A complete algorithm to solve the graph-coloring problem

Huberto Ayanegui and Alberto Chavez-Aragon

Facultad de Ciencias Basicas, Ingenieria y Tecnologia,
Universidad Autonoma de Tlaxcala,
Calzada de Apizaquito s/n, Apizaco, Tlaxcala, Mexico
{hayanegui, albertochz}@gmail.com

Abstract. The Graph *k*-Colorability Problem (GCP) is a well known NP-hard problem which consist in finding the *k* minimum number of colors to paint the vertices of a graph in such a way that any two vertices joined by an edge have always different colors. Many years ago, Simulated Annealing (SA) was used for graph coloring task obtaining good results; however SA is not a complete algorithm and it not always gets the optimal solution. In this paper GCP is transformed into the Satisfiability Problem and then it is solved using a algorithm that uses the Threshold Accepting algorithm (a variant of SA) and the Davis & Putnam algorithm. The new algorithm is a complete one and so it gets better quality that the classical simulated annealing algorithm.

Keywords: graph coloring, simulated annealing, threshold accepting, davis & putnam.

1 Introduction

Let $G=(V,E)$ be a graph where V is a set of vertices and E is a set of edges. A k-coloring of G is a partition of V into k sets $\{V_1, ..., V_k\}$, such that no two vertices in the same set are adjacent, i.e., if v, w belong to V_i, $1 \leq i \leq k$, then (v, w) not belong to E. The sets $\{V_1, ..., V_k\}$ are referred to as colors. The chromatic number, $x(G)$, is defined as the minimum k for which G is k-colorable. The Graph k-Colorability Problem (GCP) can be stated as follows. Given a graph G, find $x(G)$ and the corresponding coloring. GCP is a NP-hard problem [1].

GCP is very important because it has many applications; some of them are planning and scheduling problems [2][3], timetabling [4], map coloring [5] and many others. Since GCP is a NP-hard problem, until now there are not known deterministic methods that can solved it in a polynomial time [1]. So non-deterministic algorithms have been built to solve it; one of them is Simulated Annealing (SA) [6] that has been used on GCP with good results [7][8]. However, SA is not a complete algorithm and it not always gets the optimal solution.

The approach used in this paper is to transform GCP into a Satisfiability Problem (or SAT problem) [12] and then use the algorithm proposed in this paper. We propose to use iteratively the Threshold Accepting (TA) algorithm (a variant of Simulated Annealing) [9] and then a Davis and Putnam algorithm [10].

2 Simulated annealing and threshold accepting

Simulated annealing (SA) [6] is a stochastic computational technique derived from statistical mechanics to find near global-minimum-cost solutions to large optimization problems. In many instances, finding the global minimum value of an objective function with many degrees of freedom subject to conflicting constraints is an NP-complete problem, since the objective function will tend to have many local minimums. A procedure for solving optimization problems of this type should sample the search space in such a way that it has a high probability of finding the optimal or a near-optimal solution in a reasonable time. Over the past decade, SA has proven to be a powerful technique that meets these criteria for a wide range of problems. SA exploits an analogy between the way a metal cool and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. SA makes a random search which not only accepts changes that increase its cost function f, but also some that decrease it. For this reason, SA uses a control parameter c, which by analogy with the original application is known as the "System Temperature", c starts out high and gradually decreases.

A deteriorating random move from solution S_i to S_j is accepted with a probability $exp^{-(f(S_j)-f(S_i))/c}$. If this move is not deteriorating (the new solution S_j is better than the previous one S_i) then it is accepted and a new random move is proposed again. When the temperature is high, a bad move can be accepted. As c tends to zero, SA becomes more demanding through accept just better moves. The algorithm for minimization is shown below:

```
Procedure SIMULATED ANNEALING
Begin
   INITIALIZE(Si=initial_solution,
                     c=initial_temperature)
   k = 0
   Repeat
      Repeat
         Sj = PERTURBATION(Si)
         If COST(Sj) <= COST(Si) Then
            Si = Sj
         Else If exp(-INC_COST/c) > random[0,1) Then
            Si = Sj
      Until stochastic equilibrium
      k = k + 1
```

```
      c = COOLING(c)
   Until thermal equilibrium
End
```

The INITIALIZE function starts the initial solution S_i and the initial temperature c. The PERTURBATION function makes a random perturbation from S_i to generate a neighborhood solution S_j. The COST function gets the cost from a solution. The INC_COST function gets the difference in cost between S_j and S_i. Finally, the COOLING function decreases the actual temperature parameter c.

