Revista de Matemática: Teoría y Aplicaciones 2005 **12**(1 & 2) : 111–120
cimpa – ucr – ccss issn: 1409-2433

# UN ALGORITMO EVOLUTIVO PARA RESOLVER EL PROBLEMA DE COLORACIÓN ROBUSTA.

Pedro Lara Velázquez * Miguel Angel Gutiérrez Andrade †
Javier Ramírez Rodríguez ‡ Rafael López Bracho §

*Recibido/Received: 16 Abr 2004*

**Resumen**

Sean $G$ y $\overline{G}$ un par de gráficas complementarias. Dada una función de peso definida sobre las aristas de $\overline{G}$ , se dice que la rigidez de una $k$-coloración válida de $G$ es la suma de los pesos de las aristas de $\overline{G}$ que unen vértices del mismo color. Con base en la anterior definición, se plantea el *Problema de Coloración Robusta* al buscar la $k$-coloración válida de rigidez mínima. Yáñez y Ramírez probaron que este problema es $NP$-duro. En este trabajo se presenta un algoritmo evolutivo basado en la técnica de búsqueda dispersa, la cual obtiene soluciones óptimas, en las instancias para las que se conoce la solución óptima, y obtiene las mejores soluciones conocidas comparadas con otras heurísticas, tales como: recocido simulado, búsqueda tabú y enumeración parcial.

**Palabras clave:** Búsqueda dispersa, coloración de gráficas, heurísticas, optimización combinatoria, optimización discreta.

**Abstract**

Let $G$ and $\overline{G}$ be two complementary graphs. Given a penalty function defined over the edges of $\overline{G}$, it is said that the rigidity of a $k$-coloring of $G$ is the summation of

*División de Estudios de Posgrado de la Facultad de Ingeniería, Universidad Nacional Autónoma de México, Cd. Universitaria, México, D.F., C.P. 04510 ; E-Mail: pedro_lara@yahoo.com.

†Departamento de Ingeniería Eléctrica, Universidad Autónoma Metropolitana - Iztapalapa, Av. San Rafael Atlixco No. 186, Col. Vicentina, Del. Iztapalapa, México, D.F., C.P. 09340; Fax: (52 55)58-04-46-40; E-Mail: gamma@xanum.uam.mx.

‡Departamento de Sistemas, Universidad Autónoma Metropolitana - Azcapotzalco, Av. San Pablo No. 180, Col. Reynosa Tamaulipas, Del. Azcapotzalco, México, D.F., C.P. 02200; Fax: (52 55)53-94-45-34; E-Mail: jararo@correo.azc.uam.mx.

§Departamento de Sistemas, Universidad Autónoma Metropolitana - Azcapotzalco, misma dirección que Ramírez, E-Mail: rlb@correo.azc.uam.mx.

the penalties of the edges of $\overline{G}$ that join vertices whose endpoint are equally colored. Based on this previous definition, the *Robust Coloring Problem* is set when searching the valid $k$-coloring of minimum rigidity. Yáñez and Ramírez proved that this is an $NP$-hard problem. In this work we present an evolutive algorithm based in the scatter search technique, which obtains optimal solutions for those instances for which an optimal solution is known, and obtains the best known solutions compared to other heuristics, such as: simulated annealing, tabu search and partial enumeration.

**Keywords:** Scatter search, graph coloring, heuristics, combinatorial optimization, discrete optimization.

**Mathematics Subject Classification:** 68T20, 90C27, 90C59.

# 1 Introducción

Sea una gráfica simple G, con conjuntos de vértices y aristas denotados por $V(G)$ y $E(G)$, respectivamente, y $|V(G)| = n$. Se define una coloración válida de G como una aplicación $C : V(G) \to \{1, 2, ..., n\}$ que identifica a $C(i)$ como el color del vértice $i$, de forma tal que dos vértices adyacentes no tengan el mismo color, es decir, $(i, j) \in E(G) \Rightarrow C(i) \neq C(j)$.

