ORIGINAL PAPER

J M Jaam and A M Hasnah

Improvement of the DSATUR Algorithm for Graph Coloring

Abstract: In this paper we discuss the deterministic Brelaz's DSATUR algorithm for graph coloring. We propose a simple modification that improves the performance of the algorithm. This modification consists of assigning the color that saturates the least number of uncolored vertices, to the selected vertex. Thus, we obtain valid k-colorings better than those obtained with DSATUR without modification. We show also that the DSATUR algorithm is optimal, for a given example, and for the bipartite graphs.

Keywords: Graph coloring, DSATUR, Algorithm, lipartite graphs.

تحسين أداء خوارزمية ديساتيور في تلوين الأشكال

جهاد محمد الجعم، و أحمد مجاهد حسنة

المستخلص: في هذا البحث نناقش "خوارزمية بريلاز" المشهورة والمستخدمة في تلوين الأشكال، والمعروفة باسم "خوارزمية ديساتيور". ثم نقترح تغيرا بسيطاً في طريقة عمل تلك الخوارزمية يؤدي إلى تحسين أدائها. يتمثل هذا التغير في إسناد اللون الذي يشبع أقل عدداً من النقاط غير الملونة إلى النقطة التي تم إختيارها (النقطة قيد التلوين). وبالتالي، نحصل على حلول (ك-تلوين) أفضل من تلك التي يتم الحصول عليها باستخدام "خوارزمية ديساتيور" من غير ذلك التغير. ونبين أيضاً، باستخدام مثال معين، أن "خوارزمية ديساتيور" يوفر الحل الأمثل في تلوين بعض الأشكال، ولا سيما الأشكال ذات الجزأين.

كلمات مدخلية: تلوين الأشكال، خوارزمية ديساتيور، الأشكال ذات الجزأين.

Introduction

The graph coloring problem consists of coloring the vertices of a graph using the fewest number of colors in such a way that no two adjacent vertices are assigned the same color. This problem has a variety of applications involving scheduling and time-tabling, frequency assignments, computer register allocation, printed circuit board testing and pattern matching. Finding the fewest colors of a given graph is a typical combinatorial NP-complete problem (Garey and Johnson, (1979) and Marshall Hall, (1986)). Numerous deterministic and stochastic techniques that investigate the colorability of graphs have been published (See Johnson, (1974); Culberson, (1993) & (1995); Prestwich, (1998) & (2001); Karger, *et al.* (1998); Hertz and de Werra, 1987; Chams, *et al.* 1987; Joslin and Clements, 1999; Christofides, 1971; Wilf, 1984; Skiena, 1990; Jaam, 1995 & 1995 and Jaam, *et al.* 1995). The best known deterministic algorithms to color a graph require in the worst case an amount of time exponentially growing with the size of the graph instance. Brelaz's DSATUR algorithm (Brelaz, 1979), however, is still a good and efficient algorithm to color successfully the vertices of a graph. It is also considered as a standard to compare the efficiency of new graph coloring algorithms (Culberson, *et al.*, (1995); Culberson, (1993) and Prestwich, (1998) & (2001)).

In this paper, we discuss the Brelaz's DSATUR algorithm for graph coloring. We enhance it by a simple heuristic called the least saturated vertex-first, which improves the performance of the algorithm. This modification consists of investigating all the coloring possibilities before assigning a color to the candidate vertex. It then selects the color that saturates the least number of uncolored vertices, and leads to valid colorings better than those obtained by DSATUR algorithm without modification.The paper is structured as follows. In Section 1, we give some necessary definitions and terminology of graphs. In Section 2, we present the DSATUR algorithm and we apply it on a simple graph of 9 vertices, while in Section 3, we show how the performance of DSATUR can be

Jihad M. Jaam and Ahmad M. Hasnah
Computer Science Department
College of Science
University of Qatar
P.O. Box 2713, Doha, Qatar
Tel:(00974)-4-852145, 852962
Fax: (00974)-4-852961, 835061
E-mail: jaam@qu.edu.qa / hasnah@qu.edu.qa

increased by adding the least saturated vertex-first heuristic. In Section 4, we show that the DSATUR algorithm can be used to check if a given graph is bipartite or not, and finally, in Section 5 we conclude the paper.