A variant of Simulated Annealing (SA) is the Threshold Accepting method (TA). It was designed by Dueck & Scheuer [9] in order to get a more efficient algorithm than Simulated Annealing. The principal difference, between SA and TA, is the mechanism of accepting the solution randomly chosen from the set of neighbors of the current solution. While SA uses a probabilistic model (see equation (1)), TA uses a static model: if the difference between the cost values of the chosen solution S_j and the current one S_i is smaller than a threshold T *(or temperature)*, TA accepts moving to the chosen solution. Otherwise it stays at the current solution. Again, the threshold parameter T is a positive control parameter which decreases with increasing number of iterations and converges to value near to 0. Henceforth, in every iteration some solution deterioration are allowed; this deterioration depends on the current threshold T (see equation (2)); in this way only improving solutions with almost none deterioration solution are accepted at the end of the process.

$$p(S_1, S_2) = \exp(\min\{f(S_1)-f(S_2), 0\}/c) \quad (1)$$

$$\text{COST}(S_j) < \text{COST}(S_i) + T \rightarrow \text{accept } S_j \quad (2)$$

For SAT problems, using a good tune method Threshold Accepting yields better results than Simulated Annealing. This could be because TA does not compute the probabilistic function (1) and does not expend a lot of time making random decisions. The Threshold Accepting algorithm for minimization is the following:

```
Procedure THRESHOLD ACCEPTING
Begin
   INITIALIZE (Si = initial_solution,
               T = initial_threshold or temperature)
      k = 0
   Repeat
         Repeat
           Sj = PERTURBATION(Si)
              E = COST(Sj) – COST(Si)
              If E < T Then
                     Si = Sj
       Until stochastic equilibrium
```

```
        k = k + 1
        T = DECREASE_THRESHOLD(T)
    Until Thermal Equilibrium
End
```

Here, the DECREASE_THRESHOLD function is equivalent to the COOLING function in SA and the threshold T is named "temperature" in order to make more evident that TA belongs to Simulated Annealing Like Algorithms class (SALA). SALA uses Simulated-annealing approach with two main loops: internal loop named Metropolis Cycle and external loop called Temperature Cycle. Number of iterations in internal and external loop usually are tuned experimentally [6], [9]. However, recently an analytical method using a Markov model was proposed to tune TA solving SAT problems.

External loop executed from a initial temperature T_i until a final temperature T_f and the internal loop builds a Markov chain of length L_k which depends on the temperature value T_k (k represents the sequence index in Temperature cycle). A strong relation exists between T_k and L_k in a way that:

$$\text{If } T_k \to \infty, L_k \to 0 \text{ and if } T_k \to 0, L_k \to \infty . \tag{3}$$

Due to TA is applied through a neighborhood structure, V, (PERTURBATION function makes a random perturbation from S_i to generate a neighborhood solution S_j), the maximum number of different solutions that can be rejected from S_i is the size of its neighborhoods, $|V_{Si}|$. Then the maximum length of a Markov chain in a TA algorithm is the number of samples that must be taken in order to evaluate an expected fraction of different solutions from V_{Si} at the final temperature T_f, this is:

$$L_f = C|V_{Si}| . \tag{4}$$

where C varies from $1 \le C \le 4.6$ (exploration from 63% until 99%), L_f is the length of the Markov chain at T_f.

From (3), L_k must be incremented in a similar but inverse way that T_k is decremented. Then for the geometric reduction cooling function used by Kirkpartick [6] , and Dueck and Scheuer [9],

$$T_{k+1} = \alpha T_k. \tag{5}$$

the incremental Markov chain function must be:

$$L_{k+1} = \beta L_k. \tag{6}$$

where

$$\beta = \exp((\ln L_f - \ln L_i) / n). \tag{7}$$

Here, L_i is the length of the Markov chain at T_i, usually $L_i = 1$, and n is the number of temperature steps from T_i to T_f through (5).

Now, the maximum and minimum cost increment produced through the neighborhood structure are:

$$\Delta Z_{Vmax} = \text{Max}\{\text{COST}(S_j) - \text{COST}(S_i)\}. \quad (8)$$

$$\Delta Z_{Vmin} = \text{Min}\{\text{COST}(S_j) - \text{COST}(S_i)\}. \quad (9)$$

for all $S_j \in V_{Si}$, and for all $S_i \in S$

Then T_i and T_f must be calculated as:

$$T_i = \Delta Z_{Vmax}. \quad (10)$$

$$T_f \leq \Delta Z_{Vmin}. \quad (11)$$

This way of determining the initial temperature enable TA to accept any possible transition at the beginning of the process, since T_i is set as the maximum deterioration in cost that may be produced through the neighborhood structure. Similarly, T_f enables TA to have control of the climbing probability until the algorithm performs a greedy local search.