Una coloración válida de una gráfica $G$ con $k$ colores define una partición del conjunto de vértices en subconjuntos $V_1, V_2, ..., V_k$ , donde $V_j$ denota al conjunto de vértices que tienen asignado el color $j$. Claramente, cada $V_j$ es un conjunto independiente, al cual se le llama clase de color $j$ o clase cromática $j$. Se dice que una $k$-coloración válida de $G$ es una coloración válida que no utiliza más de $k$ colores, y una gráfica es $k$-coloreable si admite una $k$-coloración válida. Al mínimo valor $k$ tal que $G$ es $k$-coloreable se le llama número cromático de la gráfica y se denota por $\chi(G)$.

En una gráfica $G$, el Problema de Coloración Mínima (PCM) busca una coloración válida de $G$ que no utilice más de $\chi(G)$ colores.

En [5] y [7] se presenta el concepto de rigidez de una $k$-coloración, el cual establece que dadas dos gráficas complementarias $G$ y $\overline{G}$, y una función de peso $p : E(\overline{G}) \to \mathbb{R}$, la cual asigna una penalización a cada arista de $\overline{G}$, la rigidez de una $k$-coloración $C^k$ de $G$, denotada $R(C^k)$ es la suma de los pesos de las aristas de $\overline{G}$ que unen vértices de la misma clase cromática, esto es

$$R(C^k) = \sum_{(i,j)\in E(\overline{G}), C^k(i)=C^k(j)} p_{ij}$$

Con base en la definición anterior, Yáñez y Ramírez proponen el Problema de Coloración Robusta (PCR), el cual consiste en determinar una $k$-coloración $C^k$ de $G$ que sea válida y tenga mínima rigidez, *i.e.*, $R(C^k_R) = \min_{C^k} R(C^k)$; prueban además que este problema es $NP$-*duro*, por lo que es necesario utilizar métodos heurísticos para obtener buenas soluciones en un tiempo razonable.

El PCR es una generalización del PCM, lo que permite modelar problemas con restricciones adicionales a las que tienen los problemas modelados por el PCM. Aquí el objetivo no es minimizar el número de colores utilizados en la coloración de los vértices, lo que interesa es que una solución del problema sea lo más estable posible, en el sentido de que al añadir aristas a la gráfica la colo-ración continúe siendo válida. El objetivo es pues,

minimizar la suma de las penalizaciones de las aristas complementarias cuyos extremos tienen el mismo color. En [5] y [7], se presentan algunas aplicaciones del PCR a ejemplos prácticos, tales como problemas de calendarización, determinación de conglomerados, coloración de mapas, etc.

Los primeros algoritmos para resolver el PCR fueron presentados en [5] y [7]; éstos son algoritmos de aproximación; siendo uno de enumeración parcial, y otro un híbrido de un voraz con un genético, posteriormente en [6], Ramírez *et al.* proponen algoritmos de búsqueda tabú y recocido simulado que mejoraron las soluciones encontradas hasta entonces.

En la sección 2 se describe el algoritmo de búsqueda dispersa, en la sección 3 se definen los parámetros y las diferentes etapas a seguir del método de búsqueda dispersa para resolver el PCR, en la sección 4 se da el algoritmo de búsqueda dispersa para el PCR, en 5 se muestran los resultados obtenidos y en 6 se dan algunas conclusiones.

