*matemática:* Una publicación de FCNM – ESPOL
2013, Vol. 11, No. 2

# DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO GRASP PARA EL PROBLEMA DE COLORACIÓN DE GRAFOS

**Delgado Erwin**[1]

**Resumen.** *Una de las dificultades en la búsqueda de la solución óptima a problemas de optimización combinatoria es el elevado costo computacional para obtenerlas. Por ello se hace necesaria la utilización de algoritmos basados en metaheurísticas para obtener soluciones factibles a un costo computacional razonable. En este contexto, en el presente trabajo se aborda el diseño de una heurística basada en la metodología GRASP para el Problema de Coloración de Grafos. Este algoritmo se lo ha implementado en Mathematica 8.0.4.0 ® y ejecutado con diversas instancias del problema para medir la calidad del mismo.*

**Palabras Claves:** GRASP, Coloración de Grafos, Metaheurísticas.

**Abstract.** *One of the difficulties in finding the best solution for combinatorial optimization problems is the high computational cost to obtain them. Therefore it is necessary to use algorithms based on metaheuristics for feasible solutions to a reasonable computational cost. In this context, this paper addresses the design of a heuristic based on the GRASP methodology for graph coloring problem. This algorithm is implemented in Mathematica has 8.0.4.0 ® of the problem to measure quality.*

**Keywords:** GRASP, graph coloring, metaheuristics.

**Recibido:** Julio 2013
**Aceptado:** Agosto 2013

## 1. INTRODUCCIÓN

Muchos de los problemas de optimización combinatoria se caracterizan por su complejidad computacional en la determinación de una solución óptima por medio de métodos exactos. Por ello, dada esta particularidad desde hace mucho tiempo se intenta obtener soluciones factibles a estos problemas en un tiempo computacional razonable por medio de la aplicación de diversas metaheurísticas. Precisamente, El Problema de Coloración de Grafos (PCG) es un problema combinatorio el cual para grandes instancias se requiere la implementación de algoritmos que provea soluciones factibles a un costo computacional razonable en comparación a los obtenidos por métodos exactos. En este contexto, en el presente trabajo se aborda el diseño e implementación de una heurística para el PCG. Antes de definir formalmente el PCG es importante mencionar la siguiente definición establecida en [14].

**Definición 1.1.**
Dado un grafo $G = (V, E)$ donde $V$ es un conjunto de vértices y $E = \{(v_i, v_j) : v_i, v_j \in V, v_i \neq v_j\}$ un conjunto de arcos definidos en $G$. Una coloración propia de $G$ es una función $f: V \rightarrow [1, k]$ tal que $\forall (a, b) \in E$ se cumple que $f(a) \neq f(b)$, es decir, asigna a cada vértice de $G$ un color diferente al asignadoa cualquier otro vértice adyacente.

En este contexto, el Problema de Coloración de Grafos, PCG, consiste en determinar el mínimo número k de colores diferentes con el objeto de realizar una coloración propia de $G$. A este mínimo valor de $k$ se le conoce como número cromático de $G$, denotado por $\chi(G)$.

## 2. FORMULACIÓN MATEMÁTICA DEL PROBLEMA DE COLORACIÓN DE GRAFOS

A continuación, se presenta un modelo en programación lineal que permite encontrar la solución óptima al problema de coloración de colores.

### 2.1 ÍNDICES

Sea $G = (V, E)$ un grafo no dirigido y $C$ un conjunto de colores
$i, k$: Índices referidos al conjunto de vértices
$j$: índice referido al conjunto de colores

### 2.2 VARIABLES

$$x_{ij} = \begin{cases} 1, si\ el\ vértice\ i\ es\ coloreado \\ con\ el\ color\ j \\ 0, si\ no \end{cases}$$

[1] *Delgado Erwin, M.Sc., Profesor de la Escuela Superior Politécnica del Litoral (ESPOL).*
*(e_mail: edelgado@espol.edu.ec).*

### 2.3 FORMULACIÓN MATEMÁTICA

$$\text{Min} \sum_i \sum_j j x_{ij} \qquad (1)$$

s.a.

$$\sum_j x_{ij} = 1, \quad \forall i \in V \qquad (2)$$

$$x_{ij} + x_{kj} \leq 1, \quad \forall (i,k) \in E, \forall j \in C \qquad (3)$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in E \qquad (4)$$

La ecuación (1) representa la función objetivo, la cual permite minimizar el número de colores utilizados para realizar una coloración propia. La restricción (2) permite asegurar que cada vértice sea coloreado por algún color. La restricción (3) permite garantizar que para cada arco definido en el grafo, sus vértices sean coloreados con colores distintos. Por último, la restricción (4) indica la naturaleza de las variables de decisión.

