

---

# Tabu Search with Simple Ejection Chains for Coloring Graphs

José Luis González-Velarde

Centro de Sistemas de Manufactura, Tec de Monterrey, Monterrey, N.L. 64920, México

Manuel Laguna

Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA

Last revision: February 14, 2002

---

## ABSTRACT

We present a Tabu Search (TS) method that employs a simple version of ejection chains for coloring graphs. The procedure is tested on a set of benchmark problems. Empirical results indicate that the proposed TS implementation outperforms other metaheuristic methods, including Simulated Annealing, a previous version of Tabu Search and a recent implementation of a Greedy Randomized Adaptive Search Procedure (GRASP).

---

*Key Words:* Graph coloring, metaheuristics, tabu search, ejection chains.

## Introduction

A *coloring* of a graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E$  is a mapping  $c : V \rightarrow F$ , where  $F$  is a finite set, such that whenever  $(v, w) \in E$ , then  $c(v) \neq c(w)$ . The elements of  $F$  will be referred to as the *colors*; so that colloquially, the definition basically implies that adjacent vertices cannot be assigned the same color.

When a coloring exists, and  $|c(V)| = k$ ,  $c$  is said to be a *k-coloring* of  $G$ . Notice that a  $k$ -coloring partitions  $V$  into  $k$  sets  $V_1, \dots, V_k$ , such that no two vertices in the same set are adjacent, i.e., if  $v, w \in V_i$ ,  $1 \leq i \leq k$  then  $(v, w) \notin E$ . The sets  $V_1, \dots, V_k$  are referred to as *color classes* or *colors*, and  $G$  is said to be *k-colorable*. The *chromatic number*,  $\mathbf{c}(G)$ , is the smallest number of colors needed to color  $G$ , that is, the smallest  $k$  such that there exists a coloring  $c$  for  $G$  and  $|c(V)| = k$ .

There are two versions of the coloring problem:

*Optimization version:* Given an instance  $G = (V, E)$ , determine  $\mathbf{c}(G)$ .

*Decision version:* Given an instance  $G = (V, E)$  and a positive integer  $k$ , is there a coloring of  $G$  using at most  $k$  colors?

Graph coloring is a well-known *NP*-hard problem [1]; this is why a variety of heuristic and metaheuristic approaches have been proposed to produce good colorings in a reasonable amount of time.

### 1. Heuristics and Metaheuristics for Graph Coloring

A number of heuristic algorithms have been developed for the graph coloring problem, but unfortunately, on any given instance, they may produce colorings that are very far from optimal. In fact, it is known that if there were an approximation polynomial algorithm for graph coloring that could guarantee to use at most twice the optimal number of colors, then it would be possible to obtain an optimal coloring in polynomial time, implying that  $P = NP$ . This implies that obtaining near-optimal colors is as hard as getting optimal ones.

The simplest heuristic methods are sequential coloring strategies, an example can be found in [2]. This algorithm can easily be implemented so that its worst-case complexity is  $O(n^2)$ . One of the most successful deterministic heuristics is the Recursive Largest First (RLF) procedure due to Leighton [3], which is based on the idea of coloring as much as possible with one color before introducing another. Johnson, et al. [4] developed a randomized version of this method called the XRLF that improves the performance of the simple heuristic.

Metaheuristics, such as simulated annealing [5], tabu search [6], genetic algorithms [7] and GRASP [8,9], have also been applied to graph coloring. Chams, Hertz and de Werra [10] present results of applying simulated annealing to graph coloring. Both pure simulated annealing and an approach that combines XRLF and simulated annealing are discussed. The hybrid method, which we denote as SA proceeds as follows. XRLF is used to

construct color classes until the number of vertices in the graph is below a specified level. Simulated annealing is then applied to color the remaining vertices. Johnson et al [4] report on extensive experiments using several variants of simulated annealing on random graphs of varying size and density.