1- Preliminaries

A *graph* $G = (V,E)$ consists of a set of points $V(G)$ called *vertices* that are connected together by lines called *edges*. An *edge* connects exactly two vertices. The set of edges is denoted by $E(G)$. Two vertices are said to be *adjacent* if and only if they are connected by an edge. The *degree* of a vertex v, denoted by deg(v), is the number of edges connected to that vertex. A *clique* of a graph is a set $V' \subset V(G)$ of mutually adjacent vertices. A *maximum clique* is a clique whose number of vertices is at least as large as that for any other clique in the graph. The *graph coloring* problem is the assignment of colors to vertices so that no two adjacent vertices have the same color. A *valid k-coloring* of a graph G is an assignment $C : V \rightarrow \{1, ..., k\}$ such that if $C(v_i) = C(v_j)$ then $\{v_i, v_j\} \notin E(G)$. The set $\{1, ..., k\}$ represents the set of colors. A graph can be colored using different numbers of colors according to the problem being solved and the problem constraints. The least number of colors used in a *graph coloring* problem is called the *chromatic number* and denoted by $X(G)$. Thus a valid k-coloring is optimal if $k = X(G)$. Finding the chromatic number of a graph $X(G)$ is known to be an NP-complete problem (Garey and Johnson, 1979 and Marshall Hall, 1986), and there is not likely to be an efficient polynomial algorithm that can guarantee finding the chromatic number $X(G)$ for all graphs. However, practically, we relax the problem and attempt to obtain valid k-colorings, where k is an approximation of X. Such colorings are called *approximate colorings*. Note that the size of the maximum clique of a graph is a lower bound on the number of colors needed to color the graph.

2- The DSATUR Algorithm

Brelaz's DSATUR algorithm (Brelaz, 1979) is a good and efficient algorithm to generate valid approximate k-colorings for graphs. It is considered as a reference to compare the efficiency of new graph coloring algorithms (Culberson, *et al.* 1995; Culberson, 1993 and Prestwich, 1998 & 2001). This sequential algorithm is based mainly on two different heuristics: the *maximal saturated vertex-first* and the *maximal degree vertex-next.* The first heuristic consists of coloring the most saturated vertex first, and if there are many vertices of the same maximal degree of saturations the algorithm makes use of the second heuristic which consists of coloring the first vertex of maximal degree as suggested in Matula and Beck (1983). A vertex v is said to be saturated by a given color c, if it is adjacent to a vertex of color c. The *degree of saturation* of the vertex v, denoted by $DSAT(v)$, is the number of different colors assigned to its adjacent vertices. The DSATUR algorithm can be sketched as follows.

```
Step 1: Let V be the set of vertices $V = \{v_1, ...v_n\}$, and
associate with each vertex $v_i$ of $V$ a set of possible
colors, $ColorInterval(v_i) = \{1, 2, ..., n\}$, a set of
saturated colors $SatureSet(v_1) = \emptyset$ with a degree of
saturation for each vertex, $DSAT(v_i) = |SatureSet(v_i)|$

Step 2: Let SDSAT be the set of all non-colored
vertices of the same maximal DSAT.
If $|SDSAT| > 1$ Then select the first vertex $v_i$ of
SDSAT of maximal degree.
Else select the sole vertex $v_i$ of SDSAT.

Step 3: Assign to $v_i$ the first color c from its interval
ColorInterval($v_i$), and let $V = V \setminus \{v_i\}$.

    For all $u$ such that $\{u, v_i\} \in E(G)$ Do
    Begin
    $deg(u) = deg(u) - 1$
    If ( $c \notin SatureSet(u)$ ) Then
    Begin
    $SatureSet(u) = SatureSet(u) \cup \{c\}$
    $DSAT(u) = |SatureSet(v_i)|$
    End
    End

Step 4: If $V = \emptyset$ Then STOP (a valid k-coloring is
found). Otherwise go to Step 2.
```

2.1 Application of DSATUR

In this section, we apply the DSATUR algorithm on the simple graph G of 9 vertices $v_1, v_2, ..., v_9$, given in Figure 1. In the beginning all the vertices are of degree of saturations $DSAT(v_i)_{i=1,...,9}$ equal to 0.

