existe la arista difusa {x,y} , siendo $\mu(x,y)$ el grado de pertenencia de {x,y} al grafo $\hat{G}$.

Los grafos difusos considerados verifican que $\sigma(v)=1$, $\forall v \in V$. Estos grafos son llamados **grafos de aristas difusas** que se denota por $\hat{G} = (V, \mu)$; donde la imagen de μ es un conjunto totalmente ordenado, no necesariamente en el intervalo [0,1].

Definición 4. Sea el grafo de aristas difusas $\hat{G} = (V, \mu)$, para cada α que sea elemento de un conjunto totalmente ordenado, se define el **α – corte de $\hat{G}$** como el **grafo nítido** $G_\alpha = (V,E_\alpha)$ donde $E_\alpha = \{(i,j) \in E \mid \mu(i,j) \geq \alpha\}$. Ya que los grafos a tratar tienen un conjunto finito de aristas, tendrán un número finito de α – cortes diferentes.

A continuación se presenta un ejemplo y posteriormente se definirá el grafo difuso asociado.

Ejemplo 1. Sea el cruce de cuatro calles ilustrado en la figura 2.1. Las flechas indican los sentidos posibles de circulación de los automóviles.
Se trata de regular el tráfico del cruce mediante semáforos que indiquen a los conductores cuando pueden pasar. Hay que identificar el ciclo mínimo del semáforo que evite los choques, indicando en cada uno de los periodos de ese ciclo cuáles son los movimientos permitidos.

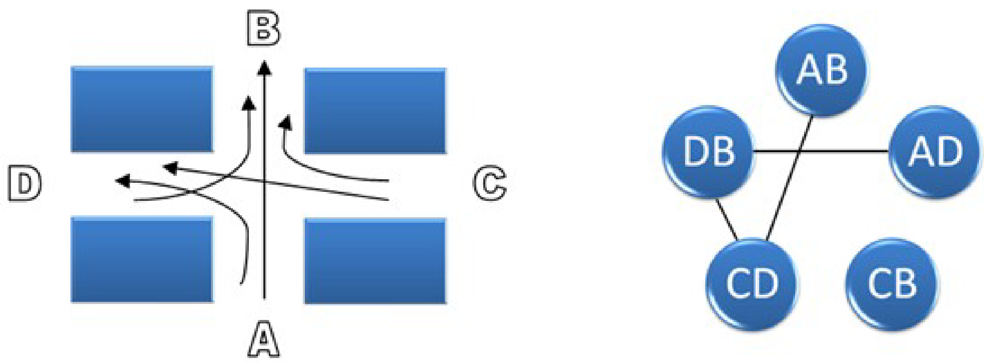


Figura 2.1. Cruce de calles y su grafo de incompatibilidades

Como se puede observar los vértices del grafo son cada uno de los sentidos de la circulación posibles y habrá una arista entre dos vértices cuando sean incompatibles, es decir, cuando en los sentidos de circulación correspondientes haya riesgo de colisión al permitir el paso a ambos sentidos por tener el semáforo en verde simultáneamente. El **grafo de incompatibilidades** *G* es el que se muestra en la figura 2.1.

Una coloración del grafo, indicará con cada clase de color los trayectos que se pueden realizar simultáneamente y con esto, qué semáforos deben tener luz verde y cuáles luz roja.

Una coloración de *G* es la siguiente:

$$C(AB) = 1,\ C(AD) = 2,\ C(CB) = 1 \text{ ó } 2,\ C(CD) = 2 \text{ y } C(DB) = 1$$

2.4 Número cromático difuso (Javier Ramírez Rodríguez, 2000[1])

El ciclo del semáforo en el ejemplo *1* depende de la definición de incompatibilidad de los sentidos de circulación. Este concepto admite **graduaciones**. Está claro que si en este ejemplo se consideran todos los giros incompatibles, el grafo de incompatibilidades es un grafo completo, por lo que se requieren *5* colores para pintarlo; lo que indica que debería haber en el semáforo un periodo de color verde para cada uno de los sentidos de circulación posibles, que no es una opción práctica. En el extremo opuesto, si todos los giros son compatibles, el grafo de incompatibilidades no tendría aristas y se pueden pintar con un solo color todos los vértices, pero esta tampoco es una opción.

Situaciones intermedias se obtienen cuando el grafo se hace más restrictivo, por ejemplo limitando cada periodo de luz verde en un semáforo a los sentidos de circulación con menos riesgo. Para esto las incompatibilidades se clasifican según su grado en ***n**ula*, ***b**aja*, ***m**edia* y ***a**lta*, como se puede observar en la figura 2.2.

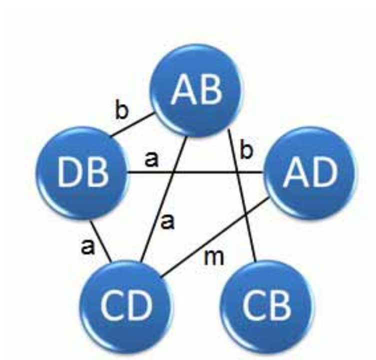


Figura 2.2. Grafo difuso

De lo anterior, el problema puede modelarse como un grafo con aristas difusas $\hat{G} = (V, \mu)$ donde $\mu : V \times V \rightarrow$ {*nulo, bajo, medio, alto*} y $\mu(i,j)$ indica el grado en que el sentido de la circulación *i* es incompatible con el sentido de la circulación *j.*

Ahora se plateará el problema de coloración del ejemplo del grafo difuso, así de esta manera usaremos la representación de α – cortes para obtener un **conjunto de problemas nítidos** que dependen del grado de pertenencia; al resolver estos problemas se obtienen soluciones que dependen de dicho grado y se llaman **soluciones difusas** del problema.