Hertz and De Werra [11] were the first to describe a tabu search implementation for graph coloring (TABUCOL), which gives good solutions and outperforms simulated annealing on several random dense graphs tested. We describe the first Tabu Search procedure in greater detail to contrast it with our current development. The method attempts to find a legal coloring with  $k$  colors, where the value of  $k$  can be changed to find improved coloring schemes. A solution  $s = \{V_1, \dots, V_k\}$  is a partition of the set of vertices  $V$  into  $k$  subsets and  $E(V_i) = \{(v, w) \in E \mid v \in V_i, w \in V_i\}$ . The quality of a solution  $s$ , is then given by the following objective function:

$$f(s) = \sum_{i=1}^k |E(V_i)|$$

The procedure stops when  $f(s) = 0$  (or after a pre-specified number of iterations is reached without finding a legal coloring for a given  $k$ ). A neighbor  $s'$  of  $s$  is found by first randomly choosing  $v' = v$  or  $w$ , such that  $(v, w) \in E(V_1) \cup \dots \cup E(V_k)$ , and then, assuming that  $v' \in V_i$ , randomly choosing a color  $j \neq i$ . The new solution  $s' = \{V_1, \dots, V_k\}$  is obtained by:

$$\begin{aligned} V_j' &= V_j \cup \{v'\} \\ V_i' &= V_i - \{v'\} \\ V_r' &= V_r \text{ for } r = 1, \dots, k \text{ and } r \neq i, j \end{aligned}$$

The procedure selects the best neighbor  $s'$ , (as measured by the objective function value) from a collection of randomly generated neighbors. The size of the neighborhood is an empirically determined search parameter. After a move of vertex  $v \in V_i$  to  $V_j$  a tabu activation rule forbids the move that returns  $v$  to  $V_i$  for a number of iterations. The number of iterations that the move remains tabu active is the so-called tabu tenure (which becomes another search parameter). A more recent tabu search procedure is due to Dorne and Hao [12].

Fleurent and Ferland [13], proposed an evolutionary method for graph coloring. An important feature of this procedure is that members of the population are subject to an improvement procedure based on tabu search.

Recently Laguna and Martí [14] implemented a GRASP for this problem. Each GRASP iteration consists of a construction and an improvement phase. The construction uses an adaptive pseudo-random and greedy function to choose vertices for coloring. The improvement phase is an exchange procedure that stops at local optima. The procedure is shown to be particularly effective when dealing with sparse random graphs.

In this paper, we present the results of developing and applying an ejection chain mechanism within a tabu search procedure for the problem of coloring graphs. The next section presents a detailed description for the proposed procedure, section 3 presents the empirical results and concluding remarks are given in section 4.

### 3. Procedure Description

The procedure starts with a polynomial algorithm that produces a feasible coloring, the pseudo-code is given in Figure 1. The input is the adjacency matrix, and the output is a feasible coloring of the graph. The pseudo-code assumes that  $n = |V|$

```

for  $i = 1$  to  $n$  assign to  $i$  color 1

for  $i = 1$  to  $n$ 
    for  $j = 1$  to  $n$ 
        if  $i$  is adjacent to  $j$  and have the same color
            (search a new color for  $j$ )
             $k =$  color of  $j + 1$ ;
            for  $l = 1$  to  $n$ 
                if color of  $l$  is  $k$  and  $l$  is adjacent to  $j$ 
                     $k = k + 1$ 
                next  $l$ 
            assign color  $k$  to  $j$ 
        rof
    fi
rof

```

**Figure 1.** Initial Coloring

Once a feasible coloring is obtained the procedure attempts to reduce the number of colors used, one at a time. This reduction may not result in a feasible coloring, we then apply our task of the tabu search method whose task is to find a feasible assignment of colors. This is summarized in the pseudo-code of figure 2.

1. reduce by the number of colors
2. apply tabu\_search
3. **if** a feasible coloring is obtained **GOTO** 1
4. **else** **END**

**Figure 2.** Reduction procedure

We define a conflict color as the number of adjacent vertices that have been assigned to that color, while the total number of conflicts is defined as the sum of all color conflicts in each color. A feasible coloring is reached when the total number of conflicts is zero. This definition is equivalent to the objective function  $f(s)$  employed in [11].