## 2 Búsqueda dispersa

Búsqueda Dispersa (en inglés, *Scatter Search*) es una técnica que cae dentro de la familia de algoritmos evolutivos; la idea original de esta técnica fue propuesta por primera vez en 1963 y 1965 por Fred Glover, pero hasta 1998, en [1], esta técnica tomó su forma definitiva. Búsqueda Dispersa construye nuevas soluciones por medio de la combinación de otras soluciones, pero difiere de otros algoritmos evolutivos, en el tamaño de la población usada, aquí es pequeña (entre 10 y 20 soluciones), y está fuertemente basada en el hecho que la calidad es mejor que la cantidad de las soluciones. Búsqueda Dispersa se basa en combinar soluciones que aparecen en el llamado conjunto de referencia. Este conjunto almacena las "buenas" soluciones que se han ido encontrando durante el proceso de búsqueda. Es importante destacar que el significado de buena no se restringe a la calidad de la solución, sino que también se considera la diversidad que esta aporta al conjunto de referencia.

### 2.1 Principales caraterísticas de la búsqueda dispersa

Las principales características de la metodología de la búsqueda dispersa son:

- Una cantidad pequeña de soluciones de alta calidad (llamadas soluciones "élite"), donde se guarda la información de las carac-terísticas de las soluciones óptimas. En la mayoría de las aplicaciones no sobrepasa de 20.

- Se requiere de mecanismos para explorar nuevas soluciones más allá de los óptimos locales. En este esquema es importante la incorporación de un proceso heurístico que combine soluciones para obtener nuevas soluciones.

- El uso simultáneo de soluciones múltiples, es una parte clave para obtener nuevas soluciones que contengan la información clave de la unión de soluciones élite.

## 2.2 Componentes de la Búsqueda Dispersa

Búsqueda Dispersa consta básicamente de cinco elementos o métodos que se describen a continuación:

1. **Generador de Soluciones Diversas**. Este elemento se basa en generar un conjunto $P$ de soluciones diversas (alrededor de 100) del que se extrae un subconjunto pequeño (entre 10 y 20) que se denomina conjunto de referencia. La generación se puede dar aleatoriamente o usando una solución inicial como semilla y a partir de allí tomar reglas deterministas para generar nuevas soluciones.

2. **Método de Mejora**. Típicamente se trata de un método de búsqueda local que mejora las soluciones, tanto del conjunto de referencia como las combinadas, antes de estudiar su inclusión en el conjunto de referencia. Si en la etapa de diversificación o de combinación de soluciones se generan soluciones infactibles, en esta etapa de mejoramiento se deben hacer factibles. El *método de mejora* transforma una o más soluciones en una o más soluciones mejoradas. Este método idealmente debe ser rápido (por ejemplo búsqueda local o alguna técnica glotona son las más comunes).

3. **Actualización del Conjunto de Referencia (*RefSet*)**. Este método crea el conjunto de referencia y lo actualiza después de cada generación de nuevas soluciones, elimina los elementos redundantes y preserva alguna diversidad dentro de las 10 a 20 soluciones que se manejan.

4. **Método de Generación de Subconjuntos**. Produce subconjuntos de soluciones como una base para crear soluciones combinadas. Generalmente el conjunto de referencia $RefSet$ se divide en dos: soluciones de calidad y soluciones diversas. Este método especifica la forma en que se seleccionan los subconjuntos para aplicarles el método de combinación.

5. **Método de Combinación de Soluciones**. Transforma un subconjunto dado de soluciones producido en el *Método de Generación de Subconjuntos* en una o más soluciones combinadas. Este método define la manera en la cual las soluciones se mezclan para crear nuevas soluciones. La solución o soluciones que se obtienen de esta combinación pueden ser inmediatamente introducidas al conjunto de referencia o almacenadas temporalmente en una lista hasta terminar de realizar todas las combinaciones.