Definición 5. Existe una secuencia $\hat{A} = \{\alpha_1,\ldots, \alpha_n\}$ de niveles de pertenencia y el conjunto de soluciones $S = \{s_1,\ldots,s_n\}$, tales que s_1 es solución de $G_{\alpha 1}$ y es óptima en $J_1 = (0, \alpha_1]$, s_2 es solución de $G_{\alpha 2}$ y es óptima en $J_2 = (\alpha_1, \alpha_2]$, … , s_n es solución de $G_{\alpha n}$ y es óptima en $J_n = (\alpha_{n-1}, \alpha_n]$.

Una solución difusa se puede considerar como el conjunto difuso

$$\hat{S} = \{(s_1, \alpha_1), \ldots , (s_n, \alpha_n)\}$$

Con esta definición cada s_i de los elementos de S será una solución nítida que es óptima para niveles de pertenencia mayores o iguales a α_i con $i \in \{1, \ldots ,n\}$.

Continuando con el ejemplo *1*, si además de las aristas que son incompatibles se consideran:

1. El grafo de *incompatibilidad alta* G_{alta} es el grafo nítido representado en la figura 2.1, que anteriormente ya se había presentado y que su número cromático fue de 2. Cabe mencionar que los grados de incompatibilidades ya fueron definidos en el grafo difuso de la figura 2.2.

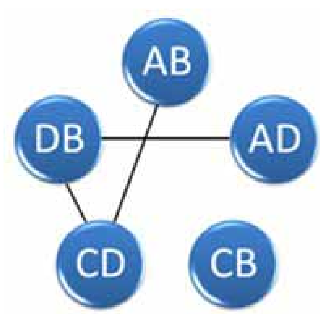


Figura 2.3. G_{alta}

2. Al añadir al grafo anterior las aristas que tienen *incompatibilidad media*, se define el grafo G_{media} , que se representa en la figura 2.4. Este grafo tiene número cromático 3; una *3 – coloración* es la siguiente:

$C(AB)$ = 1 ó 3, $C(AD)$ = 3, $C(CB)$ = 1,2 ó 3, $C(CD)$ = 2 y $C(DB)$ = 1

Se observa que con esta coloración se incrementa a *tres* el número de periodos de color verde en ciclo del semáforo.

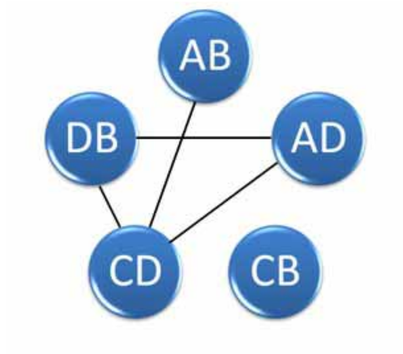


Figura 2.4. G_{media}

3. Al añadir al grafo de G_{media} las aristas que tengan *incompatibilidad baja*, es decir las aristas {*AB, CB*} y {*AB, DB*} , se define el grafo G_{baja} *,* representado en la figura 2.5. G_{baja} tiene número cromático 3 y una *3 – coloración* es la siguiente:

$$C(AB) = 1 \,,\ C(AD) = 1,\ C(CB) = 2 \text{ ó } 3,\ C(CD) = 2 \text{ y } C(DB) = 3$$

En este caso, el número de periodos de color verde en el ciclo del semáforo sigue siendo *tres* ; el tiempo que dure cada periodo dependerá, entre otros factores, del volumen de tráfico en cada sentido de circulación.

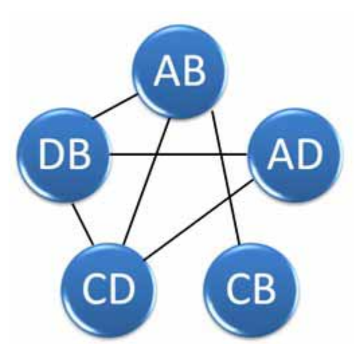


Figura 2.5. G_{media}

La ventaja de los casos G_{media} y G_{baja} sobre G, al que se designará por G_{alta}, es que disminuyen el riesgo de accidentes, pero también tienen el inconveniente de hacer más lento el tráfico.

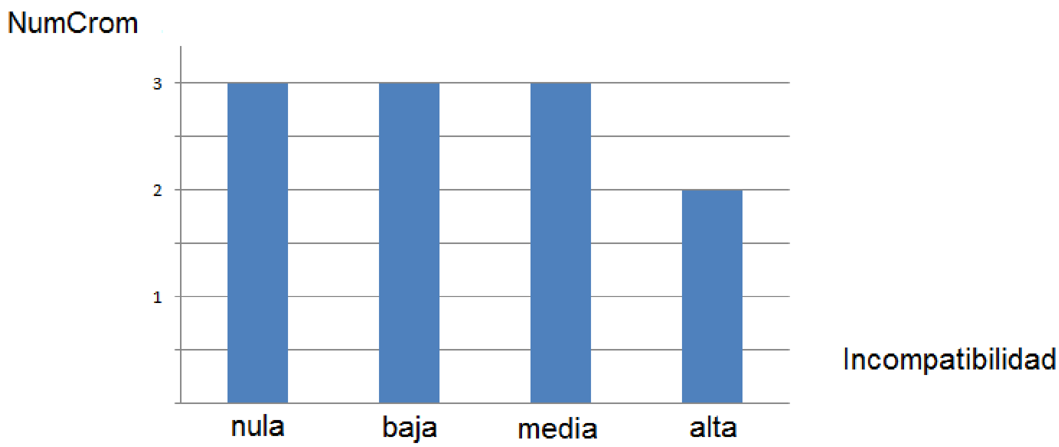


Figura 2.6. Gráfico de Número Cromático Difuso

La solución difusa es $\hat{S}$ = {(*3, baja*), (*2, alta*)} y para cada elemento proporciona el **número cromático** que se obtiene en el α correspondiente y ese valor de α es el mayor con el que se puede obtener dicho número cromático. En otras palabras basándonos en el ejemplo con el que hemos trabajado, cada elemento proporciona el número de periodos de luz verde en el semáforo y el mayor nivel de riesgo de colisiones que se corre con dichos periodos; a mayor incompatibilidad le corresponde mayor riesgo. La solución se puede mostrar en la figura 2.6.

