Revista de Matemática: Teoría y Aplicaciones 2016 **23**(2) : 421–444
cimpa – ucr issn: 1409-2433 (Print), 2215-3373 (Online)

A HYBRID ALGORITHM FOR THE ROBUST GRAPH COLORING PROBLEM

UN ALGORITMO HÍBRIDO PARA EL PROBLEMA DE COLORACIÓN ROBUSTA DE GRÁFICAS

Roman Anselmo Mora-Gutiérrez*
Javier Ramírez-Rodríguez† Eric A. Rincón-García‡
Antonin Ponsich§ Ana Lilia Laureano-Cruces¶

Received: 30 Sep 2014; Revised: 12 Apr 2016;
Accepted: 28 Apr 2016

*Universidad Autónoma Metropolitana-Azcapotzalco, Departamento de Sistemas, Av. San Pablo 180, Colonia Reynosa Tamaulipas, Ciudad de México, C.P. 02200, Mexico. E-Mail: ing.romanmora@gmail.com

†Misma dirección que/*Same address as*: R. A. Mora-Gutiérrrez.
E-Mail: jararo@correo.azc.uam.mx

‡Misma dirección que/*Same address as*: R. A. Mora-Gutiérrrez.
E-Mail: rigaeral@correo.azc.uam.mx

§Misma dirección que/*Same address as*: R. A. Mora-Gutiérrrez.
E-Mail: aspo@correo.azc.uam.mx

¶Misma dirección que/*Same address as*: R. A. Mora-Gutiérrrez.
E-Mail: clc@azc.uam.mx

Abstract

A hybrid algorithm which combines mathematical programming techniques (Kruskal's algorithm and the strategy of maintaining arc consistency to solve constraint satisfaction problem "CSP") and heuristic methods (musical composition method and DSATUR) to resolve the robust graph coloring problem (RGCP) is proposed in this paper. Experimental result shows that this algorithm is better than the other algorithms presented on the literature.

Keywords: metaheuristics; combinatorial optimization; integer programming.

Resumen

En este artículo se propone un algoritmo híbrido que combina técnicas de programación matemática (algoritmo de Kruskal y la estrategia de mantener consistencia de arcos para resolver el problema de satisfacción de restricciones) y métodos heurísticos (método de composición musical y DSATUR) para resolver el problema de coloración robusta de gráficas (RGCP). Resultados experimentales muestran que este algorimo da mejores resultados que otros presentados en la literatura.

Palabras clave: metaheurísticas; optimización combinatoria; programación entera.

Mathematics Subject Classification: 05C15.

1 Introduction

Graph theory has provided many models and efficient solution techniques for a variety of problems that have arisen in different contexts. One of such problems is to color the vertices of a graph [6, 30, 35, 36]. The graph coloring problem is, given a graph $G = (V, E)$ with sets of vertices and edges denoted by V and E, respectively and $|V(G)| = n$, to minimize the number of colors used for coloring the vertices of the graph such that no two adjacent vertices have the same color.

The problems that have been modeled as graph coloring problems are varied and range from those who only have historical or educational interest to applications in diverse areas, such as the eight queens problem, schoolgirls problem [2], course scheduling [6, 35, 36], cluster analysis [31], frequency assignment problem [30], map coloring [31], approach for image segmentation [13], design and operation of flexible manufacturing systems [7], etc.

Certain graph coloring problems can have requirements in the colorations, specifically, the possibility of converting the criterion to minimize the number of colors used in a restriction and seek new approaches of optimization that allow us to compare the various colorations obtained with a given number of colors.

It is of interest that a coloration is stable in the sense that when adding or changing edges in the graph, the coloring will continue to be valid. These considerations show that the problem of coloration is a restrictive model for this type of problems. Such comparations can be made if we associate a positive weight to each no edge and use the Robust Graph Coloring Problem (RGCP) introduced in [31].

Applications in examination timetabling problem, cluster analysis, uncertain resource constraint assignment problems in supply chain management and machine scheduling have been presented in [31, 33, 21, 19]. Mathematical formulations of the RGCP as a binary linear programming problem and quadratic assignment among others are proposed in [31].

Genetic algorithms are proposed in [31, 20], simulated annealing and tabu search algorithms are described in [12, 20, 11], a scatteer search procedure is presented in [17], other encoding schemes, neighborhood structures and search algorithms are proposed in [34], a local search procedure is proposed in [11], an ant algorithm is proposed in [18] and finally a branch-and-price algorithm is presented in [1].

In this paper we investigate the use of branch and cut to explore effectively suitable solution subspaces controlled by a simple external branching framework. The procedure is musical composition method where the neighborhoods are obtained through the introduction in the integer programming of constraints called local branching cuts.

The new solution strategy is approximate, though is designed to improve the heuristic, producing improved solutions.

The paper is organized as follows. Next section describes the robust graph coloring problem. In Section 3, the proposed algorithms are described. In Section 4 the experimental methodology is described and a computational analysis and comparisons on some instances of the RGCP is presented. Finally, in Section 5 some conclusions are given.

2 The robust graph coloring problem

Let $G = (V, E)$ be a graph, it is said that G is $k-colorable$ if each of its vertices can be assigned one of the k colors in such a way that adjacent vertices do not have the same color. The minimum value of k such that G is $k-colorable$ is the chromatic number of G denoted by $\chi(G)$.