En el Algoritmo 1 se muestra cómo interactúan los elementos descritos anteriormente. Primero se inicializa el conjunto $P$ igual al vacío. Posteriormente se aplica de manera iterativa el *Generador de Soluciones Diversas* y el *Método de Mejora* para generar tantas soluciones $x_i^*$ hasta completar el tamaño del conjunto $P$, previamente establecido, teniendo cuidado de no incorporar soluciones repetidas. A partir del conjunto $P$ se aplica el Método de Actualización del Conjunto de Referencia para generar un conjunto reducido denotado por $RefSet$ de soluciones de calidad y diversas. Terminada esta etapa, se comienza la

parte iterativa. Primero se generan subconjuntos del conjunto de referencia $RefSet$ por el *Método de Generación de Subconjuntos*, de cada subconjunto generado se obtienen nuevas soluciones por el *Método de Combinación* y se mejora cada solución con el *Método de Mejora*. Este proceso continúa hasta que se exploran todos los subconjuntos disponibles. Finalmente a todas las nuevas soluciones generadas y mejoradas se les aplica el *Método de Actualización del Conjunto de Referencia*. Si en esta etapa del algoritmo no se incorpora ninguna nueva solución al conjunto de referencia, el proceso termina. En caso contrario, nuevamente se aplica el *Método de Generación de Subconjuntos* y el proceso continúa hasta que el conjunto de referencia no cambia.

Algoritmo 1: Algoritmo de Búsqueda Dispersa

---

**inicio (Algoritmo 1)**
  $P \leftarrow \emptyset$ (Se inicia con la población $P$ vacía).
  **mientras** $P$ no alcance el número de elementos especificado **hacer**
    Construya una solución nueva $x$ con el *generador de soluciones diversas*.
    Aplique el *método de mejora* a $x$ para obtener $x^*$.
    **si** $x^* \notin P$ **entonces** incluir $x^*$ a $P$.
  A partir de $P$, construya el conjunto de referencia $RefSet$ aplicando la *actualizacion al conjunto de referencia*.
  **mientras** $RefSet$ contenga elementos nuevos **hacer**
    **para** cada subconjunto $i$ construído al aplicar el *método de generación de subconjuntos* a $RefSet$ **hacer**
      Use el *Método de combinación de soluciones* con $i$ para obtener $x_i$.
      Aplique el *Método de Mejora* a $x_i$ para obtener $x_i^*$.
    Aplique la *actualización al conjunto de referencia* a todas las soluciones $x_i^*$ para actualizar $RefSet$.
**fin**

---

## 3 Búsqueda dispersa para el problema de coloración robusta

En esta sección se describen los cinco elementos de la técnica de Búsqueda Dispersa usados en este trabajo:

1. *Generador de Soluciones Diversas.* Se diversifica de tres formas diferentes:

   (a) Se inicia en un vértice $v_i$ de la gráfica, (para $i = 1, .., n$) y se colorea con el color $j$ para $1 \leq j \leq k$ donde $k$ es el número total de colores, a los vértices

adyacentes a $v_i$ se les asigna el color siguiente, sino han sido coloreados (en caso que el color sea $k$ el siguiente color se toma igual a 1).

(b) Se selecciona una coloración inicial de manera aleatoria (esto significa que cada vértice tiene una probabilidad $1/k$ de tener cualquier color, donde $k$ es el número de colores permitidos). Este paso permite soluciones iniciales que en promedio tienen un número equilibrado de colores y presentan una buena diversidad en las soluciones iniciales.

(c) Inversión de una coloración. Se toma una coloración obtenida por alguno de los dos métodos anteriores y se intercambian los colores. El color del vértice 1 se intercambia con el del vértice $n$ , el color del vértice 2 con el $n-1$ y así sucesivamente.

En general, esta etapa no garantiza que la asignación de colores a los vértices cumpla con que dos vértices adyacentes no tengan el mismo color, es decir, si la arista $(i,j) \in E(G)$, no necesariamente cumple con que $C(i) \neq C(j)$.

2. *Método de Mejora*. Para la etapa de mejoramiento se proponen dos métodos:

(a) Para mejorar localmente las soluciones generadas se usa la técnica de hacer un cambio de color en cada vértice. La función objetivo por minimizar es *f(c)=I(c)+R(c)*, donde *I(c)* es el número de aristas incompatibles y *R(c)* es la rigidez de la coloración. Para cada vértice de la gráfica coloreada con el color $j$, se propone cambiar al color $l$ con $l \neq j$ y $l = 1, 2, ..., k$. Si al cambiar este color por el actual, se mejora la solución (es decir, *f(c)* disminuye), entonces se acepta el cambio, si no permanece con el color actual.