A partir del esquema general de optimización y de las ilustraciones mostradas anteriormente se presenta la siguiente definición.

Definición 6. Dado un grafo con aristas difusas $\hat{G} = (V, \mu)$ el número cromático difuso de $\hat{G}$ estará formado por el conjunto difuso no normalizado cuyos elementos son las parejas formadas por el número cromático del α_i - *corte* $\hat{G}_{\alpha i}$ y α_i, con i que toma valores en $\{1, \ldots, n\}$.

No es difícil demostrar que el número cromático difuso es un conjunto convexo no-normalizado cuyo valor modal está asociado con el grafo más restrictivo.

3. Método para resolver el problema

3.1 Introducción

En el lenguaje coloquial (Rafael Martí [4]), optimizar significa poco más que mejorar; sin embargo, en el contexto científico la optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema. En un problema de optimización existen diferentes soluciones y un criterio para discriminar entre ellas. De forma más precisa, estos problemas se pueden expresar como encontrar el valor de unas variables de decisión para los que una determinada función objetivo alcanza su valor máximo o mínimo.

Para el problema de coloración difusa se considero un método heurístico – evolutivo como es el algoritmo realizado en este trabajo basado en Recocido Simulado, ya que se considera como un problema "difícil de resolver"; en otras palabras es un problema de optimización difícil, ya que no se puede garantizar el encontrar la mejor solución posible en un tiempo razonable.

Los algoritmos evolutivos están basados en poblaciones de soluciones. En cada iteración del algoritmo no se tiene una única solución sino un conjunto de éstas. Estos algoritmos se basan en generar, seleccionar, combinar y reemplazar un conjunto de soluciones. Dado que mantienen y manipulan un conjunto en lugar de una única solución a lo largo de todo el proceso de búsqueda, suelen presentar tiempos de ejecución sensiblemente más altos que otros metaheurísticos.

El recocido simulado es una de las metaheurísticas más clásicas. Su simplicidad y buenos resultados en numerosos problemas, lo han convertido en una herramienta muy popular, con cientos de aplicaciones en los más variados campos, entre ellos la optimización combinatoria. En este proyecto haremos una revisión de los fundamentos del método, analizando las diferentes decisiones que se tomaron en la fase del diseño del algoritmo aplicado al problema de coloración difusa.

3.2 Introducción al Recocido Simulado (Belarmino Adenso Díaz et al, 2001[5])

Desde que Kirkpatrick et al.[6] (1983) introdujeron el concepto de Recocido Simulado, RS, (“Simulated Annealing”, SA, en inglés), está metaheurística ha demostrado ser una herramienta muy exitosa para resolver una amplia gama de problemas de optimización combinatoria. El recocido Simulado es una variante de la **búsqueda local** que permite movimientos ascendentes para evitar quedar atrapado prematuramente en un óptimo local. El nombre viene de la idea en que está basado un algoritmo diseñado en los años 50 para simular el enfriamiento de un material (un proceso denominado “recocido”).

A menudo se dice que mientras que es muy fácil hacer que RS funcione, es difícil hacer que funcione bien. Esto debido a que no es propiamente un algoritmo, sino una estrategia heurística que necesita de varias decisiones para que quede totalmente diseñado, las cuales tienen una gran influencia en la calidad de las soluciones generadas. Estas decisiones son clasificadas en genéricas y específicas.

3.3 El proceso de Recocido Simulado (Belarmino Adenso Díaz et al, 2001[5])

El algoritmo de búsqueda local parte de una solución inicial que de modo paulatino se transforma en otras que a su vez son mejoradas al introducirles pequeñas perturbaciones (como cambiar el valor de una variable). Si el cambio da lugar a una solución “mejor” que la actual, se sustituye por la nueva, continuando el proceso hasta que no es posible ninguna nueva mejora. Esto significa que la búsqueda finaliza en un óptimo local, que no tiene que ser forzosamente el global.

Un modo de evitar este problema es permitir que algunos movimientos sean hacia soluciones peores. Pero si la búsqueda está yendo hacía una buena solución, estos movimientos de escape debe realizarse de un modo controlado, esto mediante una **función de probabilidad** que disminuirá la probabilidad de esos movimientos.

Los fundamentos de este control se basan en conceptos de Metropolis et al.[7] (1953) en el campo de la termodinámica estadística. Básicamente Metropolis modelo el proceso de RS simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable (congelado). Las leyes de la termodinámica dicen que a una temperatura t la probabilidad de un incremento energético de magnitud δE se puede aproximar por $P[\delta E] = exp(-\delta E/kt)$; siendo k una constante física denominada de Boltzmann que ignoraremos ya que no tiene significado en problemas de optimización. En el algoritmo de Metropolis se genera una perturbación aleatoria en el sistema y se calculan los cambios de energía resultantes: si hay una caída energética, el cambio se acepta automáticamente; por el contrario, si se produce un incremento energético, el cambio será aceptado con una probabilidad dada por $P[\delta E] = exp(-\delta E/t)$. El proceso se repite durante un número predefinido de iteraciones en series decrecientes de temperatura, hasta que el sistema este "frío".

