Revista de Matemática: Teoría y Aplicaciones 2011 **18**(1) : 137–147
cimpa – ucr   issn: 1409-2433

HEURISTICS FOR THE ROBUST COLORING PROBLEM

HEURÍSTICAS PARA EL PROBLEMA DE COLORACIÓN ROBUSTA

Miguel Ángel Gutiérrez–Andrade[*]
Pedro Lara–Velázquez[†] Rafael López–Bracho[‡]
Javier Ramírez–Rodríguez[§]

Received: 18 Feb 2010; Revised: 14 Oct 2010; Accepted: 10 Nov 2010

[*]Departamento de Ingeniería Eléctrica, Universidad Autónoma Metropolitana - Iztapalapa, Avenida San Rafael Atlixco No. 186, Col. Vicentina, Del. Iztapalapa, 09340, México D. F. E-Mail: gamma@xanum.uam.mx

[†]Departamento de Sistemas, Universidad Autónoma Metropolitana - Azcapotzalco. Avenida San Pablo 180, Colonia Reynosa Tamaulipas, 02200 México D. F. E-Mail: pedro_lara@correo.azc.uam.mx

[‡]Misma dirección que/*same address as* P. Lara. E-Mail: rlb@correo.azc.uam.mx

[§]Departamento de Ingeniería Eléctrica, misma dirección que/*same address as* Lara. E-Mail: jararo@correo.azc.uam.mx

Abstract

Let G and $\bar{G}$ be complementary graphs. Given a penalty function defined on the edges of $\bar{G}$, we will say that the rigidity of a k-coloring of G is the sum of the penalties of the edges of $\bar{G}$ joining vertices of the same color. Based on the previous definition, the Robust Coloring Problem (RCP) is stated as the search of the minimum rigidity k-coloring. In this work a comparison of heuristics based on simulated annealing, GRASP and scatter search is presented. These are the best results for the RCP that have been obtained.

Keywords: graph coloring, robust coloring, heuristics.

Resumen

Sean G y $\bar{G}$ dos grafos complementarios. Dada una función de penalización en las aristas de $\bar{G}$, la rigidez de una k-coloración de G se define como la suma de las penalizaciones en las aristas de $\bar{G}$ cuyos vértices incidentes son del mismo color. Con base en la definición anterior, el Problema de Coloración Robusta (PCR) se define como la búsqueda de la k-coloración de rigidez mínima. Este trabajo realiza un estudio comparativo de varias técnicas heurísticas: Recocido Simulado, GRASP, y Búsqueda Dispersa. Los resultados aquí presentados son los mejores obtenidos para el PCR.

Palabras clave: coloración de grafos, coloración robusta, heurísticas.

Mathematics Subject Classification: 90C59, 78M32.

1 Introduction

Let G be a simple graph with sets of vertices and edges denoted by $V(G)$ and $E(G)$, respectively, and $|V(G)| = n$. G is said to be $k-colorable$ if to each of its vertices we can assign one of k colors in such a way that no two adjacent vertices have the same color. The minimum value of k that makes G $k-colorable$ is called the chromatic number of G and is denoted by $\chi(G)$. A coloring of a graph G with k colors defines a partition of the vertices set into subsets $V_1, V_2, ..., V_k$, where V_j denotes the set of vertices which have the color j assigned. Each V_j is an independent set, which is called color class j.

In a graph G, the Minimum Coloring Problem, MCP for short, searches a coloring of G that uses not more than $\chi(G)$ colors. Given complementary graphs G, $\bar{G}$ and a penalty function $P : E(\bar{G}) \to \Re$, the rigidity of a k-coloring C^k of G , denoted by $R(C^k)$ is the sum of the penalties of the edges of $\bar{G}$ that join vertices of the same chromatic class, i.e.

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

Robust Coloring Problem. Determine a k-coloring $C(k,R)$ of G, with minimum rigidity, i.e.,

$$C_R^k = \min_{C_k} R(C^k)$$

.

It was also proved (Yáñez and Ramírez, 2003) that this problem is NP-hard, which makes the use of heuristic methods necessary to obtain good solutions in a reasonable amount of time.

The objective is to minimize the sum of penalizations of the complementary edges whose extremes are equally colored. In Ramírez Rodríguez (2001) some practical examples are presented as applications of RCP, such as: scheduling problems, cluster analysis and maps coloring, that work also presents the first algorithms to solve the RCP one is based on a partial enumeration method, and the other is a hybrid of a greedy with a genetic algorithm.