## 3. REVISIÓN DE LITERATURA

Uno de los enfoques utilizados para resolver el PCG es por métodos exactos. Así, Isabel Méndez et al [8] propone un modelo de programación entera, el mismo que es resuelto por un algoritmo de planos cortantes. De igual manera, Hans Bodlaender [10] desarrolla un modelo exacto incorporando memoria polinomial mientras que $C$. Lucet et al [7] propone un método exacto basado en una descomposición lineal del grafo. Sin embargo estos métodos tienen limitaciones por la cardinalidad del conjunto de vértices del grafo Precisamente, el PCG es uno de los problemas de optimización combinatoria, categorizado como NP-Hard [3], por tal motivo múltiples heurísticas se han diseñado con el objeto de obtener soluciones factibles de buena calidad en tiempos de ejecución adecuados. Uno de ellos es el realizado por [14] quién implementó un algoritmo glotón, ordenando en primer lugar los vértices de acuerdo al grado de adyacencia y asignando en cada iteración un color cada vértice obteniendo así una solución factible. De igual manera, Cédric Avanthay et al [5], desarrollaron una búsqueda local en un vecindario variable. Uno de los trabajos a destacar, dentro de la categoría de métodos de búsqueda local, es el propuesto en [15], en el cual se implementa una búsqueda local con una función de evaluación de la coloración de un grafo, la misma que está en términos del tamaño de cada partición del conjunto de vértices, así como del número de arcos que violen la condición de coloración.

Dentro de los trabajos que se fundamentan en algoritmos evolutivos, podemos destacar los realizados por Zhipeng Lu [4], quien desarrolló un algoritmo memético para este problema, resolviendo iterativamente el $k$ -coloring desde un valor de $k$, disminuyéndolo hasta que no exista alguna solución factible. Este algoritmo combina un genético con una búsqueda tabú, incorporando un operador de cruce adaptativo; y el realizado por A. Eiben et al [9] que desarrolla un algoritmo evolutivo para la determinación de buenas soluciones factibles. Incorpora un mecanismo adaptativo que permite cambiar el fitness de la función objetivo en cada iteración. Por otro lado, A. Hertz et al [6], diseña un algoritmo fundamentado en la búsqueda tabú, la misma que la comparó con los resultados obtenidos por algunos trabajos desarrollados con recocido simulado observando un mayor perfomance para instancias de gran cardinalidad.

## 4. GRASP

El GRASP[2] es una metaheurística en la que cada iteración consiste de dos etapas: construcción y búsqueda local [16]. El objetivo de la fase de construcción es la determinación de una solución factible la cual es mejorada en la fase de búsqueda local, lo que se muestra en el seudocódigo mostrado a continuación, considerando un problema de minimización:

**Algorithm 1 GRASP**

```
1: procedure GRASP(MaxIter,Seed)
2:     for k=1,...,MaxIter do
3:         Solución1 ← EtapaContrucción(Seed)
4:         Solución ← BúsquedaLocal(Solución1)
5:         if Solución<BestSolución then
6:             BestSolución=Solución
7:         end if
8:     end for
9:     return BestSolución
10: end procedure
```

El fundamento de la fase de construcción de una solución factible es un algoritmo glotón no determinístico, en la que iterativamente se incorporan los elementos a una solución parcial.

El elemento a ser incorporado en la solución parcial es escogido aleatoriamente desde una lista RCL, la cual está formada por aquellos elementos que produzcan los más bajos costos incrementales en la función objetivo. Luego que el elemento seleccionado es incorporado en la solución parcial, esta es actualizada y se procede a iterar nuevamente.

---

[2] *Greedy Randomized Adaptive Search Procedure.*

## 5. DISEÑO DE HEURÍSTICA BASADA EN LA METODOLOGÍA GRASP PARA EL PROBLEMA DE COLORACIÓN DE GRAFOS

A continuación se detalla cada componente del algoritmo basado en la metodología GRASP para el problema de coloración de grafos.