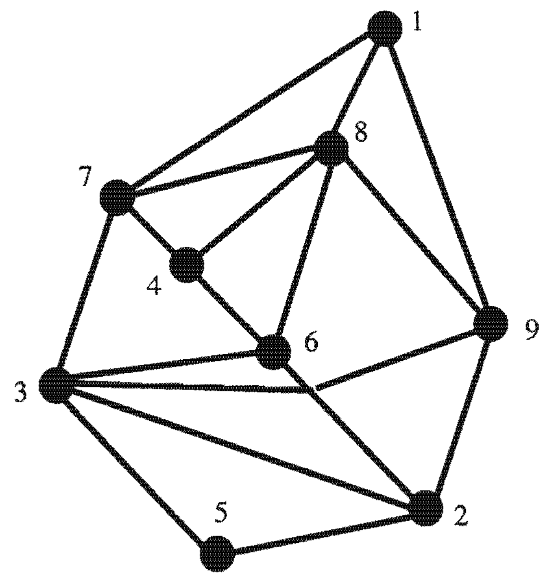

Figure 1: A simple graph of 9 vertices.

The algorithm starts first to color the vertex v_3 of maximal degree equal to 5 by the color c_1 as all the vertices are not yet saturated. The adjacent vertices to v_3 are v_2, v_5, v_6, v_7, v_9, they will be saturated by the color c_1. The second vertex to color is v_2, which is of maximal degree 3 (as all the current saturated vertices are of the same maximal degree of saturation $DSAT = 1$) It will have the color c_2, and its adjacent vertices, v_5, v_6, v_9, will be saturated by the color c_2. The third vertex of *DSAT* maximal and degree maximal is v_6 (with $DSAT(v_6) = 2$ and $\deg(v_6) = 2$). We assign to it the color c_3, and its adjacent vertices v_4 and v_8, will be saturated by the color c_3. We don't need at this stage to add an extra color as we don't have non-colored vertices which are adjacent to all colored vertices v_2, v_4 and v_6. Of the remaining non-colored vertices, we have v_9 of maximal *DSAT* and maximal degree (with $DSAT(v_9) = 2$ and $\deg(v_9) = 2$), it will have the color c_3, and its adjacent vertex v_1 will be saturated by c_3 (the vertex v_8 is already saturated by c_3). The sole non-colored vertex of *DSAT* maximal is v_5 (with $DSAT(v_5) = 2$). It will be also assigned the color c_3. Note that v_5 has no adjacent vertices in the induced sub-graph of non-colored vertices. The next candidate vertex to color is v_7 of maximal degree 3 (as all the non-colored vertices, v_1, v_4 and v_8 are all of the same maximal *DSAT* equals to 1). It will have the first possible color c_2. Its adjacent vertices v_1, v_4, and v_8, will be saturated by the color c_2. The only possible color that we can assign to v_8 of maximal degree 2, is c_1; consequently, its adjacent vertices v_1 and v_4 will be saturated by c_1, and they should be colored by a new color c_4, as they are all saturated by the colors c_1, c_2, and c_3. The order of saturation of the vertices is as follows: $v_3, v_2, v_6, v_9, v_5, v_7, v_8, v_1, v_4$ and the valid approximate 4-coloring for the graph G is the following: $\{(v_3, v_8), (v_2, v_7), (v_5, v_6, v_9), (v_1, v_4)\}$.

Note that the DSATUR algorithm starts first to color the vertices of the maximum clique presents in the graph. In fact, after coloring the first vertex, DSATUR colors, with a new color, a vertex adjacent to it, then searches a vertex adjacent to both colored vertices and assigns to it a new color, and so on. It is clear that the number of used colors k will be greater than or equal to the size of the maximum clique $MaxSize_c$ of the graph. Thus, if the algorithm generates a valid k-coloring with k equal to *MaxSizec*, then this k-coloring is optimal, and we have $k = X(G)$.