3 Davis & Putnam Method

Satisfiablity Problem [12] (or SAT) is very important in complexity theory.

Let be a propositional formula like formula (12):

$$F = F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \quad (12)$$

where every F_i is a disjunction.

Every F_i is a disjunction of propositional formulas such as X_1 v X_2 v..v X_r. Every F_i is a clause and every X_j is a literal. Every literal can take a truth value (0 or false, 1 or truth). In Satisfiability problem a set of values for the literals should be found, in such a way that the evaluation of (12) be true; otherwise if (12) is not true, we say that F is unsatisfiable. Besides we say that (12) is in Conjunctive Normal Form or CNF.

The Davis & Putnam method is widely regarded as one of the best deterministic methods for deciding the satisfiability [12] of a set of propositional clauses [10]. It is also a complete resolution method. This procedure calls itself after rewriting the input formula according to a number of rules for generating a smaller formula with the same truth value. The rules used for the Davis & Putnam method are:

Rule 1: if the input formula has no clauses, then it is satisfiable
Rule 2: if it has a clause with no literals, it is unsatisfiable

Rule 3:if it has a clause with exactly one literal, then make the literal true and rewrite the formula accordingly

Rule 4:if some variable appear only positively or negatively, then pick one such variable and assign a value to it to make the literal true, and rewrite the formula accordingly

If none rule could be applied, one picks up an arbitrary variable as a branching point and two new formulas are derived by assigning 0 and 1 to this variable. If one of the calls returns with the positive answer the input is satisfiable; otherwise, it is unsatisfiable.

The Davis & Putnam algorithm is shown below:

```
Function DAVIS-PUTNAM(In formula : clauses list)
Begin
   REDUCE(formula, vreduce)
   If formula is empty Then
      Return vreduce
   Else If formula has a clause with no literals Then
      Return fail
     Else
      Choose a literal V from formula
         valuation=DAVIS-PUTNAM(SUBSTITUTION(true,V,
                                              formula))
         If valuation != fail Then
          Return ADD(V=true, vreduce, valuation)
         valuation=DAVIS-PUTNAM(SUBSTITUTION(false,V,
                                             formula))
         If valuation != fail Then
          Return ADD(V=false, vreduce, valuation)
       Return fail
      Endif
End DAVIS-PUTNAM


Function SUBSTITUTION (TF, V, formula)
Begin
   For Each one clause C In formula Do
        If [C contain V and TF=true]or
          [C contain ~V and TF=false] Then
              delete C from formula
        Else If [C contain V and TF=false]or
                  [C contain ~V and TF=true] Then
         delete V from C
      Endif
   Endfor
   Return formula
End_SUBSTITUTION
```

```
Function REDUCE(In Out: formula, vreduce)
Begin
   vreduce = empty
   While exists clause C In formula with exactly one
              literal L
      If L is positive variable V Then
         formula = SUBSTITUTION(true, V, formula)
        vreduce = CONS(V=true, vreduce)
      Else If L is negative variable V Then
         formula = SUBSTITUTION (false, V , formula)
        vreduce = CONS(V=false, vreduce)
      Endif
   Endwhile
   Return(formula)
End_REDUCE
```

The DAVIS-PUTNAM function is the main function and it selects randomly a literal to set a true a group of values in order to create unitary clauses. If that true set values is not the correct solution the complement set of true values is tried. If the new assignment is neither a satisfiable solution, then the formula is unsatisfiable.

The function SUBTITUTION makes the propagation of one literal over all the clauses in *formula*, deleting clauses where occurs the positive literal L and its value is 1 (true). Therefore the clauses where $\sim L$ occurs can delete that literal.

The REDUCE function carries out the search of unitary clauses, so that it can be possible propagate through the function SUBSTITUTION.