### 5.1. DISEÑO DE FASE DE CONSTRUCCIÓN

El objetivo principal en la fase de construcción, es la determinación de una solución factible del problema de coloración de grafos. Esta solución se la ha de de- terminar por medio de un algoritmo glotón no determinístico. Para tal efecto, se realizó una adaptación al algoritmo desarrollado por Daniel Brélaz [14] para el problema de coloración de grafos. A continuación se muestra el seudocódigo de la fase de construcción de la solución inicial:

**Algorithm 2** Fase de constucción

1: **procedure** CONSTRUCCIÓN(Seed)
2: solución $\leftarrow \phi$
3: Ordenar los vértices de acuerdo a su grado en orden decreciente.
4: Construir la lista de candidatos restringidas $RCL$ definida por $\{v \in V | grad(v) \geq C_{min} + \alpha(C_{max} - C_{min})\}$, donde $C_{max}$ y $C_{min}$ son el máximo y mínimo grado de entre todos los vértices.
5: Seleccionar y colorear aleatoriamente un vértice $s$ de la lista $RCL$
6: **while** $|solución| \neq |V|$ **do**
7: Ordenar los vértices no coloreados de acuerdo a su grado de saturación en orden decreciente.
8: Construir la lista de candidatos restringidas $RCL$ definida por $\{v \in V | grad(v) \geq C_{min} + \alpha(C_{max} - C_{min})\}$, donde $C_{max}$ y $C_{min}$ son el máximo y mínimo grado de saturación de entre todos los vértices no coloreados.
9: Seleccionar y colorear aleatoriamente un vértice $s$ de la lista $RCL$, utilizando el mínimo número de colores para obtener una solución factible parcial.
10: solución $\leftarrow$ solución $\cup \{s\}$
11: **end while**
12: **return** *solución*
13: **end procedure**

Antes de explicar en detalle la adaptación realizada, es necesario introducirnos en la siguiente definición establecida en [14].

Definición 5.1. Sea $G$ un grafo simple y $C$ una coloración parcial de $G$. Se define al grado de saturación de un vértice como el número de diferentes colores asignados a sus vértices adyacentes.

#### 5.1.1. CRITERIO GLOTÓN

Una de las componentes del algoritmo GRASP es la construcción de un algoritmo glotón. Diversos enfoques pueden ser considerados para el efecto. Por ejemplo, se podría priorizar el grado de los vértices en el momento de colorear un vértice. En el presen- te trabajo inicialmente se procede a ordenar de forma decreciente los vértices de acuerdo a su grado de saturación. Este ordenamiento nos permite construir una lista de ≪buenos≫ candidatos a ser considerados en la siguiente iteración. Sin embargo, este criterio cambia a partir del momento en que se va a colorear el segundo vértice de la solución parcial ya que el ordenamiento se lo realiza con base en los grados de saturación de los vértices no coloreados. Esta idea surge del hecho de que vértices de un número elevado de arcos incidentes se los debe colorear primero con el objeto de minimizar el número de colores utilizados para realizar una coloración propia.

#### 5.1.2. LONGITUD DE LA LISTA RCL

Una de las características del algoritmo GRASP es que la construcción de la solución factible no es determinística ya que en cada iteración se incorpora un vértice a la solución parcial de forma aleatoria, el mismo que pertenece a una lista, denominada RCL, de buenos candidatos. Diversas estrategias pueden ser consideradas para la determinación de la longitud de la lista de candidatos; así, puede considerarse una lista con longitud fija en todas las iteraciones o de longitud variable. Por ejemplo, sean Cmax y Cmin el máximo y mínimo grado de saturación de entre todos los vértices (o el grado de los vértices) entonces la lista RCL está definida como:

$$RCL = \left\{ \begin{matrix} v \in V | grad(v) \geq C_{min} + \\ \alpha(C_{max} - C_{min}) \end{matrix} \right\} \quad (5)$$

donde $\alpha$ es un parámetro a considerar.