New heuristics are presented in this work: in section 2 one based on simulated annealing, another one based on a GRASP algorithm and a scatter search algorithm; computational experiences are included in section 3; and some conclusions are in section 4. These algorithms show the best results obtained to date for the RCP.

2 Heuristics

All the heuristics proposed in this work use the following objective function:

$$\min f(S) = \sum_{\{i,j\}\in E(\bar{G}), C^k(i)=C^k(j))} p_{ij} + \mu \sum_{i=1}^{k} E(V_i).$$

The first part of the objective function is the sum of the penalizations of the complementary edges whose extremes are equally colored, and the second part during the search process, penalizes the not valid coloring. $|E(Vi)|$ is the number of edges in G with endpoints in set V_i, p_{ij} is the associated penalization of the complementary edge $\{i,j\} \in E(\bar{G})$, and $\mu = 100$.

Heuristics admits at the beginning and intermediate stages the possibility of working with not valid coloring, if this coloring produces a little accumulated rigidity for the complementary edges, so an intermediate solution produced for the heuristics for the coloring problem is a partition $S = (V_1, V_2, \ldots, V_k)$ of the vertices set in k subsets not necessarily independent.

$S' = (V_1', \ldots, V_k')$ is a neighbor solution of S if two V_i,V_j and $x \in V_i$ exist so that:

$$V_i' \;=\; V_i - \{x\}; V_j' = V_j \cup \{x\}$$

$$V_l' \;=\; V_l,\ \forall l \neq i, j$$

the criterion to generate S' from S consists of a random selection of an index i and an index j, and also a random selection of $x \in V_i$ to move it to V_j.

2.1 Tabu Search

This method can be cosidered as an iterative technique wich explores a set X of solutions of a problem (Glover, 1997), moving from a solution S to another solution S in the neighbourhood $N(S)$ of S. These movements are performed to reach a good solution evaluating an objective function $f(S)$ to minimize.

A key mechanism to explore memory in Tabu Search consists in classifying a subset of movements in the neighbourhood as prohibited or tabu, called tabu list. The classifications depend on the story of the search and normally are the most recent movements. Tabu constraints are not immune. When a tabu movement provides a better solution than any other found, its tabu classification can be eliminated; the conditions that allows such elimination are called aspiration criteria.

The basic step consists of analyzing all posible movement is carried out and the tabu list is updated. A termination rule has to be defined too, in general this consists of giving a maximum number of iterations, or stop when after a certain number of iterations, the solution does not improved.

The basic Tabu Search algorithms is:

```
1 Begin
2 Generate initial solution
3 for $i = 1$ to $niter$
4   Analyze neighbouring solutions: change two color class vertices.
5   Choose the best solution.
6   Randomly select a vertex and assign a different color to it in a
     random way
7 end {for}
8 End
```

The components of the tabu search algorithm proposed for the RCP are shown as follows. A feasible solution is, as in the simulated annealing method, a partition $S = (V_1, V_2, , V_k)$ of the set of vertices $V(G)$ in k color clases. Let $E(V_i)$ be the set of edges in G with both ends in V_i, p_{ij} the associated penalization of the complementary edge $i, j \in E()$, and $\mu = 100$.

The stop criterion used was a number of iterations determined. The tabu list was built bye the following way: whenever a vertex x changes form a color class, v_i to a color class v_j, to obtain a new solution, a pair $\{i, x\}$ is added to the tabu list, wich means that vertex x cannot be colored again with color i during certain number of iterations.

An aspiration criterion which establishes that if a movement, belonging to the tabu list, gives a new solution which is better to ther best solution found at that moment, then the tabu status is ignored and the movement is done.

An important characteristic of the proposed algorithm is the diversification of the search when the sizes of the color clases vary. The algorithm which finds the first feasible solution, tends to homogenize the color clases in relation to the number of elements, situation that the algorithm keeps in its general stage when applying the first criterion of change, when randomly chooses two vertices of different color and interchanges them; but modifies with the second criterion of change, when choosing randomly a vertex and assigning a different color in a random way.