4 Graph Coloring through Accepting and Davis & Putnam

Informally coloring a graph with k colors or Graph k-Colorability Problem (GCP) is stated as follows: Is it possible to assign one of k colors to each node of a graph $G=(V,E)$, such that no two adjacent nodes be assigned the same color? If answer is positive we say that the graph is k-colorable and k is the chromatic number x(G). It is possible to transform Graph k-Colorability Problem (GCP) into Satisfiability problem (SAT); that means that for a given graph $G=(V,E)$ and a number k, it is possible to derive a CNF formula F such that F is satisfiable only in the case that G is k-colorable. The formulation of GCP as SAT is made assigning X Boolean variables as follow:

1) Take every node and assign a Boolean variable X_{ij} for every node i and color j; the disjunction of all these variables. In this way every node will have at least one color. Therefore, in the case of figure 1, we have the clauses:

Node 1: X_{11} v X_{12} v X_{13} v X_{14}
Node 2: X_{21} v X_{22} v X_{23} v X_{24}
Node 3: X_{31} v X_{32} v X_{33} v X_{34}
Node 4: X_{41} v X_{42} v X_{43} v X_{44}

2) To avoid the fact that a node has more that one color, add the formula $X_{ij} \rightarrow \sim X_{ik}$

3) In order to be sure that two nodes (V_i, V_j) connected with an arc have different colors, add a clause such that if V_i has color k, V_j should not be color with this color. This clause is writing as $X_{ik} \rightarrow \sim X_{jk}$.

4) In order to know which nodes are connected with an edge, an adjacent matrix A of the graph is needed; its elements are:

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ is connected with } j \\ 0 & \text{otherwise} \end{cases}$$

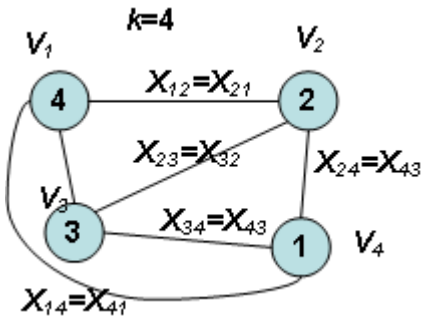

Fig. 1. Graph coloring example

The reduction of a graph to the Conjunctive Normal Form (CNF) generates so many clauses even for small graphs. For example, for a full graph with 7 nodes (42 edges), 308 clauses with 98 literals can be generated. If we use Davis & Putnam algorithm to color a graph, we could start coloring with R colors (the graph's degree or from a number given). If it is not possible to color it, then we can increase R and try again.

Due to find a large chromatic number $x(G)$ is a very hard task for a complete method as Davis & Putnam (it demands many resources), we need an incomplete method to help in this task. For this reason we have chosen the Threshold Accepting

method. TA will search the chromatic number, but as it is known TA not always get the optimal solution. By this reason, the number found by TA is send to a Davis & Putnam procedure, and this one will get the optimal solution. The complete process is shown in the figure 2.

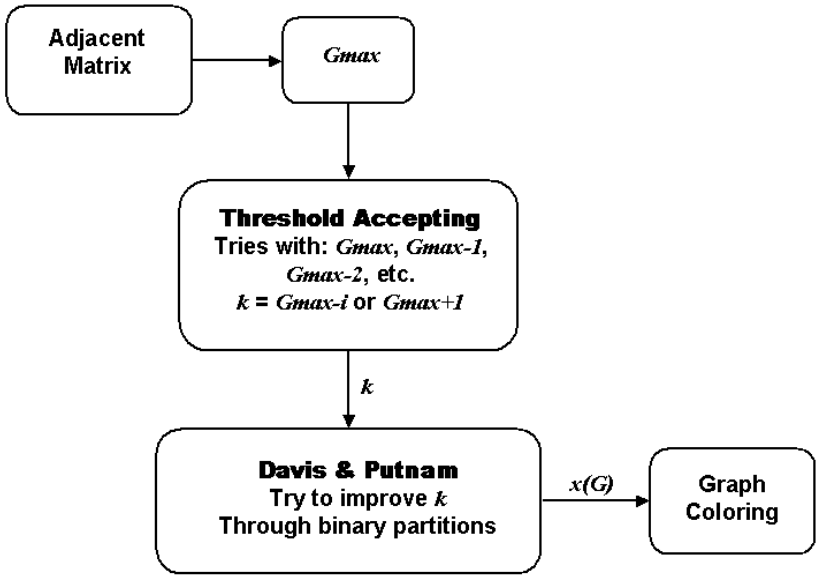

Fig.2. Description of the coloring process

Any graph can be colored with *Gmax+1* colors, where *Gmax* represents the graph degree. For this reason, TA will try coloring with *Gmax* colors. If TA gets a success, then TA will try to color with *Gmax-1*, and so on. When TA finishes, it sends to the output the minimum *k* of colors founded. In other case, when TA can not color with *Gmax* colors, then it will send *k=Gmax+1* to Davis & Putnam procedure.