#### 5.1.3. ELECCIÓN DEL PRÓXIMO VÉRTICE A SER COLOREADO

Luego de construida la lista RCL formada por los vértices considerados buenos candidatos y que no

han sido coloreados, se procede a escoger aleatoriamente un vértice $v$ de esta lista. Debido a que el GRASP debe realizarse un número MAX_ITER de iteraciones, la función generadora de números aleatorios está en términos de una variable entera SEED que cambiará en cada iteración para garantizar la exploración de otras regiones. Al vértice $v$ escogido se le asignará un color que minimice el número de colores utilizados para realizar una coloración parcial del grafo y que garantice la factibilidad del mismo.
Este procedimiento se lo realizará iterativamente hasta que todos los vértices sean coloreados, obteniendo así una solución factible inicial.
Sin embargo, no necesariamente la solución factible obtenida en la etapa de construcción es la óptima, por lo que se hace imprescindible un algoritmo de mejora de la solución actual, objetivo que se cumple con la fase de búsqueda local.

## 5.2. DISEÑO DE FASE DE BÚSQUEDA LOCAL

Como se ha explicado anteriormente, la fase de búsqueda local tiene por objeto mejorar una solución actual obteniendo así, en cada iteración, una solución de mejor calidad.
En este contexto, la fase de búsqueda local empieza con una solución inicial dada por la fase de construcción.

### 5.2.1. VECINDARIO DE UNA SOLUCIÓN

Uno de los requerimientos de cualquier búsqueda local es la forma de construcción de vecindades a una solución actual. En este sentido, una solución vecina a la actual es obtenida al colorear cualquier vértice escogido aleatoriamente por un color diferente, sea éste uno ya utilizado o por medio de otro color.

### 5.2.2. EVALUACIÓN DE UNA SOLUCIÓN

Conforme a la construcción del vecindario de una solución cualquiera, es evidente que al realizar un intercambio de colores podría generar soluciones no factibles, es decir asignar a un vértice un color que viole la restricción que cualquier vértice adyacente sea coloreado de la misma manera.
Sean $C = \{C_1, C_2, \dots, C_k\}$ una partición de colores en el grafo $G = (V, E)$ y $E_i$, $\forall i = 1,2, \dots, k$ el conjunto formado por los arcos que tienen sus vértices adyacentes en la partición $C_i$, entonces el valor de la coloración definida por $C$ está dada por [15]:

$$f(C) = -\sum_{i=1}^{k}|C_i|^2 + \sum_{i=1}^{k}|C_i||E_i| \quad (6)$$

Un hecho a considerar en la ecuación 6 es que esta función de evaluación promueve la creación de particiones de gran cardinalidad, penalizando aquellas que contenga un gran conjunto de arcos que violen las restricciones dadas.

### 5.2.3. ACTUALIZACIÓN DE LA SOLUCIÓN ACTUAL

Existen dos maneras de actualizar la solución actual. Una de ellas es explorar todo el vecindario de la solución actual y reemplazarla por la solución vecina que posea el menor valor en la función definida en la sección anterior. Sin embargo, una de las dificultades de tal estrategia es el consumo de recursos, especialmente tiempo computacional por lo que en el presente trabajo se ha considerado reemplazar la solución actual por la primera solución vecina que mejore el costo de la coloración actual.
Es importante destacar que la exploración se la realiza considerando coloraciones factibles e infactibles y que esta etapa se la ejecuta durante un tiempo máximo.
A continuación se muestra el seudocódigo de la fase de búsqueda local

**Algorithm 3** Búsqueda Local

```
1: procedure BÚSQUEDALOCAL(solución) inicial ← f(solución)
2:     while Time_max no se cumpla do Determine una solución s ∈ N(solución) por medio del reemplazo del color de un vértice escogido aleatoriamente por un nuevo color.
3:         if f(s) <inicial then
           solución ← s
           inicial ← f(s)
4:         end if
5:     end whilereturn solución
6: end procedure
```

## 5.3. CALIBRACIÓN

Uno de los aspectos a analizar en la presente sección es la estrategia a seguir con respecto al tamaño de la lista RCL. Para tal efecto, se ejecutó el algoritmo desarrollado con algunas instancias del problema de coloración de grafos[3] durante 10 iteraciones cada una de ellas con una duración de 60 segundos.