Given complementary graphs $G = (V, E)$, $\overline{G} = (V, \overline{E})$ and a penalty function $P : \overline{E} \rightarrow \mathbb{R}$, the rigidity of a $k-coloring$ of G, denoted by $R(C)$ is the sum of the penalties of the edges of $\overline{G}$ that join vertices with the same color, *i.e*.

$$R(C) = \sum_{\{i,j\}\in\overline{E},\ C(i)=C(j)} p_{ij}. \tag{1}$$

Robust graph coloring problem. Find the $k-coloring$ of minimum rigidity, *i.e.*,

$$\begin{array}{lll} Min & R(C) & \\ s.t & \sum_{c=1}^{k} x_{ic} = 1 & \forall\, i \in \{1, \cdots, n\} \\ & x_{ik} + x_{jk} \leq 1 & \forall\, \{i,j\} \in E,\ \forall\, c \in \{1, \cdots, k\} \\ & x_{ic} + x_{jc} - 1 \leq y_{ij} & \forall\, \{i,j\} \in \overline{E},\ \forall\, c \in \{1, \cdots, k\} \\ & \sum_{i=1}^{n} x_{ic} \geq 1 & \forall\, c \in \{1, \cdots, k\}, \end{array} \tag{2}$$

where the decision variables are:

$$x_{ic} = \begin{cases} 1 & if\ C(i) = c \\ 0 & if\ C(i) \neq c \end{cases} \qquad \forall\, i \in \{1, \cdots, n\} \quad \forall\, c \in \{1, \cdots, k\}.$$

The following auxiliary variables are considered

$$y_{ij} = \begin{cases} 1 & \text{if } \exists\, c \in \{1, \cdots, k\} \text{ such that } x_{ic} = x_{jc} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall\, \{i,j\} \in \overline{E}.$$

The first set of constraints ensures that to each vertex is assigned a single color. The second set of constraints ensures that the coloring is valid. The third guarantee that if two vertices not connected by an edge have the same color then the penalty is added to the objective function and finally the last set of constraints, introduced in this paper, ensures that all colors are used.

3 Algorithms

Our hybrid, denoted as MP-MMC, combines mathematical programming techniques (Kruskal's algorithm and the strategy of the maintaining arc consistency for solving constraint satisfaction problems "CSP") with heuristic methods (musical composition and DSATUR). The general structure of our hybrid is shown in Algorithm 1. Then, a brief description of methods used by this is given.

The Kruskal's algorithm is a greedy algorithm, which was proposed in [15], used to find the minimum spanning tree for a connected weighted graph.

The strategy of maintaining arc consistency, denoted MAC, is an intelligent search algorithm, which use the information on the value that assume variables for generating backtracking on possible range of the other variables, for more details of the MAC we refer the reader to [4, 16, 32, 22].

The musical composition method, denoted MMC, which was presented in [25], is a metaheuristic, which mimic the social-creativity system involved in musical composition process. The MMC use a multiagent model, into social network. This social network is composed of a set of *Nc* vertices or agents (which are called composers), and a set E of edges or links (which are relationships among composers). In this model, each composer has for knowledge (a set of solutions, each solution is called "tune" and it is represented by an n-dimensional vector, which is composed by the values of decision variables) and a set of mechanisms and policies for interaction, based on this, each composer can communicate and exchange information with other composers. For more details of the MMC we refer the reader to [25, 26, 27, 28, 29]. The DSATUR algorithm, which was presented [5], is a sequential coloring algorithm with a dynamically established order of the vertices.

Algorithm 1: General algorithm, MP-MMC

Input: Instance characteristics to solve, a set θ of parameters
Output: The best found solution
begin
  Determine both a set T of edges contained in $\bar{G}$ and cost of T based on algorithm 2
  Create a society with N_c composers, with rules of interaction among composers.
  for *each composer into society* **do**
    $P_{i,\star,\star} \leftarrow$ a set of Ns solutions create based on algorithm 3.
    for *each solution into* $P_{i,\star,\star}$ **do**
      $evaluate_{i,j,\star} \leftarrow$ evaluation of the solution $P_{i,\star,\star}$ based on algorithm 6.
    end
  end
  while *termination criterion is not met* **do**
    Update the artificial society of composers.
    Exchange information between agents.
    for *each composer into society* **do**
      Generate and evaluate a new solution $tune_{new}$ accordance with algorithm 7
      Update $P_{i,\star,\star}$ (see Algorithm 10)
    end
    Build the solution set.
  end
end

Algorithm 1 is made up by six phases, which are (1) initializing the optimization process (from input to line 9) (2) interaction among agents into society (lines 11 and 12); (3) each composer generates a new solution (line 14); (4) update the $P_{i,\star,\star}$ of each composer (line 15); (5) building the set of solutions (line 17) and (6) repeating while the stopping criterion is not fulfilled (lines 10 to 19). The basic structure of the MP-MMC is similar to the general structure of the MMC. In the following sections, the steps of our hybrid are described in detail.

3.1 Initializing the optimization process

Initially, in this phase, characteristics of the instance to be solved and the value of the set working parameters ($\theta_{\textit{MP-MMC}}$) are introduced as input for our hybrid. The set $\theta_{\textit{MP-MMC}}$ is the same as the set $\theta_{\textit{MMC}}$ implied in MMC, which is composed by the maximum number of arrangement ($\max_{\textit{arrangement}}$), factor of genius both innovation (*ifg*) and change (*cfg*) factor of exchange among agents (*fcla*), number of composers (*Nc*) and number of chords that integrate the artwork (*Ns*).