Our tabu search procedure operates on the following neighborhood structure. Given a solution  $s$ , the neighborhood is constructed considering three kind of moves:

- i) a vertex  $j_1$  changes its color from  $i_0$  to a different color  $i_1$
- ii) a vertex  $j_1$  changes its color from  $i_0$  to a different color  $i_1$ , in turn forcing some vertex  $j_2$  with color  $i_1$  to change to another color  $i_2$  (which may be different or equal to  $i_0$ )
- iii) a vertex  $j_1$  changes its color from  $i_0$  to a different color  $i_1$ , in turn forcing some vertex  $j_2$  with color  $i_1$  to change to another color  $i_2$  (which may be different or equal to  $i_0$ ) and again forcing a vertex  $j_3$  with color  $i_2$  to change to color  $i_3$ .

Move (i) is a very simple move, which Hertz and De Werra used in their tabu search procedure for coloring graphs [11]. Moves (ii) and (iii) are moves that can be constructed by a composition of successive simple moves and can be regarded as a type of ejection chain. The approaches involving ejection chains have been recently applied with significant success in several problem areas such as generalized assignment [15], clustering [16], traveling salesperson problems [17] and vehicle routing [18].

The variety of different moves that can be examined with ejection chains is wide. For example, move (ii) leads to two different structures, one occurs when  $i_0 \neq i_2$ , and another when  $i_0 = i_2$ , corresponding to the exchange of colors between vertex  $j_1$  and vertex  $j_2$ . The range of possibilities, a total of seven, is shown in Figure 3.

Move	Vertex 1	Vertex 2	Vertex 3	Move	Vertex 1	Vertex 2	Vertex 3
(i)	Color 0			(i)	Color 1		
(iia)	Color 0	Color 1		(iia)	Color 1	Color 2	
(iib)	Color 0	Color 1		(iib)	Color 1	Color 0	
(iiia)	Color 0	Color 1	Color 2	(iiia)	Color 1	Color 2	Color 3
(iiib)	Color 0	Color 1	Color 2	(iiib)	Color 1	Color 2	Color 0
(iiic)	Color 0	Color 1	Color 2	(iiic)	Color 1	Color 2	Color 1
(iiid)	Color 0	Color 1	Color 2	(iiid)	Color 1	Color 0	Color 3

Color assignment before the move

Color assignment after the move

**Figure 3. Possible moves**

During the search, the procedure maintains a list of  $n$  preferred colors, one for each vertex in the graph. The preferred color for a given vertex is defined as the one that causes the least number of conflicts if the vertex were assigned to that color. This preferred color is updated every time a move is executed. The advantage of having this array of colors is that the computing effort is considerably reduced since the moves to be examined involve only the preferred color for each vertex.

Two types of tabu activation rules are used within a short term memory structure. When a vertex  $j$  changes its color from  $i_0$  to  $i_1$ , vertex  $j$  cannot be assigned color  $i_0$  for some number of iterations  $t_1$ . This number of iterations is referred to as the tabu tenure for this move. A second tabu tenure is used to lock vertex  $j$  into color  $i_1$  for some number of iterations  $t_2$ . The length for these tabu tenures is not the same:  $t_1$  is given a larger value than  $t_2$ . The rationale behind this logic is that preventing a vertex from leaving a color is more restrictive than preventing it from acquiring some other color. Initially, the value for  $t_1$  was set to  $n$ , the number of available colors, and the value for  $t_2$ , was set to  $n/2$ . Further experimentation showed that a random value between  $n/2$ , and  $3n/2$  for  $t_1$ , and  $n/4$ , and  $3n/4$  for  $t_2$ , produced better results. Each one of the two tabu tenures takes a random value between two corresponding specified limits: the way this limits

A long-term memory is kept in a matrix  $n \times n$ , called *freq\_mat*. Each entry  $(j, j')$  of the matrix is incremented every iteration in which the two vertices  $j$  and  $j'$  are assigned the same color. A frequency measure based on these counts is thus constructed to induce the diversification of solutions generated during the search. Move values are calculated as follows:

For a simple move, i.e. change the color of a vertex  $j$ , from  $i_0$  to  $i_1$  the frequency of the move is defined as  $\text{Min} \{ \text{freq\_mat}(j, j') \mid j' \text{ has color } i_1 \}$ . For composite moves the frequency of the move is computed as the minimum of the frequencies of each simple move. The move value is then modified according to the formula:

$$\text{Move\_Value}' = \text{Move\_Value} (1 + \alpha f'),$$

where  $f' = \text{move frequency} / \text{max. frequency}$ , where  $\text{max. frequency} = \text{Max} \{ \text{freq\_mat}(i, j) \mid i, j \in V \}$  and  $\alpha = 0$  if  $\text{iter} \leq n_0$  or if the move produces an improvement on the current solution.