[3] *http://mat.gsia.cmu.edu/COLOR/instances.htmlXXLEI*

En la tabla I se muestra un resumen de los resultados obtenidos al ejecutar el algoritmo en tres instancias específicas. En la primera columna se observa el nombre de la instancia, en la segunda columna el número de colores óptimo para realizar una coloración propia del grafo dado. En la tercera columna se muestran los porcentajes de aciertos con respecto al óptimo tanto de la estrategia con longitud de lista fija (*F*) y longitud variable (*V*) y por último los tiempos promedios en la obtención del mínimo número de colores requeridos, obtenidos en la ejecución de cada instancia.
Con base en los resultados obtenidos, tanto tiempos como en calidad de la solución, se ha considerado la longitud de la lista RC L como variable, es decir, la lista RC L estará definida como en la ecuación 5

**TABLA I**
*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*
**Calibración: Análisis de resultados**

| Instancia | # óptimo de colores | Resultados | | | |
|---|---|---|---|---|---|
| | | % de aciertos | | tiempos(seg.)[a] | |
| | | F | V | F | V |
| myciel5.col | 6 | 100 | 100 | 3.852 | 3.392 |
| myciel6.col | 7 | 60 | 70 | 30.157 | 31.727 |
| myciel7.col | 8 | 0 | 0 | 49.600 | 46.425 |

***a*** Tiempo promedio que toma en alcanzar el mínimo

## 6. RESULTADOS COMPUTACIONALES

Luego de la calibración de los parámetros del algoritmo GRASP desarrollado en la sección anterior, se ejecutó el mismo con diversas instancias en una PC con procesador INTEL CORE i3 3.07 GHz de 64 bits, durante un máximo de MAX_ITER, cada una de ellas ejecutadas durante un tiempo *t _max*, cuyos resultados se muestran en la tabla II.
En las figuras 1,2 y 3 se muestra la tendencia del número de colores requeridos para una coloración propia en cada instancia utilizada en función del tiempo. Como se puede observar, el algoritmo es eficiente en la búsqueda de soluciones de buena calidad, especialmente para instancias de pequeña escala.
A continuación, en las figuras 4, 5 y 6 se muestra la tendencia del número de colores requeridos para una coloración propia, pero sólo en el mínimo valor encontrado luego de ejecutar el número máximo de iteraciones. Como se puede observar, luego que se ha determinado una solución factible inicial, el tiempo que toma en alcanzar el mínimo local es razonable en comparación con los que se toman al aplicar un modelo que los resuelva en forma exacta.

**TABLA II**
*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*
**Resultados de la ejecución del algoritmo para diversas instancias**

| Instancia | # óptimo de colores | # de nodos | # de arcos | Parámetros | | # de colores |
|---|---|---|---|---|---|---|
| | | | | MAX_ITER | t max(seg) | |
| queen8_8.col | 9 | 64 | 728 | 400 | 45 | 10 |
| queen9_9.col | 10 | 81 | 2112 | 280 | 90 | 12 |
| queen11_11.col | 11 | 121 | 3960 | 255 | 120 | 14 |
| myciel3.col | 4 | 11 | 20 | 320 | 10 | 4 |
| myciel4.col | 5 | 23 | 71 | 320 | 10 | 5 |
| myciel5.col | 6 | 47 | 236 | 320 | 10 | 6 |
| myciel6.col | 7 | 95 | 755 | 320 | 10 | 7 |
| myciel7.col | 8 | 191 | 2360 | 320 | 10 | 9 |

**FIGURA 1**
*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*
**Gráfica de la evolución del número de colores requeridos en la instancia queen8_8.col**

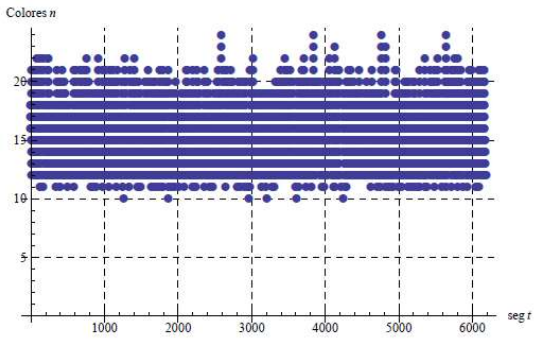



**FIGURA 2**
*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*
**Gráfica de la evolución del número de colores requeridos en la instancia queen9_9.col**