2.2 Simulated Annealing

The standard Simulated Annealing algorithm is:

```
Begin
1 Generate an initial random coloring $i$.
2 Evaluate the cost function $f(S)$
3 Set $t = \sqrt{(V(G))}$ ; $r = \exp(d/t)$; $B = 0.95$ ; $a = 1.05$
4 While a new neighboring solution is accepted in a complete cycle
5   For $i = 1$ to $S$
6     Generate a neighboring solution $j$.
7     If $\Delta f = f(i) - f(j) > 0$, then $j$ becomes the new best solution $i$.
8     Otherwise, generate a random number U(0,1).
9     If $\exp(\frac{\Delta f}{t})$ is bigger than such random number then $j$ becomes
        the new best solution $i$.
10    Adjust the parameter: $r = a * r$
11  End {For}
12  Adjust the parameter: $t = B * t$
13 End {While}
End
```

With this algorithm we produce an initial solution painting each one of the graphs vertices with a random color (line 1). Because a property of function $f(S)$ (equation 1) is that, if we take relatively large values of μ ($\mu > 5$) we assure almost always the feasibility of the solution, at the cost of having solutions with a relatively high rigidity. On the other hand, if we take relatively small values of μ, we get the best values of rigidity, at the risk of having solutions that are not feasible. In this algorithm we found that a value of $\mu = 3$ gives the best results, for it gives feasible solutions in most cases with small rigidity (line 2).

The length r of each set, where the control t parameter remains constant, was considered as $r = \exp(d/t)$, where d is the extent required to escape from any local minimum; $d = 2$ was considered, as suggested in Chams (1987).

The control parameter t was considered equal to the square root of $|V(G)|$, that is $t = ((|V(G)|)^{1/2})$, and the value was decreasing by a factor $B < 1$ after r iterations, that is $t_{i+1} = B * t_i$, where the best value of B was considered near to 1 (0.95). The algorithm ends when no new neighboring solution is accepted in a complete cycle.

In the While cycle of the algorithm (lines 4 - 13) we are testing neighbors solutions and accepting them always if they are better than the actual solution (line 7); if not, when the value of exp $(\frac{\Delta f}{t})$ is less than a number with distribution U(0,1), the j solution becomes the best i solution (line 9). The parameters numbers of neighbors revised, r and actual temperature, t are reset. The constants a, B were set to the best empirical values found.

2.3 GRASP

GRASP Algorithm for the coloring problem is:

```
Begin
1 For $m = 1$ to $PoolSize$
2   Generate a sequence in which the vertices will be painted: $Sec(j)$
3   For $j = 1$ to $n$
4     Generate a sequence in which the colors will be tried: $SeqCol(i)$
5     For $i = 1$ to $k$
6       If $Col(Sec(j)) = SeqCol(i)$ is a valid coloring,
         then paint that vertex in that color
7     End {For i}
8     If $Col(j) = \phi$, paint the vertex in a random color.
9   End {For j}
10   While we have an improvement in coloring:
11     Generate a sequence for visiting vertices $Sec(j)$
12     For $j = 1$ to $n$
13       Generate a sequence for trying colors: $SeqCol(i)$
14       For $i = 1$ to $k$
15         If $Col(Sec(j)) = SeqCol(i)$, then reduce the number of wrongly
           colored vertices or rigidity, without worsening the coloring;
           and replace the solution.
```

```
16          End {For i}
17        End {For j}
18     End {While}
19 End {For m}
End
```

In the construction phase (lines 1 - 9 in Figure 4), the graph is painted in a random sequence, trying the colors also at random, expecting that the color placed generate a valid coloring. If that is not possible, it is painted in any color. Due to the random character of this procedure, the number of ways in which the vertices could be painted is $n!$; in each one of them colors could be tried in $k!$ different ways, which produces a large diversity of solutions.

During the improvement phase (lines 11 to 14 of Figure 4), a vertex and a color are chosen at random. In line 15 the objective function considered is (1) with a value of $\mu = 20$. With values of μ greater than one, as is the case, feasible solutions are found more frequently, validity being preferred to finding non feasible solutions with very small rigidity. This is particularly useful in our case, because the last feasible solution found is considered as the best of each run. The process ends once all the colors have been tested in all the vertices and no better coloring has been found. The cycle 2-18 was executed 100 times ($PoolSize = 100$) and the feasible solution with lesser rigidity was chosen.