Algorithm 2: Determine a set T of edges

Input: graph of the instance to solve
Output: T and $cost_T$

begin
  M represents a large positive number
  $|V|$ number of vertices of the graph to solve.
  $\bar{G}$ complementary graph with penalty.
  $\bar{E}$ set of edges of the $\bar{G}$
  for $i = 1 : |V| - 1$ **do**
    for $j = i + 1 : |V|$ **do**
      if $\{i, j\} \notin \bar{E}$ **then**
        Add $\{i, j\}$ to $\bar{G}$ with a cost M
        Add $\{j, i\}$ to $\bar{G}$ with a cost M
      end
    end
  end
  Use Kruskal's algorithm to find a minimum spanning tree T on $\bar{G}$
  Delete of T whatever edge with cost M
  $cost_T$ is the sum of the costs of edges T
end

After, in step 2, a set T of edges of the complementary graph is determined based on the algorithm 2.

Subsequently, in the MP-MMC algorithm is used the algorithm 3 for generate an initial set of solution ($P_{i,\star,\star}$) for the *i-th* composer. Algorithm 3 is based on DSATUR algorithm, however algorithm 3 is a random method that uses a peak of the number of vertices colored by *k*-th color.

Algorithm 3: Generate a set of solutions for each composer

Input: N_c,N_s, adjacency matrix (A), penalty matrix (C)
Output: P
begin
  $|V| \leftarrow$ number of vertices of the graph to solve
  $K \leftarrow$ number of colors used in the instance to solve
  $p \leftarrow \left\lceil \frac{|V|}{K} \right\rceil$
  $auxiliary$ is a zeros matrix of $(K \times |V|)$
  for $i = 1 : Nc$ **do**
    for $j = 1 : Ns$ **do**
      $P_{i,j,\star} \leftarrow$ solution looks for the algorithm 5
    end
  end
end

Algorithm 4: Determine $probability$ matrix

Input: $|V|$, adjacency matrix (A), penalty matrix (C)
Output: P
begin
  Built a opportunity cost matrix (OC) considered C
  $a_1 \leftarrow \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} A_{ij}$
  $a_2 \leftarrow \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} OC_{ij}$
  for $i = 1 : |V|$ **do**
    $probability\ _{i,1} \leftarrow \dfrac{\sum_{j=1}^{|V|} A_{i,j}}{a_1}$
    $probability\ _{i,2} \leftarrow \dfrac{\sum_{j=1}^{|V|} OC_{i,j}}{a_2}$
  end
end

Algorithm 5: Randomized Dsatur algorithm with peak

Input: $|V|, K, A, p$
Output: $new\_solution$
begin
$new\_solution$ is a zeros vector $(1 \times |V|)$
$p_{selection} \leftarrow probability_{*,1}$ obtained with algorithm 4
for $k = 1 : K$ **do**
if *there is a vertex not coloured* **then**
v^a is a not yet coloured vertex in V, which is randomly selected with base in $p_{selection}$
$new\_solution_{1,v^a} \leftarrow k$
$p_{selection_{v^a}} = 0$
$a_2 = \sum_{l=1}^{|V|} p_{selection_l}$
$a_1 = 1$
$auxiliary_{k,*} \leftarrow A_{v^a,:}$
while *$(a_2 \neq 0) \wedge (a_1 < p)$* **do**
$ap_{selection} = \emptyset$
for $l = 1 : |V|$ **do**
if *the $l - th$ vertex has not been coloured and $auxiliary_{k,l} = 0$* **then**
$ap_{selection_l} = probability_{l,1}$
else
$ap_{selection_l} = 0$
end
end
$aa_2 = \sum ap_{selection}$
if $aa_2 \neq 0$ **then**
$ap_{selection_l} = \frac{ap_{selection_l}}{aa_2} \quad \forall l = 1, \ldots, |V|$
v^s is a vertex in V, which is randomly selected with base in $ap_{selection}$
$new\_solution_{1,v^s} \leftarrow k$
$p_{selection_{v^s}} = 0$
$a_2 = \sum p_{selection}$
$a_1 = a_1 + 1$
$auxiliary_{k,*} \leftarrow auxiliary_{k,*} + A_{v^s,*}$
else
$a_1 = p$
end
end
else
v^a is a vertex in V, which is arbitrarily selected
$new\_solution_{1,v^a} \leftarrow k$
end
end
for $l = 1 : |V|$ **do**
if $new\_solution_{1,l} = 0$ **then**
$new\_solution_{1,l} = 1 + round(rand * (K - 1))$
end
end
$evaluation \leftarrow$ Evaluate $new\_solution$ based on algorithm 6
$new\_solution = new\_solution \cup evaluation$
end

Algorithm 6: Evaluate $j-th$ solution

Input: $T, new\_solution, C, A, K$
Output: $evaluation$
1 Determine the number of constraints not met $C(C)$ by solution
2 **if** $C(C) = 0$ **then**
3 $\quad R(C) \leftarrow$ is the value of the objective function in the solution
4 **else**
5 $\quad R(C) \leftarrow$ " $-$ "
6 **end**
7 a_1 is the number of edges in the solution content in T
8 $a_2 = \left\lceil \frac{|V|}{K} \right\rceil$
9 $a_3 = \left\lfloor \frac{|V|}{K} \right\rfloor$
10 **for** $k = 1 : K$ **do**
11 $\quad a_4$ is number of vertices of the $k-th$ color
12 $\quad Dif_k = \frac{1}{2} * \left(\frac{(a_4-a_2)^2}{K} + \frac{(a_4-a_3)^2}{K} \right)$
13 **end**
14 $a_5 = \sum_{k=1}^{|K|} Dif_k$
15 $T(C) = (|T| - a_1) + a_5 + R(C) * C(C) + 1)$
16 $evaluation = [C(C)\ R(C)\ T(C)]$

3.2 Interacting among agents

In this phase, composers exchange information according to a interaction policy specific. The interaction policy, used in this work, is "the composer i learns from the composer k, if there is a link between them and if the artwork of composer k has more desirable characteristics than the artwork of composer i". This policy was proposed in [25, 26, 27].