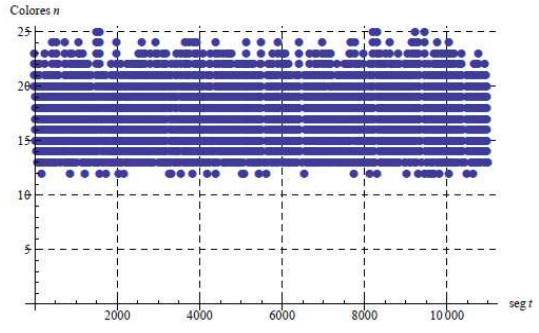


**TABLA III**

*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*

**Comparación entre heurística propuesta y el algoritmo de Brélaz**

| Instancia | Heurística Propuesta | Algoritmo Brélaz |
|---|---|---|
| queen8_8.col | 10 | 15 |
| queen9_9.col | 12 | 15 |
| queen11_11.col | 14 | 16 |
| myciel3.col | 4 | 4 |
| myciel4.col | 5 | 5 |
| myciel5.col | 6 | 6 |
| myciel6.col | 7 | 7 |
| myciel7.col | 9 | 8 |

Otra forma de validar los resultados que se obtienen al ejecutar el algoritmo propuesto para diversas instancias, es por medio de la comparación de los mismos con los que se obtienen al aplicar una función específica de la librería de Mathematica 8.0.4.0 ® denominada ≪VertexColoring≫, la misma que consiste en la implementación del algoritmo de Brélaz [14].

En la tabla III se muestra los resultados al ejecutar diversas instancias tanto con la heurística propuesta como con el algoritmo de Brélaz.

**FIGURA 3**

*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*

**Gráfica de la evolución del número de colores requeridos en la instancia queen11_11.col**

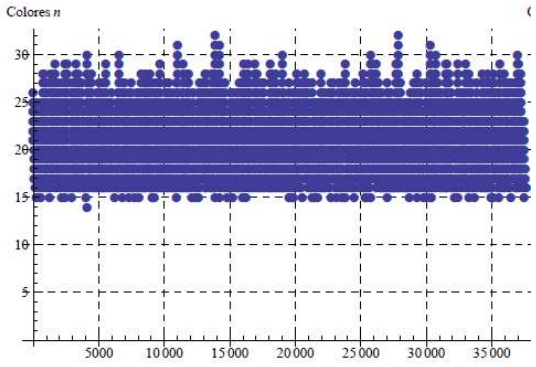



**FIGURA 4**

*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*

**Evolución del número de colores requeridos en la instancia queen8_8.col**

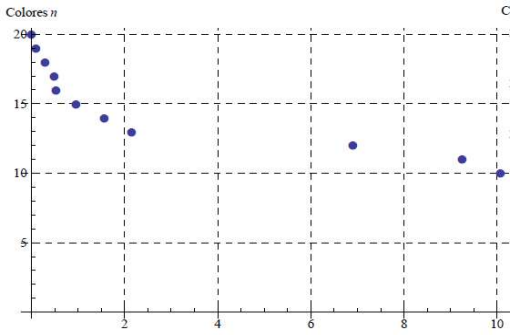



**FIGURA 5**

*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*

**Evolución del número de colores requeridos en la instancia queen9_9.col**

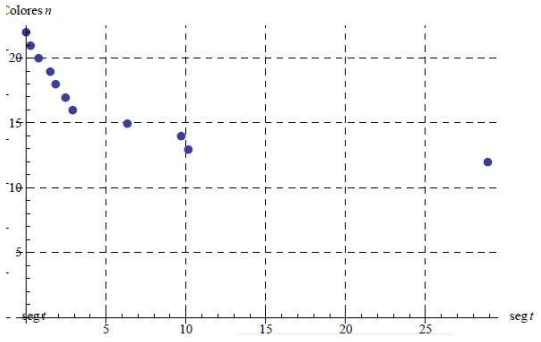
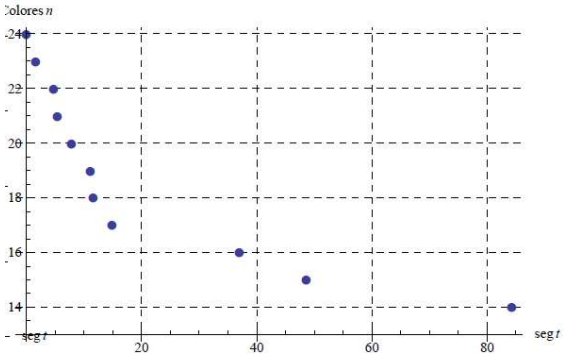