2.4 Scatter Search

The Scatter Search algorithm for the coloring problem is as follows:

```
Begin
1 A set of 100 GRASP solutions is generated
2 While new solutions are added to the RefSet:
3 Delete from the Pool all redundant solutions.
4 Choose 5 solutions with valid coloring and minimum rigidity.
5 Choose the other 5 solutions with one or two badly colored vertices and
  minimum rigidity.
6 Empty the Pool.
7 Consider all the possible combinations of sets with two elements.
8 Consider the mixture of the best 4 elements in sets with three elements.
9 For all the subsets with two elements $a$ and $b$
10   For $m = 1$ to $partitions$
11   Mix randomly the elements of the solution $Col(b)$ in $Col(a)$
12   Use the improvement method in the new solution
13   Add the solution to the Pool.
```

```
14   End {For}
15 End {For every}
16 For the four best elements of the $RefSet$:
17   Mix the three elements in a proportion of 1/3 of each one
18   Use in the solutions the improvement method and add them to
     the Pool
19 End {For the four}
20 End {While}
End
```

For the robust coloring problem the stages used were:

The Diversification Generation Method, 1, consisting in running the GRASP algorithm described in the previous section with $\mu = 1$, to obtain thus 100 test solutions; in this case, unlike the GRASP, the solution considered as better is the one that minimizes function (1), whether or not it is feasible. One consequence of using that value of μ, is that we can find either feasible solutions of small rigidity, or non feasible solutions with a pair of incompatible chords, with even less rigidity than the feasible ones; this in order to generate a pool with elements of quality and diversity.

In the Reference Set Update Method, 2 - 6, 10 elements are chosen among the 100; 5 of them are the best feasible solutions different from one another, and the other 5 are the different non feasible solutions with minimum rigidity. In this way we build the reference set with 50% quality solutions, the feasible ones, and the other 50% diversity solutions, the non feasible ones because of a few incompatible chords, with small rigidity.

Two k-colorings, $S = (V_1, V_2, \ldots, V_k)$ and $S' = (V'_1, \ldots, V'_k)$ are equal, except for a permutation of colors; if for every $i = 1, \ldots, k$ there is a $j = 1, \ldots, k$ such that $V_i = V_j$. To make the identification of two equal k-colorings easier, we now design a coloring 'S as a $n-plet$ $(u_1, u_2, \ldots, u_n)$, where u_i is the color assigned to vertex l, for $l = 1, \ldots, n$. Thus, if we have two k-colorings, $S = (u_1, u_2, \ldots, u_n)$ and $S' = (v_1, \ldots, v_n)$, they are equal if when we take a color array $(1, \ldots, k)$ and assign to the u_1 position of the array the value v_1, and then take the u_2 position in the array, and we assign to it the value v_2, in case there is no assigned value; if there is another value already assigned, check if this value is V_2, for, if it is not, that will be a violation to the fact that S is equal to S'. If the assigned value is v_2, we go on in the same way with positions $u_3, u_4, \ldots, u_n$, of the array until we find a violation. If any violation exists, then the k-colorings S and S' cannot be equal. In the opposite case, they are equal.

A measure of how different two solutions are with respect to their chromatic classes is to count, using the former procedure, the number of violations found. Those solutions with the greater number of violations with respect to the elements of each of the better ones provide the most different solutions.

For the Subset Generation Method, 7-8, all the possible subsets with two elements were taken, and from the two elements of best quality and the two with best diversity are chosen the sets with three elements.

For the Solution Combination Method, given two solutions$a$, b, the elements of each one of them are mixed at random, 9- 15. In a similar way were mixed three elements, 16-19. To the new solution produced an improvement method identical to the one used in the GRASP algorithm was applied.

3 Computer results

In this work we have used new instances to prove the four algorithms described: simulated annealing, Tabu, GRASP and scatter search, and to compare its results among them. These results are shown in Table 1, where the number of vertices n, the valid number of colors k and the minimum rigidity $R(C(k, R))$ obtained for each method appear.

Graphs equivalent to those presented in Ramírez Rodríguez (2001) were generated, having from 20 to 120 vertices (first column) second and third columns indicates the number of vertices in the instance and the number of colors used, respectively. The following columns for each algorithm presents the solution found by each method as their time of execution in seconds. The GRASP was executed 100 times and the best solution found is presented. In the Scatter Search algorithm, 100 initial solutions were taken, 50/50 between quality and diversity, 5 elements in the RefSet and 5 steps for reconnecting trajectories. In the Simulated Annealing algorithm it was considered $a = 0.99$ and a number of iterations in each generation equal to $exp(2/d^n)$, were n is the generation number.

