01 Jan 1989

An Improved Exact Graph Coloring Algorithm

Thomas J. Sager

Shi-Jen Lin

AN IMPROVED EXACT GRAPH COLORING ALGORITHM

Thomas J. Sager and Shijen Lin

CSc-89-1

Department of Computer Science

University of Missouri-Rolla

Rolla, Missouri 65401 (314)341-4491

AN IMPROVED EXACT GRAPH COLORING ALGORITHM

by

THOMAS J. SAGER and SHIJEN LIN

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF MISSOURI - ROLLA
ROLLA, MO 65401

KEYWORDS:

graph-coloring, scheduling, chromatic number, complexity of algorithms, heuristic algorithms.

ABSTRACT:

We present two algorithms for exact graph coloring of the vertex sequential with dynamic reordering of vertices variety. The first, W-DEG, is a straight-forward improvement on Korman's original algorithm. The second, SWAP2, is a not so straight forward improvement on Korman's algorithm and appears to offer the best performance of known exact graph coloring algorithms.

INTRODUCTION:

The graph coloring problem can be stated as: Given an undirected graph, G = (V,E) with no loops or multiedges find a function f: V -> {1..n} for some positive integer n such that if (v,w) in E then f(v) <> f(w). Such a function, f is called a coloring function. If n is minimal over all coloring functions then f is called an exact coloring function. The minimal value of n is the chromatic number and is written χ(G). An algorithm, A, which given input G guarantees an output which is an exact coloring function is called an exact graph coloring algorithm. An algorithm whose output is a coloring function which is not necessarily exact is called an heuristic coloring algorithm.

Exact graph coloring is known to be NP-complete. Indeed, it has been shown that heuristic graph coloring within a factor of 2 of exact graph coloring is NP-Complete [2]. The performance on large graphs of known exact coloring algorithms has been very disappointing. Many known graph coloring algorithms have both an exact and a heuristic form.

Graph coloring can be applied to solve scheduling problems with constraints of the form: events e and e' can not be scheduled together. A classical problem to which graph coloring may be applied is:

Find the minimum number of periods in which a set of examinations can be scheduled under the constraint that examinations v and w can not be scheduled in the same period if $h(v) \cap h(w) \neq \emptyset$ where h(v) is the set of people who will take examination v. Here V is the set of examinations and (v,w) in E iff $h(v) \cap h(w) \neq \emptyset$.

Graph coloring algorithms generally fall into four categories: vertex sequential, color sequential, dichotomous search, and integer linear programming. A study of exact algorithms by Korman [3] found that vertex sequential exact algorithms usually give the best performance in practice and among the vertex sequential algorithms those that use dynamic reordering appear to be superior. Similar results were also found by Kubale and Jackowsky [4]. In addition, a recent study by Campers et al. [1] found vertex sequential with dynamic reordering of vertices among the best performing heuristic algorithms too[1]. Figure 1 shows the generic vertex sequential algorithm with dynamic reordering of vertices.

Vertex sequential algorithms for exact graph coloring were first proposed by Korman [3]. Korman suggested choosing a v_i of maximal c-degree in statement L1 of Figure 1. where

$$\text{c-degree}(v) = | \{f(j) \mid j \text{ in } 1..i-1 \ \& \ (v_j, v) \text{ in } E\} |$$

In statement L2, Korman assigns values to k in increasing order.

A STUDY OF TWO EXACT VERTEX SEQUENTIAL GRAPH COLORING ALGORITHMS WITH DYNAMIC REORDERING:

In the folowing discussion, we consider that the vertices of a graph are originally named 1, 2, 3, ... As colors are created they are named -1, -2, -3, ... A partially colored graph contains uncolored vertices with names from the positive integers and colored vertices with names from the negative integers. We let C be the set of colored vertices and W (white) be the set of uncolored vertices. C-adj(v) is the set of colored vertices to which v is adjacent and c-degree(v) is the cardinality of c-adj(v). Similarly, w-adj(v) is the set of uncolored vertices to which v is adjacent and w-degree(v) is the cardinality of w-adj(v). There is never more than one vertex of a particular color in a partially colored graph. However, we let vertices(v) stand for the set of names of the vertices of the original graph

that have been merged together to form the vertex v of the partially colored graph. Originally, vertices(v) = {v} for all vertices.