**FIGURA 6**

*Diseño e implementación de un algoritmo GRASP para el problema de coloración de grafos*

**Evolución del número de colores requeridos en la instancia queen11_11.col**




## 7. CONCLUSIONES Y RECOMENDACIONES

Conforme a los resultados obtenidos al aplicar la heurística propuesta al problema de coloración de grafos, es evidente la gran utilidad de aplicar

heurísticas en la búsqueda de soluciones factibles de buena calidad siendo computacionalmente mejores en tiempos de ejecución en comparación a los obtenidos por métodos exactos. Sin embargo, para problemas de gran cardinalidad aún se observa una gran variación con respecto a los valores óptimos.

Esto se debe principalmente por la no utilización de estructuras de memorias a corto y largo plazo, por lo que se recomienda en futuros trabajos incorporarlos.

## REFERENCIAS BIBLIOGRÁFICAS Y ELECTRÓNICAS


**[1]. DING-ZHUE DU AND PANOS M. PARDALOS.** (2005). "*Handbook of combinatorial optimization*". Volume B, Springer.

**[2]. MCCOLLUM B., MCMULLAN P.** (2007). "*The Second International Timetabling Competition: Examination Timetabling Track*". Technical Report. Queen's University Belfast.

**[3]. DING-ZHUE DU AND PANOS M. PARDALOS**, (2005). "*Handbook of combinatorial optimization*", Volume B, Springer.

**[4]. ZHIPENG LU, JIN-KAO HAO.** (2009). "*A memetic algorithm for graph coloring*", European Journal of Operations Research, ELSEVIER.

**[5]. AVANTHAY C., HERTZ A., ZUFFEREY N.** (2003). "*A variable neighborhood search for graph coloring*". European Journal of Operational Research, ELSEVIER.

**[6]. A. HERTZ, D. DE WERRA.** (1987). "Using Tabu Search Techniques for Graph Coloring. Computing". By Springer-Verlag.

**[7]. C.LUCET, F. MENDES, A. MOUKRIM**. (2004). "*An Exact method for graph coloring. Computer & Operations Research*", ELSEVIER.

**[8]. MÉNDEZ I., ZABALA P.** (2007). "*A cutting plane algorithm for graph coloring. Discrete Applied Mathematics*". ELSEVIER.

**[9]. EIBEN A., VAN DER HAUW J., VAN HENNERT J. "***Graph Coloring with Adaptive Evolutionary Algorithms*". Journal of Heuristics, volume 4:1.

**[10]. BODLAENDER H., KRATSCH D.** (2006). "An exact algorithm for graph coloring with polynomial memory". Utrecht University.

**[11]. JOHNSON D., ARAGÓN C., MCGEOCH L., SCHEVON C.** (1990). "*Optimization by simulated annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partition, Operations Research Society of America*".

**[12]. SOPENA E.** (2001). "*Oriented Graph Coloring*", ELSEVIER.

**[13]. WERRA, D.** (1990). "*Heuristics for Graph Coloring, Computational Graph Theory, Compute*". Suppl. 7, Springer, Vienna, 191-208.

**[14]. BRÉLAZ, D.** (1979). "*New methods to color the vertices of a graph*", Communications of the Assoc. of Com- put. Machinery 22, 251-256.

**[15]. JOHNSON, D. S., ARAGON, C. R., MCGEOCH, L.A., AND SCHEVON, C. "***Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning*". Operations Research, 39(3):378406.

**[16]. RESENDE M., RIBEIRO C.** (2002). "*Greedy Randomized Adaptive Search Procedures*", AT&T Labs Research Technical Report. Hanbook in Metaheuristic, Fred Glover.

**[17]. GLOVER FRED, KOCHENBERGER.** (2003). "*HandBook of Metaheuristics*", International Series in Operations Research & Management Science.

**[18]. TALBI EL-GHAZALI.** (2009). "*Metaheristics from design to implementaction*", John Wiley & Sons, Inc., Hoboken, New Jersey.