(b) Se hace un conteo de la frecuencia de cada color en una solución (En [7] se muestra que las mejores soluciones tienen aproximadamente la misma frecuencia de cada color). Si existe algún color que su frecuencia esté por encima de la frecuencia promedio $n/k$, entonces se tratan de colorear algunos vértices con el color menos frecuente en la solución de tal manera que los cambios mejoren la solución actual.

3. *Actualización del Conjunto de Referencia* $RefSet$. Se consideraron tres características para formar y actualizar el conjunto de referencia $RefSet$:

(a) Si se está creando un *RefSet* inicial, se toman las mejores soluciones del Generador de Soluciones Diversas para crear el $RefSet$, para generaciones subsecuentes, se eliminan los candidatos con la misma rigidez y mismo número de aristas incompatibles, y con los elementos restantes seguimos los puntos siguientes:

(b) Se selecciona el 70% de los elementos de $RefSet$ con coloraciones válidas de menor rigidez (considerando los mejores valores del $RefSet$ anterior, en generaciones subsecuentes). Es decir, incluimos en este paso las soluciones válidas y de buena calidad.

(c) El 30 % restante del $RefSet$ (diversidad) se crea con soluciones que tienen uno o dos vértices mal coloreados, *i.e.*, son coloraciones no válidas y con rigidez más pequeña.

(d) Sino se completa el tamaño del conjunto de referencia con los dos pasos anteriores, entonces se seleccionan el resto de las soluciones con el menor número de vértices en conflicto y menor rigidez.

4. *Método de Generación de Subconjuntos.* Se tomó el tamaño del conjunto de referencia igual a $b = 10$ elementos. Se generaron todos los subconjuntos posibles de tamaño 2 para el proceso de combinación; si se tienen $b$ elementos en el conjunto de referencia $RefSet$ entonces se generan $\frac{b(b-1)}{2}$ subconjuntos de parejas $([1,2],[1,3],...,[1,b],[2,3],\ [2,4],...,[1,b],...,[b-1,b])$. Únicamente se toman los mejores 4 elementos en el conjunto $RefSet$ para generar subconjuntos de tres elementos, de tal manera que se generan 4 posibles subconjuntos con 3 elementos. Se tomó esta decisión, ya que todas las combinaciones posibles con subconjuntos de tres elementos son $\frac{b(b-1)(b-2)}{6}$, muchos más elementos que los de tamaño 2, y de acuerdo con [4] es mejor utilizar más tiempo de cálculo en probar las combinaciones con dos elementos.

5. *Método de Combinación de Soluciones.* La mezcla de soluciones se hizo definiendo una secuencia de inclusión al azar. Por ejemplo para una gráfica que tiene los vértices (a,b,c,d,e,f) si queremos mezclar las coloraciones (2,3,1,1,3,2) y (1,2,3,2,1,3), vamos incluyendo aleatoriamente elementos de una secuencia en la otra, por ejemplo, para la secuencia aleatoria f,c,a,b,e,d, si la inclusión la hacemos en 5 pasos tendríamos:

| | |
|---|---|
| (2,3,1,1,3,2) | Solución Inicial |
| (2,3,1,1,3,**3**) | Paso 1 |
| (2,3,**3**,1,3,**3**) | Paso 2 |
| (**1**,3,**3**,1,3,**3**) | Paso 3 |
| (**1**,**2**,**3**,1,3,**3**) | Paso 4 |
| (**1**,**2**,**3**,1,**1**,**3**) | Paso 5 |
| (**1**,**2**,**3**,**2**,**1**,**3**) | Solución Final |

A la coloración obtenida en cada paso se le aplica el método de mejora. Si el número de vértices entre el número de pasos + 1 no da un número entero, como en el ejemplo, simplemente se hace un redondeo y se toma esa cantidad de elementos por paso.