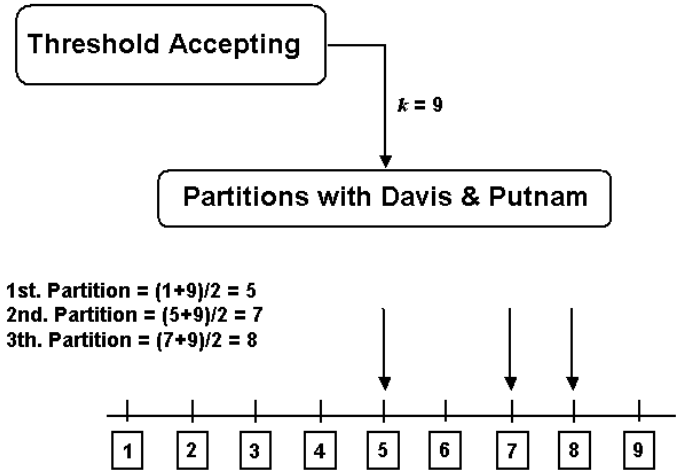

Fig.3. Binary partitions

Davis & Putnam will attempt to decrease the value of k through binary partitions. The first attempt, Davis & Putnam will choose the number of colors given by $(1+k)/2$. If the coloring is right, it will color with $(1+(1+k)/2)/2$ colors, i.e., the left half. Otherwise, the algorithm will color with $((1+k)/2+k)/2$ colors, the right half. This process continues until Davis & Putnam cannot decrease k. So, the chromatic number was found. This situation is shown in figure 2.

The figure 3 shows an example where TA found the number nine as its better solution and it is send to Davis & Putnam procedure. When Davis & Putnam takes the last TA solution, using binary partitions and other rules the optimal solution is waited. For example in the case of the figure 3, if Davis & Putnam can not color with five colors, it moves to other alternative, trying with seven colors. Finally, in the last partition, i.e. (7+9)/2, can not color the graph and so the result is a chromatic number equal to nine.

5 Conclusion

In this paper we presented an algorithm based on Threshold Accepting and Davis & Putnam, to solve the Graph k-Colorability Problem. Because this problem is an NP-hard problem there is not a known deterministic efficient (polinomial) method. Non-deterministic methods are in general more efficient but an optimal solution is not guarantee. This method is a new alternative that promises to be more efficient that the

previous ones. The main contributions of this paper are enumerated below. 1) We proposed a way to transform the graph *k*-colorability problem into a satisfiability problem. 2) In order to solve the former problem we proposed a new approach which makes use of the threshold accepting and Davis & Putnam algorithms. 3) The resulting algorithm is complete and using it we can get better results that the well-known simulated annealing algorithm.

References

1. Garey, M. R. and Johnson, D. S., Computers and Interactability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.
2. Stecke, K., Design Planning, Scheduling and Control Problems of Flexible Manufacturing, *Annals of Operations Research*, Vol. 3, 1985, pp. 3-12.
3. Leighton, F. T., A Graph Coloring Algorithm for Large Scheduling Problems, *J. Res. Nat. Bur. Standard*, Vol. 84, No. 6, 1979, pp. 489-506.
4. Wood, D. C., A Technique for Coloring a Graph Applicable to Large Scale Timetable Problems, *Computer Journal*, Vol. 12, 1969, pp. 317-322.
5. Brelez, D., New Methods to Color Vertices of a Graph, *Comm. ACM*, Vol. 22, 1979, pp. 251-256.
6. Kirkpatrick, S, Gelatt, C.D., Vecchi, M.P., Optimization by Simulated Annealing, *Science*, No. 220, 1983, pp. 671-680.
7. Chams, M., A. Hertz and D. de Werra, Some Experiments with Simulated Annealing for Coloring Graphs, *European Journal of Operational Research*, Vol. 32, 1987, pp. 260-266.
8. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., Optimization by Simulated Annealing: An Experimental Evaluation; Part II: Graph Coloring and Number Partitioning, *Oper. Res.,* No. 39, 1991, pp. 378-406.
9. Dueck Gunter, Scheuer Tobias, Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, No. 90, 1990, pp.161-175.
10. M. Davis and H. Putnam, A Computing Procedure for Quantification Theory. Journal of the Association for Computing Machinery, Vol. 7, No. 1, 1960, pp. 201-215.
12. Science and Technology Center in Discrete Mathematics and Theoretical Computer Science, "Satisfiability Problem: Theory and Applications", Dimacs Series in Discrete Mathematics and Theoretical Computer Science, Editors: Jun Gu, Panos Pardalos, Ding-Zhu.