Initially the parameter  $\alpha$  is given the value zero, so that the frequency has no influence at all, the value is increased after some number of iterations is reached, in order to provide diversity to the search.

The tabu search stops when the number of conflicts is zero, and then proceeds to the reduction phase, or else after a specified number of iterations is performed. A number of iterations equal to  $n / 2$  was used in our tabu search implementation.

### 3. Computational Results

Two families of graphs were chosen for computational testing. The first class includes instances with known optimal solutions from Michael Trick's "Graph Coloring Instances" page (<http://mat.gsia.cmu.edu/color/instances.html>).

LEI: Leighton Graphs [3]

MYC: Graphs based on the Mycielski transformation (Michael Trick)

REG: Graphs based on register allocation for variables in real code (Gary Lewandowski)

SGB: Graphs from Donald Knuth's Stanford Graph Base

The LEI instances are generated by a procedure proposed by Leighton [3], which constructs graphs of known chromatic number. Given the number of vertices  $n$ , the desired chromatic number  $k$ , the average vertex degree  $d$ , and a random vector of nonnegative integers  $(b_k, b_{k-1}, \dots, b_2)$  with  $b_k \geq 1$ , the method introduces edges such that there are  $b_i$  cliques of size  $i$  and the chromatic number and average vertex degree are  $k$  and  $d$ , respectively. Since the true chromatic number is known, the graphs are useful when assessing the performance of a heuristic. The instances in the LEI set consist of twenty-seven 150-vertex graphs and twelve 450-vertex graphs with chromatic number ranging from 5 to 25, and with average vertex degrees ranging from 11 to 77. We restrict our testing to the twelve instances with 450 vertices.

The MYC set consists of 5 instances with known optima. We use all five instances for testing. The REG set consists of 14 instances with known optima. We also use all 14 instances for testing. The SGB instances are divided into four sets: book graphs, game graphs, mile graphs, and queen graphs.

Given a work of literature, a *book graph* is created by first assigning a character to a vertex. Two vertices are connected by an edge if the corresponding characters encounter each other in the book. Knuth has created the corresponding graphs for five classic works: Tolstoy's Anna Karenina (anna), Dickens' David Copperfield (david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck), and Hugo's Les Miserables (jean).

The games played in a college football season can be represented by a *game graph*, where each vertex is associated with one college team. Two teams are connected by an edge if they played each other during the season. Knuth has created a graph for the 1990 college football season.

*Mile graphs* are similar to geometric graphs in that vertices are placed in space and an edge exists between two vertices if the vertices are sufficiently close. These graphs are not random, since the vertices represent a set of United States cities and the distance between them are given by the road mileage from 1947. These graphs are also due to Knuth.

Given an  $n \times n$  chessboard, a *queen graph* has  $n^2$  vertices, where each vertex corresponds to a position (square) on the board. Two vertices are connected by an edge if the corresponding squares are in the same row, column or diagonal. A feasible solution to the

$n$ -coloring problem on this graph means that it is possible to place  $n$  sets of  $n$  queens on the board so that no two queens of the same set attack each other (i.e., they are not in the same row, column, or diagonal).

We start by comparing our TS implementation with other procedures reported in the literature. We use the procedures RLF [3], XRLF [4], a Simulated Annealing procedure [4], TABUCOL [11], an evolutionary method [13], and a GRASP procedure [14]. The results of this experiment are reported in Table 1.