La siguiente tabla muestra una relación entre los elementos de la simulación termodinámica y la optimización combinatoria.

Simulación termodinámica	Optimización combinatoria
Estados del sistema	Soluciones factibles
Energía	Costo
Cambio de estado	Solución en el entorno
Temperatura	Parámetro de control
Estado congelado	Solución heurística

Tabla 3.1 Relación establecida entre simulación termodinámica y optimización combinatoria

El parámetro t es un parámetro de control denominado generalmente *temperatura.* Si se permite que t alcance valores suficientemente pequeños, ya no habrá movimientos a peores soluciones y la convergencia será a un óptimo local.

A continuación se presenta el algoritmo básico de Recocido Simulado.

```
Sea f(s) el costo de la solución s y sea N(s) su entorno.
Selecciona una solución inicial s0.
Seleccionar una temperatura t0 > 0.
Seleccionar una función de reducción de la temperatura α.
Seleccionar un número de iteraciones nrep.
Seleccionar un criterio_de_parada.
REPETIR
   REPETIR
      Seleccionar aleatoriamente una solución s ∈ N(s0);
      Sea δ = f(s) - f(s0);
      SI δ < 0 ENTONCES s0 = s;
      SINO
         Generar aleatoriamente u ∈ U (0,1);
         SI u < exp (-δ/t) ENTONCES s0 = s;
      FINSINO
   HASTAQUE cuenta_iteraciones = nrep
   t = α(t);
HASTAQUE criterio_de_parada = CIERTO
```

Figura 3.1 Pseudocódigo del algoritmo básico de RS

Cabe mencionar que cualquier implementación de búsqueda local puede convertirse en una implementación RS.

Para poder implementar el algoritmo de la figura 3.1, aplicado al problema de coloración difusa, es preciso tomar una serie de decisiones, que las dividiremos en genéricas y específicas.

Las **decisiones genéricas** se refieren principalmente a cómo controlar la temperatura (incluyendo la definición de su valor inicial t_0 y la función de decrecimiento α, el número de iteraciones *nrep* antes del decrecimiento de la temperatura, y las condiciones que nos permitan considerar que el sistema ya está "frío".

Las **decisiones específicas** comprenden la definición del espacio de soluciones y la estructura de entornos, la función costo, y como se obtendrá la solución inicial s_0.

Ambos tipos de decisiones se deberán tomar con cuidado pues, como se comentó anteriormente, ejercen una gran influencia sobre la calidad de las soluciones. En el siguiente bloque se presentará el algoritmo de RS adecuado al problema de coloración difusa y se presentará la explicación de este, así como el porqué de la toma de decisiones tanto genéricas como específicas.

4. El algoritmo del problema de coloración difusa

A continuación se presenta el algoritmo dividido en los siguientes bloques:

1. Generación de instancias aleatorias
2. Difuminado del grafo
3. Soluciones nítidas parciales
4. Número cromático difuso

La representación de resultados se hará una vez que el algoritmo se haya ejecutado ya con el conjunto de valores que corresponden al número cromático de cada corte (soluciones nítidas parciales) y que en conjunto integran el número cromático difuso del grafo de aristas difusas generado por el bloque de generación de instancias aleatorias. Esta representación se hará graficando los valores obtenidos (de forma manual), para así observar el comportamiento desde un punto de vista estadístico.

4.1 Diagrama de bloques

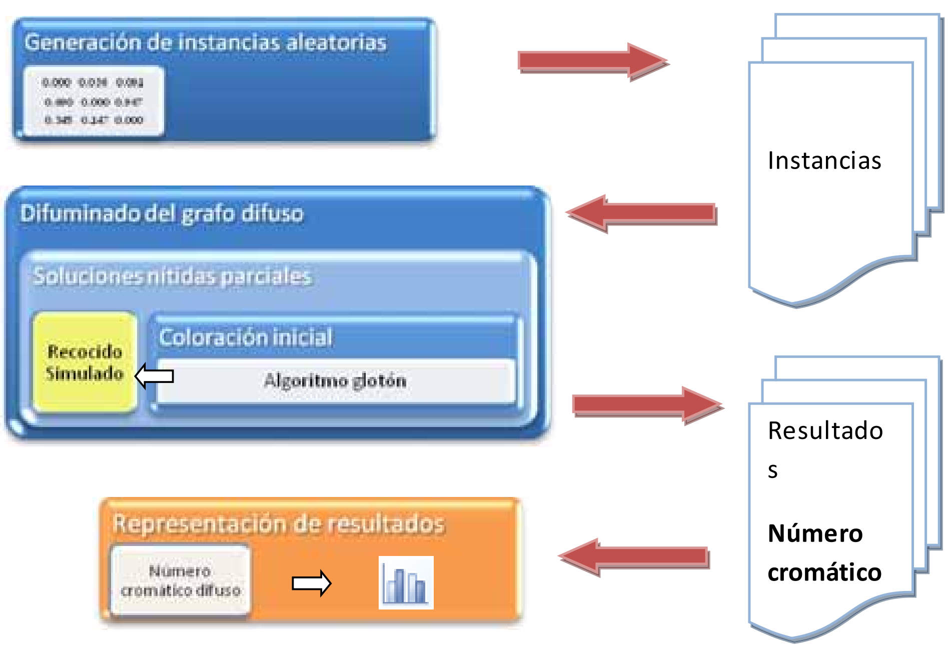


Figura 4.1 Diagrama de bloques del algoritmo que resuelve el problema de coloración difusa

4.2 Generación de instancias aleatorias

Figura 4.2. Sección Generación de instancias aleatorias

Antes de ejecutar el algoritmo se deberán asignar valores a un conjunto de constantes, que influenciarán en la exactitud del algoritmo, tiempo de ejecución, conjunto de soluciones y tamaño del problema.