In this paper we look at two exact coloring algorithm with dynamic reordering. The first, W-DEG, contains a straight forward improvement on Korman's procedure for choosing a vertex in

---

[1] Camper's CSG and CSGI algorithms are vertex sequential with dynamic reording of vertices.

statement L1 of Figure 1. The second, SWAP2, is somewhat less straight forward but appears to outperform W-DEG and all other known exact graph coloring algorithms.

In W-DEG, in statement L1 of Figure 1, v_i is chosen first to maximize c-degree(v_i) as suggested by Korman. Ties are then broken by maximizing w-degree(v_i). Further ties are broken arbitrarily. The result here is that we use the tie-breaker to maximize the number of edges connecting the colored and uncolored components of the graph. In our implementation of statement L2, values are assigned to k in an arbitrary order except that c+1 is never assigned to k before a lower number.

In SWAP2, we introduce the posibility of uncoloring a vertex that has already been colored, if we can find two adjacent uncolored vertices that are both adjacent to all the colored vertices except the one that is being uncolored. These two uncolored vertices are then colored in place of the one colored vertex that has been uncolored. This action is called a swap. It is performed, if possible, only when there are no uncolored vertices adjacent to all the colored vertices. If more than one swap is possible, we choose a swap that maximizes the number of edges connecting the colored and uncolored components of the graph. Where swaps are not possible, SWAP2 behaves the same as W-DEG. The details of SWAP2 are shown in Figure 2.

METHODOLOGY:

Both algorithms where programmed in Turbo Pascal version 4.0 and run on an IBM 6152 workstation under DOS. 100 random graphs of each of several characteristics were generated using the minimal standard random number generator of Park and Miller [5] with the primary author's social security number as the original seed. The characteristics used were N <- 28 to 56 by 4 and D <- 0.1 to 0.9 by 0.2, where N is the cardinality of the vertex set and D is the edge density. The cardinality of the edge set of a graph is round(D * N * (N-1) / 2).

Each time an algorithm was applied to a graph we generated two statistics: time, the number of seconds used by the algorithm and moves, the number of calls to the recursive procedure, Color. Moves is independent of the implementation except in so far as an arbitrary choice has been made. However, certain moves are more time consuming then others. In particular, in procedure Choose, searching for a swap move has time complexity $O(n^2)$, whereas the other parts of Choose and functions Swap, Merge and Newcolor have complexity O(n). Time, on the other hand, is extremely dependent on hardware, software tools and programming implementation. Our implementations of the two algorithms are similar. No attempt to optimize in any manner was made. Thus, the time statistic should be used only for comparison between the two algorithms and not in any absolute sense.

For each 100 graphs of characteristics N and D and each of the parameters time and move, we computed the mean of each algorithm, the ratio mean(SWAP2)/mean(W-DEG) and the p-value. The p-value is computed using the paired-t test and represents the theoretical probability of observing a mean(W-DEG - SWAP2) for a random sample of size 100 greater than or equal to the mean(W-DEG - SWAP2) of the observed sample of size 100 subject to the hypothesis that mean(W-DEG - SWAP2) = 0 over the entire population. These statistics and mean(χ(G)) are summarized in Tables 1 through 3.

CONCLUSIONS:

For graphs of vertex size 28 through 40 with a density of 0.1, W-DEG outperforms SWAP2. However, at the other sizes and densities tested SWAP2 outperforms W-DEG, sometimes by as much as 32% in moves and 28% in time. Typical savings for graphs with N between 40 the 56 and D between 0.3 and 0.9 appear to be around 24% in moves and 19% in time. In most cases p-values are less than 0.05 and in many case 0 to three decimal places, but there are exceptions.

REFERENCES:

[1] Campers, G., Henkes, O. and LeClerq, J.P.: Graph coloring heuristics: a survey, some new propositions and computational experiences on random and Leighton's graphs., Operational Research '87, Proc. 11th Intl. Conf, Buenos Aires, Aug. 1987, pp917-32.

[2] Garey, M.R. and Johnson, D.S.: The complexity of near-optimal graph coloring., J. ACM, **23**, 1, Jan. 1976, pp43-9.

[3] Korman, S.M.: The graph coloring problem., in **Combinatorial Optimization**, Ed. N. Christofides et al., Wiley, New York 1979, pp211-235.

[4] Kubale, M. and Jackowski, B.: A general implicit enumeration algorithm for graph coloring., Commun. ACM, **28**, 4, April 1985, pp412-418.

