4th WSEAS Int. Conf. on COMPUTATIONAL INTELLIGENCE, MAN-MACHINE SYSTEMS and CYBERNETICS Miami, Florida, USA, November 17-19, 2005 (pp65-69)

Graph *k*-Colorability through Threshold Accepting and Davis-Putnam

JUAN FRAUSTO-SOLÍS[1] HÉCTOR SANVICENTE-SANCHEZ [2]
MARCO ANTONIO CRUZ-CHAVEZ[3] MÓNICA LARRE BOLAÑOS-CACHO[1] JOSÉ CRISPIN ZAVALA-DÍAZ [3] HUBERTO AYANEGUI[1]
[1]Department of Computer Science, ITESM, Campus Cuernavaca
Paseo de la Reforma 182-A, 62589, Temixco, Morelos, MÉXICO
[2]Instituto Mexicano de Tecnología del Agua
Paseo Cuauhnáhuac 8532, Col. Progreso, C.P.62550, Jiutepec, Morelos, MÉXICO
[3]Engineering and Applied Science Research Center UAEM
Av. Universidad1001, Col. Chamilpa, CP 62270, Cuernavaca Morelos
MÉXICO

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

Key-Words: - Graph Coloring, Simulated Annealing, Davis & Putnam, Threshold Accepting, Chromatic Number, Combinatorial optimization.

1 Introduction

A graph *G* can be defined as $G=(V,E)$ where *V* is a set of vertexes and *E* is a set of edges. A *k*-coloring of *G* is a partition of *V* into *k* sets $\{V_1, \ldots, V_k\}$, such that no two vertexes in the same set are adjacent, i.e., if *v*, *w* belong to V_i, $1 \leq i \leq k$, then (v, w) no belong to *E*. The sets $\{V_1, \ldots, V_k\}$ 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. Because GCP is a NP-hard problem, until now there are not known deterministic methods that can solved it in a polynomial time [1]. So no-deterministic algorithms have been built to solve it; one of them is Simulated Algorithm (SA) [6] that has been used on GCP with good results [7][8]. However, SA is not a complete algorithm and so not always gets the optimal solution. The alternative used in this paper is to transform GCP into a SAT problem and then use the hybrid algorithm proposed here. 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 minima. 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 shown to be a powerful technique which 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 the state 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 old 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 is shown below:

```
Procedure SIMULATED ANNEALING
Begin
  INITIALIZE($S_i$=initial_state, $c$=initial_temperature)
  $k = 0$
  Repeat
    Repeat
      $S_j$ = PERTURBATION($S_i$)
      If COST($S_j$) <= COST($S_i$) Then
        $S_i = S_j$
      Else If exp(-INC_COST/$c$) > random[0,1) Then
        $S_i = S_j$
    Until stochastic equilibrium
    $k = k + 1$
    $c$ = COOLING($c$)
  Until thermal equilibrium
End
```

The INITIALIZE function set 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*, 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. Thus, every iteration allows some solution deterioration depending on the current threshold *T*; finally only improving solutions with almost none deterioration solution are accepted.

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

For SAT problems, using a good tune method [11] Threshold Accepting yields better results than Simulated Annealing, at least with the instances tested in [12]. 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 is shown:

```
Choose an initial solution $S_i$
Choose an initial threshold $T > 0$
Generate a new solution $S_j$ from the actual solution $S_i$
    through a stochastic small perturbation.
Compute $E$ = COST($S_j$) – COST($S_i$)
If $E < T$ Then
  $S_i = S_j$
If a long time no increase in cost quality or too many
    iterations Then
  lower threshold $T$
If sometime no change in quality anymore Then
  stop
```

3 Davis & Putnam method

The Davis & Putnam method is widely regarded as one of the best deterministic methods for deciding the satisfiability 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 of the later rules could be apply, one picks up an arbitrary variable as a branching point and two new formulae are derived by assigning 0 and 1 to this variable. If one of the calls returns with the positive answer, then the input is satisfiable; otherwise, it is unsatisfiable.

Davis & Putnam Algorithm

```
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 set of values in order to create unitary clauses. If that true set values is not the correct solution then 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 ~*L* occurs can delete that literal.

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

4 Graph Coloring through Threshold Accepting and Davis-Putnam

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.

Because to find a large chromatic number *x(G)* it 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 1.

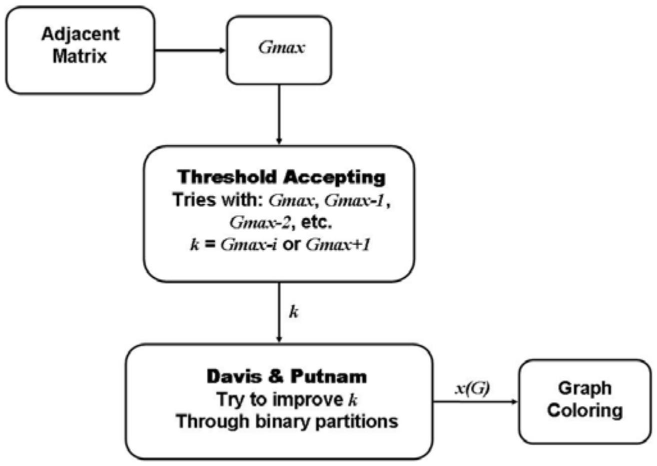

Fig. 1: Description of the coloring process.

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.

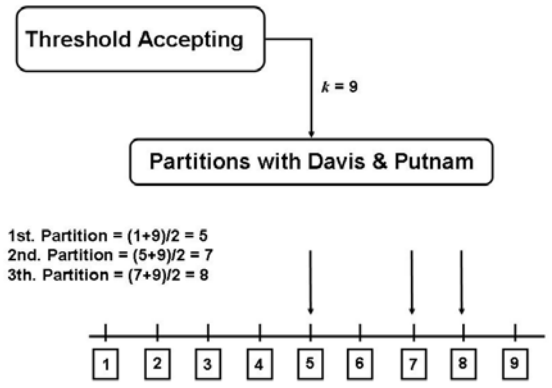
Threshold Accepting
k = 9
Partitions with Davis & Putnam
1st. Partition = (1+9)/2 = 5
2nd. Partition = (5+9)/2 = 7
3th. Partition = (7+9)/2 = 8
1
2
3
4
5
6
7
8
9

Fig. 2: 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, will try color with (1+(1+*k*)/2)/2 colors, i.e., the left half. Otherwise, will try color with ((1+*k*)/2+*k*)/2 colors, the right half. This process continues until Davis & Putnam can not decrease *k*. So, the chromatic number was found. This situation is shown in figure 2.

The figure 2 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 2, 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 present a hybrid algorithm 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. No deterministic methods are in general more efficient but an optimal solution is not guarantee. This hybrid method is a new alternative that promises to be more efficient that the previous ones.

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.

[11] Hector Sanvicente-Sanchez, Juan Frausto-Solís and Froilan Imperial-Valenzuela, Solving SAT Problems with TA Algorithms Using Constant and Dynamic Markov Chains Length, AAIM 2005, *LNCS* Vol. 3521, 2005, pp. 281-290.

[12] Froilan Imperial Valenzuela, M.Sc. Thesis, Tecnológico de Monterrey, Campus Cuernavaca, Mayo 2005.