La constante que juega un papel muy importante en este bloque es la del tamaño del grafo "***TAM***" y de ella dependerá el número de vértices de nuestro grafo difuso. Cabe mencionar que entre más vértices contenga el grafo mayor será el tiempo el tiempo de ejecución ya que la función de costos tendrá que analizar un conjunto mayor.

En este bloque se generara el grafo difuso con valores aleatorios dentro del rango [0, 1] con una precisión de 4 dígitos (ejemplo 0.1259), la representación es una matriz de adyacencia simétrica y cuadrada, la cual se escribe en el archivo cuyo nombre está contenido en la constante "***ARCHIVO***", en donde el primer valor corresponde al número de vértices seguido de la matriz de adyacencia.

A continuación se presenta el algoritmo:

Figura 4.3. Pseudocódigo de la función que genera las instancias aleatorias

```
FUNCION generarInstanciasAleatorias ()
COMIENZO
   indice ← 0;
   arreglo [TAM] [TAM];
   Abrir ARCHIVO en modo de escritura;
   PARA  i  ← 0  HASTA TAM -1
        PARA j  ← 0  HASTA  TAM -1
             SI  j < indice
                  r ←  valor aleatorio, r ∈ [0,1];
                  arreglo [i][j] ← r;
                  arreglo [j][i] ← r;
             FINSI
             SINO
                  arreglo [i][j] ← 0;
             FINSINO
       FINPARA
       indice ← indice + 1;
   FINPARA
   escribir en ARCHIVO TAM;
   PARACADA  x  ∈  arreglo
        escribir en ARCHIVO  x;
   FINPARACADA
TERMINAR
```

4.3 Difuminado del grafo

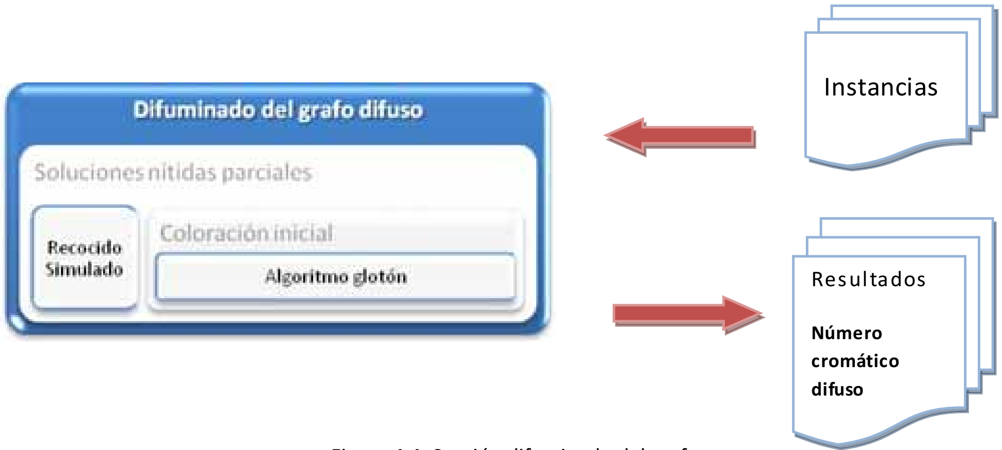


Figura 4.4. Sección difuminado del grafo

La constante "***PARTICIONES***" indica el número de cortes α .Se obtendrá un número cromático del α_i - *corte* $\hat{G}_{\alpha i}$, con i que toma valores en {*1,2, … ,PARTICIONES*} y corresponderá a la solución nítida parcial del α_i - *corte* $\hat{G}_{\alpha i}$.

Este bloque mandará llamar por cada corte al algoritmo de Recocido Simulado. Pero dado que el RS necesita una coloración inicial (de donde se generara la solución inicial s_0) para poder partir de un punto no muy lejano, el valor s_0 toma su valor con ayuda de un algoritmo glotón que se aproxime a la primera coloración, en otras palabras el α_1 - *corte* $\hat{G}_{\alpha 1}$.

A su vez ese valor obtenido (número cromático) se escribirá en el archivo de resultados para así formar el conjunto llamado "número cromático difuso".

A continuación se presenta el algoritmo:

Figura 4.5. Pseudocódigo de la función que difumina el grafo de acuerdo al número de particiones

```
FUNCION difuminadoDelGrafo ()
COMIENZO
   umbral ← 1 / (PARTICIONES * 1);
   solucionInicial ← algoritmoGloton ( umbral );
   Escribir (solucionInicial , umbral) ∊ Ŝ | Ŝ = {(s_1, α_1), … , (s_PARTICIONES, α_PARTICIONES)} en archivo de resultados;
   PARA i ← 2   HASTA   i < PARTICIONES
         umbral ← 1 / (PARTICIONES * i);
         solucion ←  recocidoSimulado ( umbral, solucionInicial );
         SI solucionInicial != solucion
               REPETIR
                     solucionInicial ← solucionInicial - 1;
                     solucion ←  recocidoSimulado ( umbral, solucionInicial );
               HASTAQUE solucionInicial != solución
         FINSI
         Escribir (solucionInicial , umbral) ∊ Ŝ | Ŝ = {(s_1, α_1), … , (s_PARTICIONES, α_PARTICIONES)} en archivo de resultados;
   FINPARA
TERMINAR
```

4.4 Soluciones nítidas parciales

Figura 4.6. Sección soluciones nítidas parciales

El objetivo de este bloque es encontrar la coloración mínima de cada grafo nítido correspondiente al α_1 - *corte* $\hat{G}_{\alpha 1}$, esa coloración se llama número cromático (solución nítida parcial) y pertenece al conjunto convexo llamada número cromático difuso. Para encontrar esta coloración mínima entra en acción el algoritmo de Recocido Simulado que a continuación se presenta, así como la toma de decisiones genéricas y específicas, que se mencionaron en la sección anterior.