3- Modified DSATUR Algorithm

In this section, we show how the least saturated vertex-first heuristic can improve dramatically the efficiency of the DSATUR algorithm. This modification consists of coloring the selected vertex with the color that saturates the least number of non-colored vertices whenever possible. For example, if we have to color a vertex v_i with c_1 and c_2 (with $c_1 < c_2$), and there is a non-colored vertex v_k already saturated by c_1 and not yet by c_2, and the color of v_i will affect v_k, we should color v_i with c_2 and not with c_1 even if c_1 is lower than c_2. This choice will relax the vertex v_k and avoid its being saturated by both colors c_1 and c_2. We then adjust the *Step 4* of the DSATUR algorithm taking into consideration the least saturated vertex-first heuristic.

Step 4:

If v_i is saturated by all used colors or v_i is the first vertex to color Then

assign to it the first color c from its interval *ColorInterval*(v_i).

Else color v_i with the color c that saturates the least number of non-colored vertices.

3.1-Application of DSATUR with Modification

As we have seen in the previous section, the DSATUR algorithm selects a candidate vertex v to color based first on its saturation degree *DSAT(v)* and on its adjacency degree "deg(v)" second. It then assigns to this vertex v the first possible color c from its set of colors *ColorInterval*(v) (i.e., the minimum possible value) regardless of the effect of this choice on the other non-colored vertices not yet saturated by the color c. In other words, the algorithm doesn't investigate all the possibilities to color the selected vertex v. Some selected vertices, however, may be colored with more than one possible color as, for example, the vertex v_7 of the graph of Figure 1, which could be colored with the colors c_2 and c_3. The DSATUR algorithm has blindly assigned to v_7 the first color c_2 (as c_2 is lower than c_3), and has totally ignored the color c_3. However, coloring v_7 with c_3 leads to a valid 3-coloring instead of a valid 4-coloring generated previously by DSATUR. In fact, assigning c_3 to v_7, relaxes its adjacent vertices v_1, v_4, and v_8, which are already saturated by c_3 and not yet saturated by c_2. The least saturated vertex-first heuristic enforces DSATUR to color v_7 with c_3 and not with c_2. As concerning the previously colored vertices v_3, v_2, v_6, v_9, v_5, the algorithm had only one possibility to color them, so the least saturated vertex-first heuristic cannot be used at this stage. The next vertex to color which is of maximal *DSAT* and maximal degree is v_8.

It can be colored with either c_1 or c_2. As the non-colored vertices v_1 and v_4 are not yet saturated by c_1 and c_2, we can assign to v_8, either the color c_1 and both v_1 and v_4 will have the color c_2, or the color c_2 and both v_1 and v_4 will have the color c_1. Thus a valid 3-coloring is generated for the same graph, which is the following: $\{(v_3, v_8), (v_1, v_2, v_4), (v_5, v_6, v_7, v_9)\}$. In addition, this 3-coloring is also optimal for this graph as it has a maximum clique of size 3.

4-DSATUR for Bipartite Graphs

In this section we show that the DSATUR algorithm can be used efficiently to check if a given graph is bipartite or not.

Definition **1**
A graph $G = (V,E)$ is called bipartite if its set of vertices $V(G)$ can be partitioned into two disjoint non-empty and independent sets $V_1(G)$ and $V_2(G)$ such that an edge of $E(G)$ connects only a vertex of $V_1(G)$ with a vertex of $V_2(G)$.

Definition **2**
A graph is called a cycle of length n, and denoted by C_n (with $n > 2$), if it has n vertices v_1, v_2, ..., v_n, and n edges $\{v_1, v_2\}$, $\{v_2, v_3\}$, ..., $\{vn-_1, v_n\}$, $\{v_n, v_1\}$.

Proposition **1**
A graph is not bipartite if DSATUR algorithm colors it with more than two colors.

Proof: The DSATUR algorithm assigns to a vertex v a new color if v is adjacent to a colored vertex (except for the first vertex), and as we have seen, DSATUR searches to color first the clique of maximal size. Thus, if DSATUR generates a valid k-coloring (with k>2), this means the graph contains a cycle C_n of odd length (with $n = (m \bmod 2) + 1$ and m>3). Thus, the graph is not bipartite, as any odd length cycle is not bipartite.