This phase is made up by two sub phases, which are 1) *updating the links between composers*, in which each composer can choose to modify his relation with other composer into society and 2) *information exchange procedure*, in this sub phase, each composer interacts with other composers into society so the $i-th$ composer takes and gives information with other composers into society, after, the ***i-th*** composer builds his matrix of the acquired knowledge ($ISC_{i,\star,\star}$). Routines employed by this phase were presented in [25, 26, 27].

3.3 Generating a new solution

In this phase, each composer will create a new tune utilizing his knowledge. This phase is divided into two sub phases: 1) *building the knowledge matrix* (KM). Each composer constructs his KM_i through of combining his $P_{i,\star,\star}$ with $ISC_{i,\star,\star}$ after, the $i-th$ composer assesses the fitness of each solution into KM_i. And 2) *creating a new solution*, in this sub phase, each composer generate a new solution based on both his KM_i and the algorithm 7.

The strategy of MMC for generating a new solution is used, in the step from 8 to 12 of the algorithm 6, to create a input for the strategy of maintaining arc consistency, which is contained in steps from 16 to 41.

Algorithm 7: Creating a new solution

Input: KM_i, ifg, cfg, A, p
Output: $new_{solution}$

begin
  for *each composer in society* **do**
    $FKM = \emptyset$
    $FKM_{1,:}$ is the best solution content in KM_i
    $FKM_{2,:}$ is a solution randomly take of KM_i with base in $fitness(KM_i)$
    $FKM_{1,:}$ should be different to $FKM_{2,:}$
    $FKM_{3,:}$ is a solution arbitrarily take of KM_i
    if $rand_1 \leq (1 - ifg)$ **then**
      $base$ is generated trough algorithm 8
    else
      $base$ is generated trough algorithm 5, but $ap_{selection_l}$ assigned $probability_{l,2}$
    end
    $\alpha \leftarrow$zeros matrix $(K \times |V|)$
    $new_{solution} = \emptyset$
    $\beta \leftarrow$ zeros vector $(1 \times K)$
    for $l = 1 : |V|$ **do**
      $a_1 \leftarrow base_{1,l}$
      if $(\alpha_{a_1,l} = 0) \bigwedge (\beta_{1,a_1} < p)$ **then**
        $new_{solution\ 1,l} \leftarrow a_1$
        $\alpha_{a_1,:} = \alpha_{a_1,:} + A_{a_1,:}$
        $\beta_{1,a_1} = \beta_{1,a_1} + 1$
      end
    end
    $\alpha_1 = \max\{\max_{k=1,2,\ldots,K;\forall l}(a_{kl})\}$
    $\alpha = \left\{ \left\lceil \frac{\alpha_{k,l}}{\alpha_1} \right\rceil \right\}$ $\forall l = 1, \ldots, |V|$ y $k = 1, \ldots, K$
    for $k = 1 : K$ **do**
      while $\beta_{1,k} < p$ **do**
        for $l = 1 : |V|$ **do**
          $visit_{1,l} = \begin{cases} 1 & if\ new_{solution\ 1,l} \neq 0 \\ 0 & if\ new_{solution\ 1,l} = 0 \end{cases}$
        end
        $\psi \longleftarrow \left\{ \left\lceil \frac{\alpha_{k,l} + visit_{1,l}}{2} \right\rceil \right\}$
        if $\sum_{l=1}^{|V|} \psi_{1,l} < |V|$ **then**
          γ is the index of a cell with value equal zero into vector $\alpha_{k,:}$
          $new_{solution\ 1,\gamma} \leftarrow k$
          $\alpha_{k,:} = \alpha_{k,:} + A_{\gamma,:}$
          $\beta_{1,k} = \beta_{1,k} + 1$
          $\alpha = \left\{ \left\lceil \frac{\alpha_{l,k}}{\alpha_1} \right\rceil \right\}$ $\forall l = 1, \ldots, |V|$ y $k = 1, \ldots, K$
        else
          $\beta_{1,k} = \beta_{1,k} + 1$
        end
      end
    end
    Call algorithm 9 on $new_{solution}$
  end
end

Algorithm 8: Creating a base

```
Input: KM_i, cfg, K, |V|
Output: base
begin
    for l = 1 : |V| do
        MH_{1,l} = max(KM_{i,⋆,l})
        MH_{2,l} = min(KM_{i,⋆,l})
    end
    base = ∅
    for l = 1 : |V| do
        if rand_2 < (1 − cfg) then
            a_1 = rand
            if a_1 ≤ 1/3 then
                base_l = FKM_{1,l}
            else
                if a_2 ≤ 2/3 then
                    base_l = FKM_{2,l}
                else
                    base_l = FKM_{3,l}
                end
            end
        else
            a_2 = rand
            if a_1 ≤ 1/2 then
                if a_2 ≤ 1/2 then
                    base_l = MH_{1,l}
                else
                    base_l = MH_{2,l}
                end
            else
                base_l = 1 + round(rand ∗ (K − 1))
            end
        end
    end
end
```