[5] Park, S.K. and Miller, K.W.: Random number generators: good ones are hard to find., Commun. ACM, **31**, 10, Oct. 1988, pp1192-201.

```
ALGORITHM DYNAMIC_VS;

  input:    G = (V,E): graph;
  output:   ub: positive integer; exactf: function:V -> 1..ub;

  procedure Color(i, c);
    if i > |V| then exactf <- f; ub <- c;
    else
L1:   choose a vertex from V - {v ..v   } call it v .
      feasibleset <- {1..min(c+1,ub-1)} -
                {f(j) | j in 1..i-1 & (v ,v ) in E}
L2:   forall k in feasibleset do f(i) <- k; Color(i+1, max(k,c));
  end Color;

  ub <- upper bound χ(G) + 1;
  Color(1, 0);

end algorithm;
```

FIGURE 1. GENERIC VERTEX SEQUENTIAL EXACT ALGORITHM WITH DYNAMIC REORDERING

```
ALGORITHM SWAP2;

  input:    G = (V,E): graph;
  output:   ub: positive integer; exactf: function: V -> 1..ub;

  function  Choose(G:  graph):  W ∪ (W x W);
      -- W is the set of uncolored vertices of G
      -- C is the set of colored vertices of G
    S <- {v in W | c-degree(v) is maximal}
    if c-degree(v) = c-1, forall v in S then
      T <- {(v,w) | v in S,  w in S, (v,w) in E and
                        c-adj(v) = c-adj(w)}
      if T <> Ø then
        return member {(v,w) in T |
          w-degree(v) + w-degree(w) - {w-degree(k) is maximal}
          where k is the colored vertex not adjacent to v and w.
    return member {v in S | w-degree(v) is maximal};
  end Choose;

  function Merge(G: graph; v: G.W; k: G.C): Graph;
    with G do
      forall w in V | (w,v) in E do E <- E - {(v,w)} ∪ {k,w};
      V <- V - {v}
      vertices(k) <- vertices(k) ∪ vertices(v);
    return(G);
  end Merge;
```

Figure 2. ALGORITHM SWAP2

```
ALGORITHM SWAP2;   -- continued

  function Newcolor(G:graph, v: G.W);
    with G do
      rename vertex v,  -(c+1);
      forall i in -c..-1 do E <- E ∪ {(-(c+1),i)};
    return(G);
  end Newcolor;

  function Swap(G: graph, v,w: G.W): graph;
    with G do
      k <- y | y in C and (v,y) notin E} );
      rename vertex v and k, k and v respectively;
      G <- Newcolor(G,w);
    return(G);
  end Swap;

  procedure Color(G, c);
    if G.W = ∅ then
      ub <- c;
      for i := 1 to c do
        forall v in vertices(-i) do exactf(v) <- i;
    else
      v <- Choose(G);
      if v in W then
        feasibleset <- {k in -c..-1 | (k,v) notin E}
        forall k in feasibleset do
          if c < ub then G <- Merge(G,v,k);  Color(G, c);
        if c < ub-1 then G <- Newcolor(v); Color(G, c+1);
      else -- v in W x W
        G <- Swap(G, w, x) where w and x are the components of v;
        Color(G, c+1);
  end Color

  ub <- upper bound χ(G) + 1;   -- |V|+1 will do
  name the members of G.V: 1,2,3...
  forall v in G.V do vertices(v) <- {v};
  color(G, 0);

end SWAP2;
```

Figure 2. **ALGORITHM SWAP2 (continued)**

D \ N	28	32	36	40	44	48	52	56
0.1:	3.01	3.02	3.09	3.16	3.54	3.94	4.00	4.01
0.3:	5.01	5.01	5.35	5.93	6.00	6.03	6.74	7.00
0.5:	6.94	7.30	7.97	8.28	8.96	9.12	9.94	10.02
0.7:	9.61	10.38	11.09	12.04	12.81	13.37	14.04	14.83
0.9:	15.29	16.46	17.99	19.32	20.52	21.94	23.11	24.45

TABLE 1: MEAN χ(G).