# 4 Algoritmo de búsqueda dispersa para el PCR en pseudo código

Algoritmo 2: Algoritmo de búsqueda dispersa para el PCR.

---

**Inicio (Algoritmo 2)**
**Paso 1:** *Generador de Soluciones Diversas.*
De $i = 1$ a 100
Genere una solución $x_i$ aleatoria.
Use el Método de Mejora en $x_i$ para obtener $x_i^*$.
Si $x_i^* \notin P$ incluya la solución $x_i^*$ en $P$.
**Paso 2:** *Actualización del Método de Mejora.*
Seleccione en $P$ el 70% de las soluciones con una coloración válida y menor rigidez.
Seleccione en $P$ el 30% de las soluciones con 2 o más vértices mal coloreados y menor rigidez.
**Paso 3:** *Método de Generación de Subconjuntos.*
Considere todos los subconjuntos de dos elementos de $RefSet$.
Considere los subconjuntos de tres elementos formados a partir de los mejores cuatro elementos de $RefSet$.
**Paso 4:** *Método de Combinación de Soluciones.*
Para cada subconjunto de dos elementos de $RefSet$.
para $j = 1$ a 5
Combine los elementos de las dos soluciones para obtener una nueva solución.
Use el Método de Mejora en la nueva solución.
Guarde solución mejorada en $BigSet$.
Para cada subconjunto de tres elementos de $RefSet$.
Combine los tres elementos con probabilidad de 1/3.
Use el *Método de Mejora* en la nueva solución.
Guarde la solución mejorada en $BigSet$.
Si en $BigSet$ aparecen soluciones mejoradas o más variadas que las que tiene actualmente $RefSet$, regrese al Paso 2, de otra manera **finalice**.

---

# 5 Resultados

En [7], se generan gráficas aleatoriamente en donde se usa una densidad de 0.5 y tienen entre 10 a 15 vértices; allí también se encontraron buenas soluciones pero únicamente la mitad de ellas fueron óptimas de acuerdo a un modelo exacto 0-1. En este artículo se consideran gráficas similares y siempre se obtienen los valores óptimos para este rango de vértices. Se puede considerar que el algoritmo explicado en la sección anterior obtiene

mejores soluciones que las presentadas en [7].

En [6] se sugieren algunas instancias específicas. En este trabajo se tomaron las mismas instancias, y los resultados comparativos se muestran en la Tabla 1, en donde $n$ es el número de aristas, $k$ es el número de colores usados para colorear la gráfica. El asterisco (*) significa que se encontró la solución óptima con el modelo 0-1 (usando CPLEX).

En la Tabla 1 se observa que el Algoritmo de Búsqueda Dispersa propuesto, encuentra siempre las mejores soluciones. Las soluciones óptimas siempre se obtuvieron en todos los casos donde se conoce. Para instancias de mayor tamaño, en donde la solución óptima se desconoce, siempre se encontró la mejor solución conocida o se mejoró.