Algorithm 9: Making feasible to $new_{solution}$

Input: $new_{solution}$,$|V|$, K
Output: $new_{solution}$
begin
  a is a zero vector $(1 \times K)$
  for $l = 1 : |V|$ **do**
    if $new_{solution\ 1,l} = 0$ **then**
      $new_{solution\ 1,l} = round\,(1 + rand * (K - 1))$
    end
    $a_{1,new_{solution\ 1,l}} = a_{1,new_{solution\ 1,l}} + 1$
  end
  for $k = 1 : K$ **do**
    if $a_{1,k} = 0$ **then**
      $a_2 \leftarrow round(1 + rand(|V| + 1))$
      $a_3 \leftarrow new_{solution\ 1,a_2}\ new_{solution\ 1,a_2} \leftarrow k$
      $a_{1,k} = a_{1,k} + 1$
      $a_{1,a_3} = a_{1a_3} - 11$
    end
  end
end

3.4 Updating the $P_{i,\star,\star}$

In this phase, each composer makes a decision on either replacing or not the worst tune ($tune_{worst}$) in his score matrix $P_{i,\star,\star}$ with $new_{solution}$. The decision is based on the value of the objective function, so if the value of objective function of the $new_{solution}$ is better than the value of objective function of the $tune_{worst}$, then $new_{solution}$ replaces the $tune_{worst}$ in $P_{i,\star,\star}$. Algorithm 10 illustrates the procedure used for this purpose.

3.5 Building the set of solutions

In this phase, the MP-MMC selects the melody contained in artwork of every composer that achieves the best objective function value. The corresponding routine is shown in Algorithm 11.

Algorithm 10: Updating the $P_{i,\star,\star}$

Input: $new_{solution}$, $P_{i,\star,\star}$
Output: $P_{i,\star,\star}$
begin
    $tune_{worst}$ is the worst solution into $P_{i,\star,\star}$ depend on objective function
    $R(C)_{worst}$ is value of objective function of the $tune_{worst}$
    $C(C)_{worst}$ is the number of constrained no met by $tune_{worst}$
    $T(C)_{worst}$ is the number edge contend both $tune_{worst}$ and T
    $R(C)_{new}$ is value of objective function of the $new_{solution}$
    $C(C)_{new}$ is the number of constrained no met by $new_{solution}$
    $T(C)_{newt}$ is the number edge contend both $new_{solution}$ and T
    if $C(C)_{new} \leq C(C)_{worst}$ **then**
        if $R(C)_{new} \leq R(C)_{worst}$ **then**
            if $R(C)_{new} < R(C)_{worst}$ **then**
                Replacing of the $tune_{worst}$ for $new_{solution}$ in $P_{i,\star,\star}$
            else
                if $T(C)_{new} > T(C)_{worst}$ **then**
                    Replacing of the $tune_{worst}$ for $new_{solution}$ in $P_{i,\star,\star}$
                end
            end
        end
    end
end

Algorithm 11: Building the set of solutions

Input: $P_{\star,\star,\star}$,Nc
Output: $Solutions$
begin
    $Solutions \leftarrow \emptyset$ **for** $i : 1 : Nc$ **do**
        $Solution_i \leftarrow$ is the element, within $Pi, \star, \star$ with the best value based on $C(C)$, $R(C)$ and $T(C)$
    end
end

4 Experimental methodology and test problem

This section presents the computational experiments and associated results obtained by the *MP-MMC* algorithm on a set of instances of the robust graph colouring problem (RGCP), the general structure of the RGCP is shown in Equation (2).

4.1 Test problems

The characteristics of the instances used in this work are shown in the table 1, where n is the number of vertices in the graph and k is the number of colors. This instances were propose in [31] and these have been used in several works e.g: [31, 33, 17].

Table 1: Instances of the RGCP.

$G_{n,0,5}$	n	k	$G_{n,0,5}$	n	k
al(20)	20	7	al(60)	60	20
al(20)	20	8	al(60)	60	21
al(30)	30	10	al(70)	70	24
al(30)	30	11	al(70)	70	25
al(40)	40	14	al(80)	80	27
al(40)	40	15	al(80)	80	28
al(50)	50	17	al(90)	90	30
al(50)	60	18	al(90)	90	31

4.2 Design of the experimental test

The experiment was designed in order to analyze the performance of the MP-MMC on sixteen instances of the RCPs.

Taking into account the stochastic nature of the MP-MMC algorithm, 20 independent replications were performed for each instance. The time run and value of objective function were registered for each replication. Then for each instance and both objective functions the maximum, minimum, variance and standard deviation values were calculated.

The numerical result obtained by our hybrid was compared versus the results get by following algorithms:

- Tabu Search (*TS*) [9].
- Greedy randomized adaptive search procedure (*GRASP*) [8].
- Scatter Search (*SS*) [10, 24].

The information of these algorithms on the selected test set was taken from [12, 17].

With the aim of comparing the results obtained by the above mentioned metaheuristics on each instance, the results were normalized through the following equation:

$$f(x^{\text{normalized}-\alpha}) = \frac{f(x^{\text{method}-\alpha}) - f(x^\star)}{f(x^{\text{worst in }\beta}) - f(x^\star)} \tag{3}$$

where: $f(x^\star)$ is the value of the objective function at the global optimal point, $f(x^{method-\alpha})$ is the average value of the objective function found by metaheuristic α, $f(x^{\text{worst in }\beta})$ is the worst average of the objective function found by metaheuristics on test case β , and $f(x^{\text{normalized}-\alpha})$ is the normalized value of the objective function found by metaheuristic α.