4.4.1 Algoritmo de Recocido Simulado

Decisiones específicas	
Espacio de soluciones y estructura de entornos	El espacio de soluciones es el conjunto de coloraciones, lo cual se puede representar por un conjunto de particiones de vértices en subconjuntos. El costo de cada partición es el número de subconjuntos que contiene. La definición de entorno más simple es mover un vértice de un subconjunto a otro, de modo que siga manteniendo la factibilidad.
Función de costo ***f(s)***	Simplemente se calculan las incompatibilidades del grafo en ese momento según la estructura de entornos. Es decir un espacio de soluciones será aceptable en automático por el RS si la función de costo arroja un número menor en comparación con el espacio de soluciones anterior. Cuando la función de costo arroje un cero como resultado nos dirá que ya se encontró una coloración mínima, lo cual significa en este caso que se pudo reducir un color.
Solución inicial s_0	Como se menciono anteriormente el algoritmo de Recocido Simulado necesita de entrada una coloración inicial s_0 , pero debemos de tener un número inicial de colores para pintar el grafo nítido correspondiente al corte α_1, este valor es obtenido de un **algoritmo glotón** que se describirá en breve, y las solución inicial se obtiene de forma aleatoria tomando colores en base al algoritmo glotón. Es importante plantear que la solución del corte α_1 encontrada con ayuda del algoritmo glotón será la solución inicial del algoritmo RS para el corte α_2 y así para todo α_i - *corte* $\hat{G}_{\alpha i}$, con *i* que toma valores en {*1,2, … ,PARTICIONES*}.

Decisiones genéricas	
Temperatura inicial y función de decrecimiento ***t_0 , α(t)***	La temperatura tomara su valor en base a la función de costo, en donde la función de costo nos da el número de incompatibilidades según la estructura de entornos, dividida por el tamaño de vértices del grafo *TAM*. Así t_0 tomará su valor de $f(s_0)/TAM$, y *α(t)* estará dada por ***α(t) ← f(s)/TAM***. Al elegir esta ecuación se reduce mucho la probabilidad de que se acepte una solución "peor", ya que al permitir muchas malas decisiones el tiempo de ejecución se incrementa y tal vez se necesiten más iteraciones para encontrar una buena solución. En este punto se descubrió que el simple algoritmo de búsqueda local era óptimo, pero el tomar "pocas" malas decisiones le daba un contexto más amplio a la búsqueda de soluciones.
Número de iteraciones ***NREP***	La última constante por analizar es "***NREP***"; esta constante representa el número de "intentos" para tratar de eliminar un color y así llegar hasta la coloración mínima o que el criterio de parada sea cierto. Es importante saber también que este valor depende del tamaño del problema; es decir entre más vértices tenga un grafo, más iteraciones se necesitaran para tratar de eliminar un color. En los intentos computados tomamos un valor de 10,000 iteraciones para 120 vértices, ya que se obtuvo buenos resultados y el tiempo de ejecución fue rápido.
Criterio de parada	Este criterio nos indica si el sistema ya está "frio", y simplemente consideramos que está frio cuando se ha alcanzado las *NREP* iteraciones, ya que el margen dado a *NREP* es lo suficientemente amplio como para eliminar un color. Si se elimina un color se ejecuta de nuevo RS pero ahora partiendo de una coloración inicial disminuida en uno, así hasta que el criterio de parada sea cierto.

A continuación se presenta el pseudocódigo del algoritmo de Recocido Simulado aplicado al problema de coloración difusa

```
FUNCION recocidoSimulado ( IN umbral, IN  k , OUT  numeroCromatico)
COMIENZO
   s0 ← k;
   arreglo [TAM] [TAM]  ← leer instancias de ARCHIVO;
   contadorCiclo ← 0;
   colores [k] ← k  ∊  {0,1,…,k-1};
   solucion  [TAM]  ← tomar valores aleatorios  ∊  colores ;
   incAntes   ←  obtenerIncompatibilidades (arreglo, solución, umbral);
   temperatura  ←  incAntes / TAM;
   Seleccionamos un color aleatoriamente  e   y lo eliminamos de  colores;
   MIENTRAS   no se elimine un color  &&  contadorCiclo <  NREP
        Se selecciona un valor aleatoriamente del arreglo solucion  y se cambia aleatoriamente,
        dando prioridad a aquellos elementos de  solucion  que coincidan con e;
        incDespues  ←  obtenerIncompatibilidades (arreglo, solución, umbral);
        δ  ←  incDespues –  incAntes;
        SI  δ ≤  0
             Acepto la solución;
             temperatura ←  incDespues / TAM;
             incAntes ← incDespues;
        FINSI
        SINO
           Generar aleatoriamente  u  ∊  U (0,1);
             SI u <  exp (-δ/temperatura)
                  Acepto la solución aunque sea peor;
                  temperatura ←   incDespues  / TAM;
                  incAntes ← incDespues;
             FINSI
             SINO
                   No acepto la solución;
                   Se deshace el cambio hecho al  vector  solucion;
             FINSINO
       FINSINO
       SI  Ya no existen elementos en el vector solucion que coincidan con  e   &&  incDespues == 0
            k  ←  k -1;
          ¡Ya elimine un color!
       FINSI
       contadorCiclo ← contadorCiclo + 1;
   FINMIENTRAS
   REGRESA  k ;
TERMINAR;
```