In 18 of the 22 instances a solution better than or equal to the best known solution could be found, and only in four cases the best solution found happens to be that of the Scatter Search algorithm. As for the execution times, although the SA algorithm is relatively slow for small instances, it begins to be evident that for instances of more than 90 vertices it becomes the best option.

4 Conclusions

In this paper heuristic algorithms for the RCP were presented. No exact solutions were found for instances with more than 15 vertices. The proposed algorithms find solutions for instances of up to 120 vertices. All the

$G_{n,0.5}$	n	k	Tabu	GRASP	Scatter S.	Sim. Ann.
al(20)	20	7	7.0970 / < 1	7.1423 / 0.14	6.9046 / 1.3	6.6796 / 8.26
al(20)	20	8	4.7710 / < 1	4.6934 / 0.15	4.6934 / 1.3	3.5934 / 5.67
al(30)	30	10	8.0623 / < 1	7.5749 / 0.44	7.5749 / 3.6	7.5749 / 8.17
al(30)	30	11	6.0565 / < 1	5.9318 / 0.49	5.8890 / 3.6	5.8890 / 6.57
al(40)	40	14	7.1709 / 15	7.3950 / 1.0	7.0837 / 8.5	6.9901 / 21.49
al(40)	40	15	5.8173 / 14	6.3117 / 1.0	5.6708 / 7.2	4.9947 / 19.29
al(50)	50	17	9.8259 / 33	8.9531 / 1.9	8.2587 / 13	8.2587 / 15.89
al(50)	50	18	7.4966 / 33	7.1464 / 1.9	6.7164 / 12	6.7164 / 24.47
al(60)	60	20	9.8331 / 69	9.9687 / 3.3	8.8676 / 19	8.7568 / 23.03
al(60)	60	21	8.2181 / 69	8.1430 / 3.4	7.2380 / 20	7.2048 / 23.30
al(70)	70	24	11.1307 / 128	11.2388 / 5.31	9.2634 / 36	9.3555 / 35.02
al(70)	70	25	9.5478 / 128	9.2145 / 5.56	7.7048 / 33	7.3799 / 50.33
al(80)	80	27	11.1946 / 218	11.7512 / 8.0	9.9835 / 52	9.7132 / 75.41
al(80)	80	28	10.5845 / 219	10.2631 / 8.2	8.5961 / 43	8.6118 / 52.42
al(90)	90	30	12.2832 / 350	13.4919 / 11.5	10.8911 / 47	10.6400 / 41.2
al(90)	90	31	11.3699 / 349	11.5060 / 11.9	9.5008 / 89	9.5570 / 60.9
al(100)	100	34	12.1932 / 544	12.8675 / 15.8	10.0470 / 123	9.9658 / 99.59
al(100)	100	35	12.0650 / 885	11.1317 / 15.6	9.4259 / 124	9.0303 / 78.0
al(110)	110	37	—	12.7681 / 20.7	10.8463 / 149	10.6524 / 152.2
al(110)	110	38	—	11.6574 / 20.2	9.9558 / 171	9.8614 / 115.7
al(120)	120	40	—	15.0014 / 26.8	11.3507 / 190	11.1545 / 144
al(120)	120	41	—	13.5266 / 27.5	10.1258 / 191	10.1307 / 150

Table 1: Comparrative results.

heuristics found the optimal solution of problems whose solution is known, Yáñez and Ramírez (2003). The results are encouraging, which drives us to work on larger graphs and think of refining some of the proposed methods; for instance, considering adaptive and learning implementations in simulated annealing; using reactive GRASP, path relinking, ants, among others.

References

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

[2] Feo, T.A.; Resende, M. (1995) "Greedy randomized adaptive search procedures", *Journal of Global Optimization* **6**: 109–134.

[3] Glover, F.; Laguna, M. (1997) *Tabu Search.* Kluwer Academic Publishers, Boston.

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

[5] Johnson, D.S.; Aragon, C.R.; McGeoch, L.A.; Schevon. C. (1991) "Optimization by simulated annealing: an experimental evaluation. Part II: graph coloring and number partitioning", *Operations Research* **39**(3): 378–406.

[6] Ramírez-Rodríguez, J. (2001) *Extensiones del problema de coloración de grafos.* Tesis Doctoral, Universidad Complutense de Madrid, Madrid. Disponible en: `http://eprints.ucm.es/4479/`

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