Conclusion

We have improved the performance of Brelaz's DSATUR algorithm for graph coloring by adding to it a simple heuristic called the least saturated vertex-first. This heuristic consists of investigating all the possible colorings, whenever possible, and assigns then to the selected vertex the color that saturates the least number of non-colored vertices. Thus, DSATUR with this heuristic can generate valid k-colorings better than those generated without the heuristic, as shown in the paper. In addition, this heuristic is not expensive as the complexity of the algorithm is still $O(n^2)$. We have also shown that the DSATUR algorithm can be used to check if a graph is bipartite or not.

References

Brelaz, D.(1979) New Method to Color the Vertices of a Graph. *Communications of the ACM* **22**(4): 251-256.

Christofides, N. (1971) An Algorithm for the Chromatic Number of a Graph. *Computer Journal* **14**: 38-39.

Chams, M., Hertz, A. and **de Werra, D.** (1987) Some Experiments with Simulated Annealing for Coloring Graphs. *European Journal of Operational Research* **32**(2): 260-266.

Culberson, J., Beacham, A. and **Papp, D.** (1995) Hiding our Colors. *Proceedings of CP'95-International Workshop, Studying and Solving Really Hard Problems (SSRHP),* Cassis, France, 31-42.

Culberson, J.C.(1993) Iterated Greedy Graph Coloring and the Difficulty Landscape. *Technical Report, TR 90-07,* Univ. of Alberta.

Garey, R. M. and **Johnson, D.S.** (1979) *Computers and Intractability: A Guide to the Theory of NP—Completeness.* Freeman and Company, New York.

Hertz, A. and **de Werra, D.** (1987) Using Tabu Search Techniques for Graph Coloring. *Computing* **39**(4): 345-351.

Jaam, J. (1995) Ramsey Numbers by Stochastic Algorithms with New Heuristics. *Combinatorics and Computer Science.* Springer Verlag, LNCS 1120: 161-181.

Jaam, J. (1995) Coloring of Hypergraphs Associated with Ramsey Numbers by Stochastic Algorithms. *Proceedings of 4th Twente Workshop on Graphs and Combinatorial Optimization,* Twente, The Netherlands, 153-157.

Jaam J., Fliti, T. and **Hussain, D.** (1995) New Bounds of Ramsey Numbers Via a Top-Down Algorithm. *Proceedings of CP'95-International Workshop, Studying and Solving Really Hard Problems (SSRHP),* Cassis, France, pp. 110-118.

Johnson, D.S. (1974) Worst-Case Behavior of Graph-Coloring Algorithms. *Proceedings of 5th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pp. 513- 528.

Joslin, D.E. and **Clements, D.P.** (1999) Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research* **10**: 353-373.

Karger, D., Motwani, R. and **Sudan, M.** (1998) Approximate Graph Coloring by Semi-Definite Programming. *Journal of the ACM* **45**(2): 246-65.

Marshall Hall, J.R.(1986) *Combinatorial Theory* (Second Edition). John Wiley and Sons.

Matula, D.W. and **Beck, L.L.** (1983) Smallest-Last Ordering and Clustering and Graph Coloring Algorithms. *Journal of The ACM* **30**(3): 417-427.

Prestwich, S.D. (2001). Local Search and Backtracking vs Non-Systematic Backtracking. *AAAI 2001 Fall Symposium on Using Uncertainty within Computation.*

Prestwich, S.D. (1998) Using an Incomplete Version of Dynamic Backtracking for Graph Colouring. *CP'98 Workshop on Large Scale Combinatorial Optimisation and Constraints, Electronic Notes in Discrete Mathematics,* Vol. 1.

Skiena, S. (1990). Finding a Vertex Coloring. §5.5.3 *In: Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.* Addison-Wesley, Reading, MA, 214-215.

Wilf, H. (1984). Backtrack: An O (1) Expected Time Algorithm for the Graph Coloring Problem. *Information Processing Letter* **18**: 119-121.

Ref. 2200
Received 07/09/2002.
In revised form 08/02/2003