<i>Name</i>	<i>Num</i>	<i>n</i>	<i>m</i>	OPT	RLF	XRLF	SA	TS1	GRASP	TS2
LEI	12	450	11005.5	15.0	17.8	16.9	18.0	19.0	16.5	18.7
MYC	5	73.4	688.4	6.0	6.0	6.0	6.0	6.0	6.0	6.0
REG	14	362.1	7608.1	37.4	37.4	40.1	40.1	57.1	37.4	37.4
SGB	10	113.9	2835.2	22.6	22.6	26.0	26.0	24.1	22.7	22.7
Tot./Av	41	249.9	20.2	20.2	20.9	20.8	22.5	26.6	20.6	21.2

The columns in Table 1 consist of: (1) identifier of instances, (2) number of instances, (3) number of vertices, (4) average number of edges, (5) average number of colors used in the optimal coloring, (6) to (11) average number of colors used per each procedure.

It is interesting to note that the simple heuristics RLF and XRLF perform remarkably well in this set of problems (with average number of colors equal to 20.9 and 20.8, respectively). We should also mention that the performance of SA and TABUCOL is highly dependent on the chosen values of their search parameters. Most notably is the selection of the initial  $k$  (i.e. the number of colors for which the procedure is attempting to find a legal coloring). The results reported in Table 1 correspond to our own implementation of the competitive algorithms, and were found after an extensive experimentation to choose parameter values that yield the best performance for each method. The number of optima found by each procedure and the average computer time are reported in Table 2. The times (in seconds) for all the methods were obtained by running the procedures (coded in C) on a personal computer with a 233 MHz Pentium processor.

<i>Name</i>	RLF	XRLF	SA	TABUCOL	GRASP	TS2
LEI	3 / 0.31	5 / 6.69	2 / 427.12	0 / 40.92	6 / 82.77	3/108.8
MYC	5 / 0.01	5 / 0.23	5 / 0.37	5 / 2.36	5 / 2.36	5 / 0.04
REG	14 / 0.20	14 / 3.65	3 / 62.81	0 / 131.50	14 / 49.95	14 / 2.79
SGB	10 / 0.02	8 / 0.59	6 / 0.70	8 / 7.38	9 / 6.14	9 / 2.37
Total/Avg	32 / 0.14	32 / 2.79	16 / 122.75	13 / 45.54	34 / 35.07	31/28.32

Table 2 reveals the inferior performance of both SA and TABUCOL when considering the current set of problems. These procedures find the least number of optimal solutions and require the most amount of time. TS2 is very competitive in terms of number of optimal solutions; however, it requires several orders of magnitude more time than RLF. It is worth

to notice that GRASP is the method that found more optimal solutions, but the time required is higher than the time required by TS2.

The graphs in the MYC set were optimally colored by the procedure that produces the initial solution (Figure 1), so when TS2 tried to reduce the number of colors by one the procedure stopped. This explains the short time employed in achieving optimality. The same happened with the Books subset in the SGB family. This is evidence that those graphs are easily colored.

The computational studies in all the procedures reviewed above employ graphs for which the probability that an edge exists is 0.5 or more. A  $G_{n,p}$  random graph has  $n$  vertices and a probability  $p$  that an edge exists between any pair of edges (independently from the existence of any other edge).

A second set of experiments was conducted using random graphs. For this experiment we employ 100  $G_{n,0.1}$  instances generated by Laguna and Martí [13]. Table 3 reports the results of this experiment.

$n$	$m$	RLF	XRLF	SA	TABUCOL	GRASP	TS2
50	218.5	5.20	5.16	5.84	5.00	4.92	4.84
100	889.8	7.88	7.56	8.60	7.76	7.08	7
250	5654.5	14.08	13.48	14.36	14.00	13.04	13.04
500	22708.8	23.20	22.16	21.48	23.48	21.88	21.84
Ave.	7367.9	12.59	12.09	12.57	12.56	11.73	11.68

Table 3 shows that GRASP obtains good results for the sparse graphs. TS2 improves these results, except for the 250 vertex instances for which both procedures worked equally well. For the 500 vertices, TS2 performs better than GRASP but not better than SA, which still remains the best for that size. On the overall however, TS2 performed better than all the other heuristics on these graph instances!