The value of $f(x^{\text{normalized}-\alpha})$ ranges from 0 to 1. If $f(x^{\text{normalized}-\alpha})$ is close to 0, the value of $f(x^{\text{method}-\alpha})$ is near to $f(x^\star)$. If $f(x^{\text{normalized}-\alpha})$ is close to 1, the value of $f(x^{\text{method}-\alpha})$ is far from $f(x^\star)$.

Furthermore, a non-parametric Wilcoxon rank sum test was applied to the results obtained by *MMC* and the other tested heuristic algorithms. The null hypothesis is that data from two solution sets are independent: if the value returned by the test is $h = 1$, the null hypothesis is rejected with a 5% significance level, while $h = 0$ indicates a failure to reject the null hypothesis with a 5% significance level. Parameters *p* (standing for the symmetry and mean of the distribution) and *h* (which is the hypothesis test result) were computed from this statistical test.

4.3 Parameter setting for the *MP-MMC* hybrid

In the first tuning, an arbitrary set θ of parameters was fixed, later parameters $max_{arrangement}$, *Nc* and *Ns* were adjusted with the brute-force approach [3]. The Nc is expressed as a percentage λ of the $|V|$ (see equation 4) . The $max_{arrangement}$ were determined in function of the *Ns* through equation 5. The set θ obtained in this phase, was used as input for tuning of the *ifg*, the *cfg* and the *ifcla* parameters with a technique semi-factorial experimental design.

$$Nc = \lambda * |V| \tag{4}$$

$$\max_{arrangement} = Ns * \kappa. \tag{5}$$

Table 2: Parameter settings of MP-MMC.

Parameter	value
κ	1000
λ	0.3
Ns	5
ifg	0.2
cfg	0.1
$fcla$	0.1

In the semi-factorial experimental design, combinations generated by values $ifg : \{0.1, 0.2, 0.3, 0.4, 0.5\}$, $cfg : \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and $cfla : \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ were tested, so 180 experiment were tried out. Five repetitions were made for each experiment. Also in each repetition, the value of objective function ($f(x)$) was registered. Then, the mean squared error (*MSE*) was calculated, through equation 6, for each repetition:

$$MSE_i = \sum_{j=1}^{5} \frac{(f(x) - f(\bar{x}))^2}{5}. \quad (6)$$

The minimum value of the *MSE* was 0.035, which was get with $ifg = 0.2$, $cfg = 0.1$ and $cfla = 0.1$. In contrast, the maximum value of *MSE* was 32.96, which was found with $ifg = 0.3$, $cfg = 0.5$ and $cfla = 0.2$. In Table 2, the parameter setting is shown.

4.4 Experimental results and discussion

The MP-MMC was implemented in Matlab R2010a on a MacBookAir processing unit 1.8 GHz intel core i7.

The results obtained are structured in Table 3, which synthesize, for each instance the best (x_{best}), the worst (x_{worst}), the mean $\bar{x}$, the variance s^2 and standard deviation s, computed over 20 runs of the best objective function found by MP-MMC.

In Table 4 are shown 95% confidence intervals determined with bootstrap method on the mean.

A comparative of the best results obtained by *MMC*, *GRASP*, *TS* and *SS* is shown in Table 5 and Table 6 .

Table 3: Results obtained by MP-MMC.

n	k	x_{best}	x_{worst}	$\bar{x}$	s^2	s
20	7	6.9046	7.3472	7.0030	0.0136	0.1167
20	8	4.6934	4.8391	4.7379	0.0036	0.0603
30	10	7.5749	11.041	9.2173	1.1238	1.0601
30	11	5.889	6.6233	6.1184	0.0370	0.1925
40	14	7.149	8.3658	7.5801	0.1132	0.3364
40	15	5.6708	6.747	6.1286	0.1152	0.3395
50	17	8.8613	10.781	9.4673	0.2331	0.4828
50	18	7.0506	8.7703	7.6847	0.1946	0.4411
60	20	9.6732	12.033	10.7683	0.4981	0.7058
60	21	7.5521	9.1749	8.3152	0.2065	0.4544
70	24	10.395	17.16	11.5579	2.0758	1.4408
70	25	8.773	11.581	9.8447	0.3721	0.6100
80	27	10.884	20.375	13.8058	5.5948	2.3653
80	28	9.8818	19.367	11.4210	4.0702	2.01747
90	30	12.744	22.772	16.1659	6.1485	2.4796
90	31	11.702	20.925	14.3109	4.7573	2.1811

Table 4: Results boot strap test with $\alpha = 0.05$.

n	ks	Lower limit	Upper limit
20	7	6.9561	7.0451
20	8	4.7158	4.7684
30	10	8.7883	9.6566
30	11	8.0561	8.3987
40	14	7.4516	7.7449
40	15	5.9688	6.2570
50	17	9.2542	9.6508
50	18	7.4800	7.8490
60	20	10.4617	11.0847
60	21	8.1183	8.5176
70	24	11.1014	12.3329
70	25	9.6131	10.1025
80	27	12.7755	14.6737
80	28	10.7909	12.5324
90	30	15.0804	17.2922
90	31	13.3681	15.2844

Based on the previous result, we can say that MP-MMC generates the best results in 31.25 % of the instances. Also in 62.5% of the instances the MP-MMC produced the second bests results. Our heuristic is better than *TS* and *GRASP* in the most cases.