D \ N		28	32	36	40	44	48	52	56
0.1	SWAP2 MEAN	28.9	35.0	41.5	53.5	66.8	60.2	61.0	61.9
	W-DEG MEAN	28.9	34.9	42.7	54.0	72.4	64.2	65.3	65.9
	SWAP2/W-DEG	1.000	1.002	0.973	0.991	0.922	0.938	0.935	0.939
	P-VALUE	0.514	0.627	0.070	0.369	0.005	0.000	0.000	0.000
0.3	SWAP2 MEAN	34.2	47.5	118	129	170	779	222E1	185E1
	W-DEG MEAN	38.8	57.0	154	165	201	954	277E1	246E1
	SWAP2/W-DEG	0.883	0.883	0.767	0.779	0.848	0.817	0.800	0.753
	P-VALUE	0.000	0.000	0.000	0.000	0.006	0.010	0.000	0.000
0.5	SWAP2 MEAN	48.8	118	177	721	121	608E1	106E2	327E2
	W-DEG MEAN	61.0	156	257	976	149	775E1	148E2	481E2
	SWAP2/W-DEG	0.800	0.755	0.690	0.738	0.813	0.784	0.716	0.679
	P-VALUE	0.000	0.000	0.000	0.000	0.000	0.004	0.000	0.020
0.7	SWAP2 MEAN	55.8	131	281	610	210E1	715E1	189E2	804E2
	W-DEG MEAN	72.5	190	407	820	273E1	993E1	260E2	976E2
	SWAP2/W-DEG	0.769	0.688	0.690	0.744	0.769	0.720	0.728	0.824
	P-VALUE	0.000	0.002	0.000	0.000	0.000	0.000	0.004	0.079
0.9	SWAP2 MEAN	28.9	35.8	503	67.6	110	328	790	337E1
	W-DEG MEAN	30.1	40.3	582	85.5	162	467	106E1	448E1
	SWAP2/W-DEG	0.962	0.888	0.864	0.791	0.678	0.701	0.739	0.754
	P-VALUE	0.005	0.000	0.000	0.000	0.000	0.000	0.011	0.050

TABLE 2: MEAN OF MOVE STATISTIC FOR ALGORITHMS SWAP2 AND W-DEG, RATIO OF MEANS AND P-VALUE.

D \ N		28	32	36	40	44	48	52	56
0.1	SWAP2 MEAN	0.148	0.193	0.259	0.362	0.518	0.511	0.560	0.578
	W-DEG MEAN	0.145	0.186	0.253	0.347	0.532	0.522	0.578	0.588
	SWAP2/W-DEG	1.019	1.041	1.024	1.046	0.973	0.979	0.967	0.983
	P-VALUE	0.811	0.988	0.879	0.937	0.208	0.115	0.019	0.194
0.3	SWAP2 MEAN	0.254	0.389	1.17	1.46	2.01	10.3	33.7	31.1
	W-DEG MEAN	0.276	0.442	1.45	1.77	2.26	12.0	39.8	39.1
	SWAP2/W-DEG	0.919	0.882	0.810	0.826	0.888	0.858	0.847	0.795
	P-VALUE	0.004	0.002	0.001	0.000	0.040	0.040	0.000	0.000
0.5	SWAP2 MEAN	0.453	1.28	2.27	10.2	19.2	104	200	634
	W-DEG MEAN	0.534	1.60	3.07	13.1	22.4	124	264	881
	SWAP2/W-DEG	0.848	0.795	0.737	0.781	0.855	0.833	0.760	0.719
	P-VALUE	0.000	0.000	0.000	0.000	0.004	0.017	0.000	0.033
0.7	SWAP2 MEAN	0.584	1.64	4.00	9.83	37.8	142	400	183E1
	W-DEG MEAN	0.716	2.23	5.37	12.4	46.01	184	515	209E1
	SWAP2/W-DEG	0.816	0.735	0.745	0.790	0.822	0.773	0.778	0.877
	P-VALUE	0.000	0.005	0.002	0.000	0.000	0.000	0.013	0.150
0.9	SWAP2 MEAN	0.260	0.386	0.617	0.963	1.75	5.71	15.6	70.0
	W-DEG MEAN	0.263	0.412	0.692	1.15	2.50	7.59	19.51	46.5
	SWAP2/W-DEG	0.992	0.938	0.893	0.840	0.702	0.753	0.802	0.810
	P-VALUE	0.356	0.004	0.003	0.000	0.000	0.000	0.044	0.056

TABLE 3: MEAN OF TIME (IN SECONDS) STATISTIC FOR ALGORITHMS SWAP2 AND W-DEG, RATIO OF MEANS AND P-VALUE.