$N$	RLF	XRLF	SA	TABUCOL	GRASP	TS2
50	17 / 0.00	18 / 0.13	8 / 0.20	22 / 0.67	24 / 0.47	25 / 17.69
100	5 / 0.01	12 / 0.44	16 / 5.13	7 / 2.35	24 / 2.42	25 / 39.72
250	0 / 0.11	14 / 2.52	20 / 40.48	1 / 15.46	25 / 26.13	25 / 480.4
500	0 / 0.75	7 / 11.37	23 / 989.35	0 / 88.26	13 / 183.71	14 / 1497.2
Tot/Av.	22 / 0.22	51 / 3.61	67 / 258.79	30 / 26.68	86 / 53.18	89 / 508.75

The results in Table 4 show the number of best solutions achieved by each of the heuristics along with the computational time. GRASP was superior to the first four heuristics, but TS2 could find more best solutions than GRASP. However, it should also be noted that TS2 procedure is the most time consuming of all the methods tested.

## **5. Conclusions**

In this paper we describe a Tabu Search implementation that uses ejection chains for coloring graphs. We have compared this method with some of the heuristics and metaheuristics that have previously appeared in the literature. Experimentation proved that the method is competitive in some of the classic benchmark problems. Additionally a set of sparse random graphs was used, and the tests showed that for this kind of problems our method worked better than competing solution methods.

## References

- [1] M. R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, 1979
- [2] Johnson, D.S., “Worst-case behavior of graph coloring algorithms”, *Proceedings of the Fifth Southeastern Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica Publishing Winnipeg, Canada (1974) 513-528.
- [3] Leighton, F.T. A Graph Coloring Algorithm for Large Scheduling Problems, *J. Res. Nat. Bur. Standards*, **84**, (1979) 489-506.
- [4] D. S. Johnson, C. A. Aragon, L. A. Mcgeoch and C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation – Part II (Graph Coloring and Number Partitioning), *Operations Research*, **31**, (1991) 378-406.
- [5] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi Optimization by Simulated Annealing, *Science*, 220, (1983) 671-680.
- [6] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.
- [7] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI. 1975
- [8] T. A. R. E. Feo, and M. G. C. Resende, A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem, *Operations Research Letters*, **8**, (1989) 67-71.
- [9] T. A. R. E. Feo and M. G. C. Resende, Greedy Randomized Adaptive Search Procedures, *Journal of Global Optimization* **2**, (1995) 1-27.
- [10] M.. Chams, A. Hertz and D. de Werra, Some Experiments with Simulated Annealing for Coloring Graphs, *European Journal of Operational Research*, **32**, (1987) 260-266.
- [11] A. Hertz and D. de Werra, Using Tabu Search Techniques for Graph Coloring, *Computing* **39**, (1988) 345-351.
- [12] R. Dorne and J.-K. Hao “Tabu Search for Graph Coloring, T-Coloring and Set T-Colorings,” in S. Voß, S. Martello, I. Osman, C. Roucairol (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston (1999) 33-47.
- [13] Fleurent, C. and J. A. Ferland “Genetic and Hybrid Algorithms for Graph Coloring,” *Annals of Operation Research*, **63**, (1996) pp. 437-464.
- [14] M. Laguna and R. Martí “A GRASP for Coloring Sparse Graphs,” *Computational Optimization and Applications*, **19:2**, (2001) 165-178.

- [15] M. Laguna, J. P. Kelly, J. L. González-Velarde and F. Glover, Tabu Search for the Multilevel Generalized Assignment Problem, *European Journal of Operational Research*, **42**, (1995) 677-687.
- [16] U. Dorndorf and E. Pesch, Fast Clustering Algorithms, *ORSA Journal on Computing*, **6**, (1994) 141-153.
- [17] E. Pesch and F. Glover, TSP Ejection Chains, *Discrete Applied Mathematics* **76** (1997), 165-181.
- [18] C. Rego and C. Roucairol, A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, in: *Meta-Heuristics: Theory and Applications*, eds. I. H. Osman and J. P. Kelly, Kluwer Academic Publishers, 1996 pp. 661-675.