The results of the time run of the MP-MMC are shown in the Table 7.

Table 5: Comparative of results obtained by heuristics.

n	k	MMC	TS	GRASP	SS
20	7	6.9046	7.097	7.1423	6.9046
20	8	4.6934	4.771	4.6934	4.6934
30	10	7.5749	8.0623	7.5749	7.5749
30	11	5.889	6.0565	5.9318	5.889
40	14	7.149	7.1709	7.395	7.0837
40	15	5.6708	5.8173	6.3117	5.6708
50	17	8.8613	9.8259	8.9531	8.2587
50	18	7.0506	7.4966	7.1464	6.7164
60	20	9.6732	9.8331	9.9687	8.8676
60	21	7.5521	8.2181	8.143	7.238
70	24	10.395	11.1307	11.2388	9.2634
70	25	8.773	9.5478	9.2145	7.7048
80	27	10.884	11.1946	11.7512	9.9835
80	28	9.8818	10.5845	10.2631	8.5961
90	30	12.744	12.2832	13.4919	10.8911
90	31	11.702	11.3699	11.506	9.5008

Table 6: Comparative of normalizing results.

	Instances															
n	20	20	30	30	40	40	50	50	60	60	70	70	80	80	90	90
k	7	8	10	11	14	15	17	18	20	21	24	25	27	28	30	31
1	3	2	2	2	3	3	2	2	3	2	3	2	3	2	3	1
0.9									2	3	2	3				3
0.8	2													3		2
0.7									1				2		1	
0.6								3			1	1		1		
0.5													1		2	
0.4							1,3	1								
0.3				3	2					1						
0.2					1	2										
0.1																
0	1	3,1,4	3,1,4	1,4	4	1,4	4	4	4	4	4	4	4	4	4	4

where: 1 is MMC; 2 is TS; 3 is GRASP; 4 is SS.

Table 7: Time run obtained by MP-MMC.

n	k	$time_{best}$	$time_{worst}$	mean time	s^2	s
20	7	38.86	41.99	40.9195	0.5446	0.73796
20	8	37.46	42.41	38.8305	1.6859	1.2984
30	10	116.18	124.19	121.491	4.4845	2.1177
30	11	107.27	117.93	111.567	15.3845	3.9223
40	14	236.07	280.25	255.898	180.4200	13.4321
40	15	240.22	270.23	251.73	35.6636	5.971903
50	17	481.05	561.13	509.119	906.7228	30.1118
50	18	483.49	539.71	494.233	145.0870	12.0452
60	20	542.09	646.01	581.0425	1103.0155	33.2117
60	21	544.19	574.51	556.8515	67.1736	8.1960
70	24	1455.4	1710.4	1531.145	6033.4331	77.6752
70	25	1470	1721	1536.795	2370.7847	48.6907
80	27	1457	1731.8	1572.52	6668.0122	81.6579
80	28	1469.9	1732.2	1549.285	4006.8401	63.2996
90	30	3536	3843.9	3674.49	6948.9725	83.3605
90	31	2312	2672.2	2412.665	7383.1182	85.9251

5 Conclusions

In this paper, a hybrid between mathematical programming techniques and metaheuristics was presented, which was called MP-MMC. The numerical results illustrate that the MP-MMC has a higher capability to solve instances of the RCPs, so the MMC generates the best or second best results in 93.75% of the test instances.

Future works might focus on extending the use of the MP-MMC to solve larger instances of the RGCP. Also we must improve the structure of the MP-MMC for making it more effective.

References

[1] Archetti, C.; Bianchessi, N.; Hertz, A. (2014) "A branch-and-price algorithm for the robust graph coloring problem", *Discrete Applied Mathematics* **165**(11): 49–59.

[2] Berge, C. (1973) *Graphes et Hypergraphes*. Dunod, Paris.

[3] Birattari, M. (2009) *Tuning Metaheuristics: A Machine Learning Perspective*. Springer, Berlin.

[4] Brailsford, S.C.; Potts, C.N.; Smith, B.M. (1999) "Constraint satisfaction problems: Algorithms and applications", *European Journal of Operational Research* **119**(3): 557–581.

[5] Brélaz, D. (1979) "New methods to color the vertices of a graph", *Communications of the ACM* **22**(4): 251–256.

[6] Burke, E.; Jackson, K.; Kingston, J H.; Weare, R. (1997) "Automated university timetabling the state of the art", *The Computer Journal* **40**(9): 565–571.

[7] Chams, M.; Hertz, A.; de Werra, D. (1987) "Some experiments with simulated annealing for coloring graphs", *European Journal of Operational Research* **32**(2): 260–266.

[8] Feo, T.A.; Resende, M.G.C. (1995) "Greedy randomized adaptive search procedures", *J. of Global Optimization* **6**(2): 109–133.

[9] Glover, F. (1986) "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research* **13**(5): 533–549.

[10] Glover, F.; Laguna, M.: Martí, R. (2003) "Scatter search", in *Advances in Evolutionary Computing*, Springer Berlin Heidelberg: 519–537.

[11] Guo, S.; Ying, K.; Lim, A.; Wang, F. (2004) "A new neighborhood based on improvement graph for robust graph coloring problem", in: G.I. Webb & X. Yu (Eds.) *AI 2004: Advances in Artificial Intelligence*, LNAI 3339, Springer, Berlin: 636–645.