| Instancia | | | Rigidez en una solución válida | | | | |
|---|---|---|---|---|---|---|---|
| $G_{n,0.5}$ | $n$ | $k$ | Enumer. Parcial | Búsqueda Tabú | Recocido Simulado | Modelo 0-1 | Búsqueda Dispersa |
| al10 | 10 | 4 | 4.2386* | 4.2386* | 4.2386* | 4.2386* | 4.2386* |
| al10 | 10 | 5 | 1.8716* | 1.8716* | 1.8716* | 1.8716* | 1.8716* |
| al11 | 11 | 4 | 3.4450* | 3.4450* | 3.4450* | 3.4450* | 3.4450* |
| al11 | 11 | 5 | 2.4988 | 2.0227* | 2.0227* | 2.0227* | 2.0227* |
| al12 | 12 | 4 | 5.0815* | 5.0815* | 5.0815* | 5.0815* | 5.0815* |
| al12 | 12 | 5 | 3.8561 | 2.9041* | 2.9041* | 2.9041* | 2.9041* |
| al13 | 13 | 5 | 4.8958 | 4.0440* | 4.0440* | 4.0440* | 4.0440* |
| al13 | 13 | 6 | 2.7845 | 2.2765* | 2.2765* | 2.2765* | 2.2765* |
| al14 | 14 | 5 | 6.4479 | 5.3659 | 5.9756 | 5.2836* | 5.2836* |
| al14 | 14 | 6 | 3.7227 | 2.9387* | 2.9387* | 2.9387* | 2.9387* |
| al15 | 15 | 5 | 8.3658 | 6.0683 | 5.6376* | 5.6376* | 5.6376* |
| al15 | 15 | 6 | 5.0757 | 3.5592* | 3.5592* | 3.5592* | 3.5592* |
| al20 | 20 | 7 | 8.6477 | 6.0236 | 4.9557 | — | 4.9557 |
| al20 | 20 | 8 | 5.3147 | 3.9421 | 3.2285 | — | 3.2285 |
| al25 | 25 | 9 | 7.0819 | 6.7866 | 5.5344 | — | 5.2584 |
| al25 | 25 | 10 | 5.5987 | 4.8113 | 3.7923 | — | 3.7368 |
| al30 | 30 | 10 | 10.5818 | 10.1346 | 7.6417 | — | 6.8881 |
| al30 | 30 | 11 | 8.5675 | 7.4842 | 4.7623 | — | 4.7623 |

Tabla 1: Rigidez encontrada por los diferentes algoritmos.

## 6 Conclusiones

En este artículo se desarrolló un algoritmo evolutivo denominado de Búsqueda Dispersa y se probó en las instancias usadas en trabajos previos. Los resultados obtenidos con este algoritmo, son mejores que los previamente obtenidos con enumeración parcial, búsqueda tabú y recocido simulado. El Algoritmo de búsqueda dispersa presentado en este artículo siempre obtiene las soluciones óptimas en las instancias donde se conoce el valor óptimo y encuentra las mejores soluciones conocidas, o las mejora, en aquellas instancias donde no se conoce el óptimo.

## Referencias

[1] Glover, F. (1998) "A template for scatter search and path relinking", in: J.K. Hao, E. Lutton, E. Ronald, M. Shoenauer & D. Snyers (Eds.) *Artificial Evolution, Lecture Notes in Computer Science*, 1363, Springer: 13–54.

[2] Glover, F.; Laguna, M.; Martí, R., (2004) "Scatter search and path relinking: foundations and advanced designs", in: G. Onwubolu (Ed.) *New Optimization Techniques in Enginneering*, Springer Verlag.

[3] Glover, F.; Laguna, M.; Martí, R. (2004) "New ideas and applications of scatter search and path relinking", in: G. Onwubolu (Ed.) *New Optimization Techniques in Enginneering*, Springer Verlag.

[4] Laguna, M.; Armentano, V. (2004) "Lessons from applying and experimenting with scatter search", in: C. Rego & B. Alidaee (Eds.) *Adaptive Memory and Evolution: Tabu Search and Scatter Search*,

[5] Ramírez Rodríguez, J. (2001) *Extensiones del Problema de Coloración de Grafos.* Tesis de Doctorado, Universidad Complutense de Madrid, Madrid, España.

[6] Ramírez Rodríguez J.; Gutiérrez Andrade, M.A.; López Bracho, R.; Yáñez Gestoso, J., (2004) "Heuristics for the robust coloring problem", Preprint, Universidad Autónoma Metropolitana, México.

[7] Yáñez, J., Ramírez, J. (2003) "The robust coloring problem", *European Journal of Operational Research*, **148**(3): 546–558.