Figura 4.7. Pseudocódigo de la función de Recocido Simulado

4.4.2 Algoritmo glotón

La lógica del algoritmo glotón es sencilla, se considera la variable *numeroCromatico* que llevará el conteo del número de colores con los que se colorea el grafo hasta el momento; se comienza con una coloración *numeroCromatico* = 1. Por cada vértice del grafo se verifica que sus vecinos no tengan la misma coloración, en el caso que así sea se incrementa la coloración del vértice vecino en uno, si al hacer este incremento se rebasa el valor de *numeroCromatico,* se incrementa en uno *numeroCromatico* ya que se necesita de un color mas pintar el grafo.

```
FUNCION algoritmoGloton ( IN umbral, OUT  numeroCromatico)
COMIENZO
   numeroCromatico ← 0;
   arreglo [TAM] [TAM]  ← leer instancias de ARCHIVO;
   colores [TAM] ;
   PARA  i  ← 0   HASTA TAM -1
       indice  ←  colores [i];
       PARA j  ← i + 1   HASTA  TAM -1
           col   ←   arreglo [i][j];
           SI  col  ≥  umbral
               crom ←  colores [j];
               SI   indice  ==  crom
                   colores [j] ← crom + 1;
                   SI  crom + 1  >  numeroCromatico
                       numeroCromatico  ← crom + 1;
                   FINSI
               FINSI
           FINSI
       FINPARA
   FINPARA
   REGRESA  numeroCromatico + 1;
TERMINAR
```

Figura 4.8. Pseudocódigo de la función del algoritmo glotón

4.5 Número cromático difuso

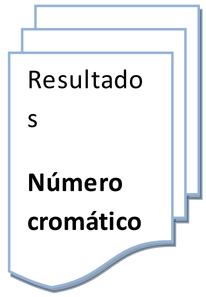


Figura 4.9. Sección número cromático difuso

Las soluciones nítidas parciales (número cromático del corte α_i) obtenidas mediante RS y que representan el número cromático difuso son escritas en el archivo de resultados.

Primero se escribirá el valor correspondiente al número de vértices *"TAM",* seguido del número de cortes "*PARTICIONES*", posteriormente se escribirá el conjunto de valores $\hat{S}$, recordando que

$$\hat{S} = \{(s_1, \alpha_1), \ldots , (s_n, \alpha_n)\}$$

$\hat{S}$ es el conjunto difuso no normalizado cuyos elementos son las parejas formadas por el número cromático del α_i - *corte* $\hat{G}_{\alpha i}$ y α_i , con i que toma valores en $\{1, \ldots ,PARTICIONES\}$.

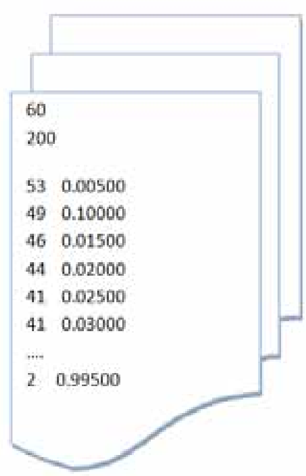
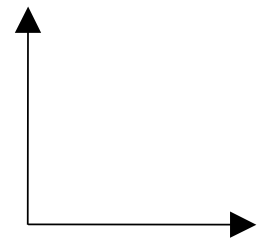


Figura 4.10. Ejemplo real de archivo de resultados

4.6 Presentación de Resultados

Después de ejecutar el programa múltiples veces se obtuvo siempre un comportamiento particular que en breve describiremos. Es importante mencionar que el programa fue escrito en el lenguaje de programación C y que los resultados se graficaron de la siguiente forma:

Coincidencias de Número Cromático

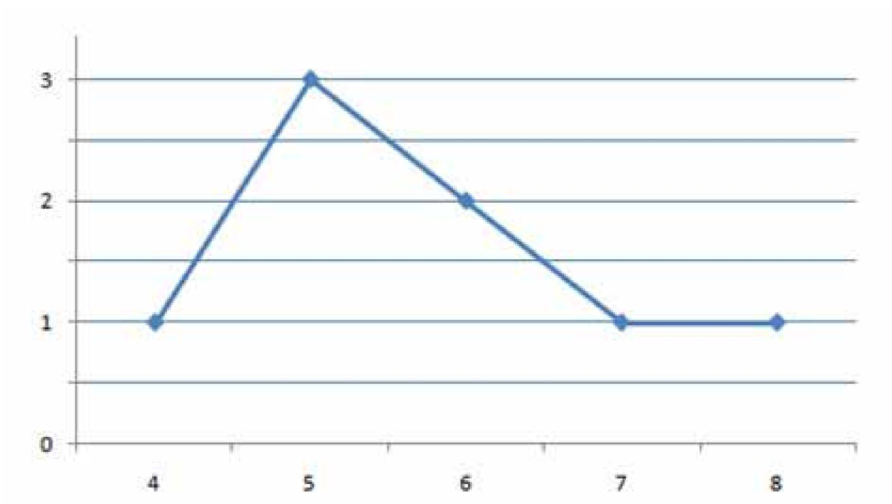


Número cromático

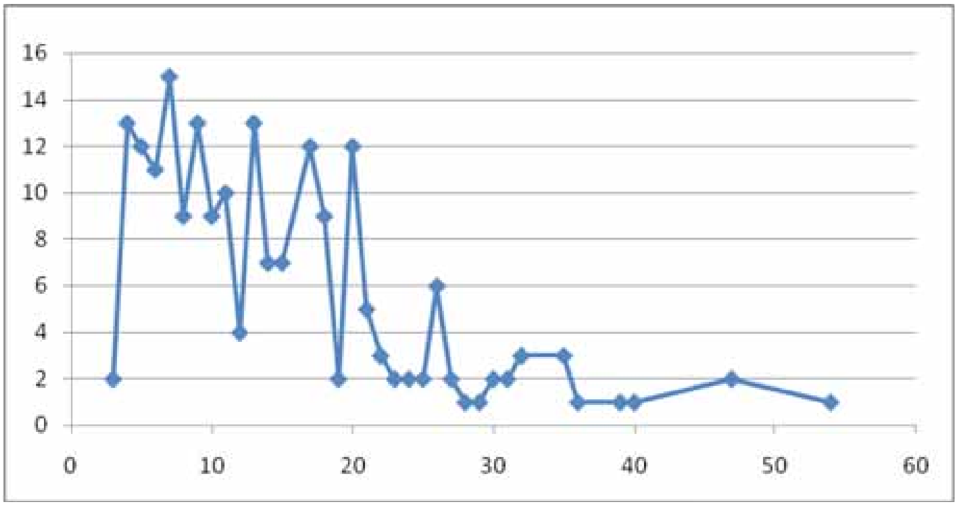
Siendo que el conjunto de valores de "Número cromático" pertenecen al número cromático difuso, ósea la solución difusa *"$\hat{S}$";* pero son sólo un subconjunto de estos ya lo que se graficará son las coincidencias de los distintos números cromáticos obtenidos en cada corte α_i.

Por ejemplo si obtuvimos los siguientes resultados (número cromático difuso) se graficara de la siguiente manera:

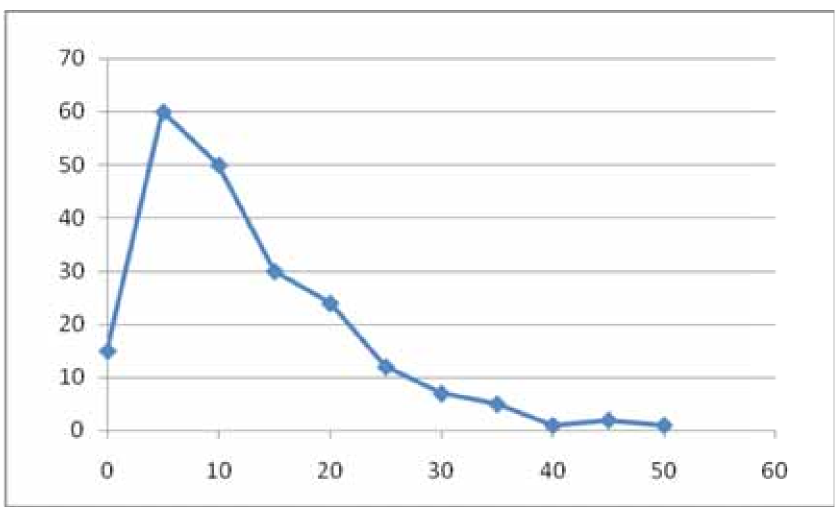
a,8
b,7
c,6
d,6
e,5
f,5
g,5
h,4



Así para un grafo difuso de 60 vértices y 200 cortes (cada corte de la misma proporción) estos fueron los resultados obtenidos.



De primera instancia se observa cómo crece muy rápidamente y posteriormente decrece lentamente, asi como un sesgo positivo (a la derecha). Para observar mejor este comportamiento se agruparan estos valores en clases, cada una con un intervalo de 5 (lo que en estadística descriptiva se llama "histograma") y esta es la grafica asociada:



Agrupados con esta marca de clase, se observa de una manera más clara el comportamiento descrito anteriormente.

Se esperaría una grafica con una distribución normal, dado que las instancias generadas tienen este comportamiento, pero al correr el programa múltiples veces se observa este mismo patrón, que en trabajos posteriores se podría estudiar.

Referencias

[1] Javier Ramírez Rodríguez, *Extensiones del problema de coloración de grafos,* PhD thesis. Universidad Complutense de Madrid, Madrid; 2000.
http://www.ucm.es/BUCM/tesis/mat/ucm-t24770.pdf
Revisada el 13 de Diciembre del 2009

[2] Zadeh, L.A (1965). Fuzzy sets. *Information and control.*

[3] Rosenfeld, A. (1975). Fuzzy Graphs. *Fuzzy sets and their applications to cognitive and decision process, Academic Press New York.*

[4] Rafael Martí, *Procedimientos Metaheurísticos en Optimización Combinatoria.*
Departamento de Estadística e Investigación de Operaciones, Facultad de Matemáticas, Universidad de Valencia.
http://www.uv.es/rmarti/paper/docs/heur1.pdf
Revisada el 13 de Diciembre del 2009

[5] Kathryn A. Dowsland, Belarmino Adenso Diáz, *Diseño de Heurísticas y Fundamentos de Recocido Simulado.*
University of Nottingham, Universidad de Oviedo.

[6] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, *Optimization by Simulated Annealing.*
Science, Volume 220, Number 4598; 1983.
http://www.idi.ntnu.no/emner/tdt4/kirkpatrick83optimizationsimulatedannealing.pdf
Revisada el 13 de Diciembre del 2009

[7] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth and Augusta H. Teller, *Equation of State Calculations by fast Computing Machines.*
The Journal of Chemical Physics, Volume 21, Number 6; 1953.
http://people.sc.fsu.edu/~beerli/mcmc/metropolis-et-al-1953.pdf
Revisada el 13 de Diciembre del 2009

Extra:

- *Compilación de archivo C:*
  gcc coloracionDifusa.c -o dif -lm
- *Ejecución*
  ./dif

Palabras clave:
Optimización combinatoria, Grafo, Grafo difuso, Número Cromático, Coloración difusa, Recocido Simulado, Búsqueda local, Algoritmo evolutivo, Heurística, Coloración mínima.