[12] Gutiérrez-Andrade, M.A.; Lara-Velázquez, P.; López-Bracho, R.; Ramírez-Rodríguez, J.(2010) "Heuristics for the robust coloring problem", *Revista de Matemática: Teoría y Aplicaciones* **18**(1): 137–147.

[13] Gómez, D.; Montero, J.; Yáñez, J.; Poidomani, C. (2007) "A graph coloring approach for image segmentation", *Omega* **35**(2): 173–183.

[14] Kong, Y.; Wang, F.; Lim, A.; Guo, S. (2003) "A new hybrid genetic algorithm for the robust graph coloring problem", in: T.D. Gedeon & L.C.C. Fung (Eds.) *Lecture Notes in Artificial Intelligence*, Vol. 2903, Springer: 125–136.

[15] Kruskal, J.B. (1956) "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proceedings of the American Mathematical Society* **7**(1): 48–50.

[16] Kumar, V. (1992) "Algorithms for constraint-satisfaction problems: A survey", *AI magazine* **13**(1): 32–44.

[17] Lara-Velázquez, P.; Gutiérrez-Andrade, M.A.; Ramírez-Rodríguez, J.; López-Bracho, R. (2005) "Un algoritmo evolutivo para resolver el problema de coloración robusta", *Revista de Matemática: Teoría y Aplicaciones* **12**(1-2): 111–120.

[18] Laureano-Cruces, A.L.; Ramírez-Rodríguez, J.; Hernández-González, D.E.; Méndez-Gurrola, I.I. (2011) "An ant colony algorithm for the robust graph coloring problem", *ICGST AIML-11 Conference*, Dubai: 57–60.

[19] Leus, R.; Herroelen, W. (2007) "Scheduling for stability in single-machine production systems", *Journal of Scheduling* **10**: 223–235.

[20] Lim, A.; Wang, F. (2004) "Metaheuristics for robust graph coloring problem", *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*: 514–518.

[21] Lim, A.; Wang, F. (2005) "Robust graph coloring for uncertain supply chain management", *Proceedings of the 38th Hawaii International Conference on System Sciences*: 1–10.

[22] Liu, Z.(1998) *Algorithms for Constraint Satisfaction Problems (CSPs)*. Master of Mathematics in Computer Science, University of Waterloo, Canada.

[23] Maniezzo, V.; St́utzle, T.; Voβ, S. (Editors) (2009) *Matheuristics. Hybridizing Metaheuristics and Mathematical Programming*, Volume 10. Springer, New York.

[24] Martí, R.; Laguna, M. (2003) "Scatter search: Diseño básico y estrategias avanzadas", *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* **7**(19): 123–130.

[25] Mora-Gutiérrez, R.A.; Ramírez-Rodríguez, J.; Rincón-García, E. (2012) "An optimization algorithm inspired by musical composition", *Artificial Intelligence Review*: 1–15.

[26] Mora-Gutiérrez, R.A.; Ramírez-Rodríguez, J.; Rincón-García, E.; Ponsich, A.; Herrera, O. (2012) "An optimization algorithm inspired by social creativity systems", *Computing* **94**(11): 887–914.

[27] Mora-Gutiérrez, R.A. (2013) *Diseño y Desarrollo de un Método Heurístico Basado en un Sistema Socio-Cultural de Creatividad para la Resolución de Problemas de Optimización Continuos no Lineales y Diseño de Zonas Electorales*. Tesis de Doctorado, Universidad Nacional Autónoma de México, México.

[28] Mora-Gutiérrez R.A.; Rincón-García, E.A.; Ramírez-Rodríguez, J.; Ponsich, A.; Herrera-Alcántara, O.; Lara-Velázquez, P. (2013) "An optimization algorithm inspired by musical composition in constrained optimization problems", *Revista de Matemática: Teoría y Aplicaciones* **20**(2): 183–202.

[29] Mora-Gutiérrez, R.A.; Rincón-García, E.A.; Ramírez-Rodríguez, J.; Ponsich, A.; Herrera-Alcántara, O.; Lara-Velázquez, P. (2013) "Adaptation of the musical composition method for solving constrained optimization problems", *Soft Computing* **18**(10): 1931–1948.

[30] Pardalos, P.; Mavridou, T.; Xue, J. (1998) "The graph coloring problem: A bibliographic survey", in: D.Z. Du & P.M. Pardalos (Eds.) *Handbook of Combinatorial Optimization*, Vol 2, Kluwer, New York: 331–395.

[31] Ramírez-Rodríguez, J. (2001) *Extensiones del Problema de Coloración de Grafos*. Tesis de Doctorado, Universidad Complutense, Madrid, España. `http://eprints.ucm.es/4479/`, accessed March 26, 2014.

[32] Régin, J.C. (1994) "A filtering algorithm for constraints of difference in CSPs", *Proceedings of AAAI*, Vol. 1: 362–367.

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

[34] Wang, F.; Xu, Z. (2013) "Metaheuristics for robust graph coloring", *Journal of Heuristics* **19**(4): 529–548.

[35] de Werra, D. (1996) "Extensions of coloring models for scheduling purposes", *European Journal of Operations Research* **92**(3): 474–492.

[36] de Werra, D. (1996) "Some combinatorial models for course scheduling", in: E. Burke & P. Ross (Eds.) *Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer: 296–308.