Programming Techniques and Data Structures

Ellis Horowitz Editor

A Generalized Implicit Enumeration Algorithm for Graph Coloring

MAREK KUBALE and BOGUSŁAW JACKOWSKI

ABSTRACT: A generalized algorithm for graph coloring by implicit enumeration is formulated. A number of backtracking sequential methods are discussed in terms of the generalized algorithm. Some are revealed to be partially correct and inexact. A few corrections to the invalid algorithms, which cause these algorithms to guarantee optimal solutions, are proposed. Finally, some computational results and remarks on the practical relevance of improved implicit enumeration algorithms are given.

1. INTRODUCTION

It is well known that the problem of coloring the vertices of a graph with a minimum number of colors so that no adjacent vertices are the same color is probably intractable. Not only is the general problem NP-complete, but even a drastically simplified problem of determining the 3-colorability of a 4-regular planar graph remains NP-complete [4, 7]. Also, the problem of determining the chromatic number of an arbitrary graph within a worst-case ratio of less than 2 has been shown to be NP-complete [6]. That is why no polynomial time algorithm for exact graph coloring is likely to exist.

In general, there are three tree search strategies to solve this classical combinatorial problem: maximal independent sets, dichotomous search, and implicit enumeration. The most flexible and space efficient of these is the *implicit enumeration* (*backtracking*) approach. The first coloring algorithm of this type is due to Brown [2]. The main idea of his method is based on the following simple backtracking rule. Suppose that the vertices of graph G have been ordered in some way and are reindexed so that v_i is the ith vertex in this ordering. Then, two steps: *forward* and *backward*, are performed alternately. The forward step colors the vertices sequentially up to a vertex which cannot be colored because of a lack of colors available to it. The backward step moves sequentially back in search of the first vertex which can be colored with another feasible color and then the forward step is resumed, etc. If a new, better coloring of G has been found, the algorithm attempts to find the next one. The activity terminates when a backtrack reaches v_1.

The forward step of Brown's original algorithm can be improved either by using a look-ahead procedure [2] or by reordering yet uncolored vertices [9]. On the other hand, the backward step turns out to be much harder to refine. Christofides [3] was the first to attempt to improve on the backward step. Unintentionally, he obtained an algorithm which does not always find a chromatic coloration and which, moreover, fails to terminate properly on some graphs. Brélaz [1], who made a few errors and also arrived at a partially correct approximation algorithm, was the next to improve Brown's coloring algorithm. A simple counterexample to both algorithms is shown in Figure 1. Brélaz's modification was corrected independently by Kubale and Kusz [10] and Peemöller [11]. (Peemöller's correction seems to be closer to Brélaz's intention.)

This article is motivated by the above-mentioned errors. We begin by presenting, in the next section, a concept of a generalized (frame) implicit enumeration algorithm for graph coloring. In Section 3 we give a survey of various instances of the general algorithm and discuss its correctness and efficiency. Among them we formulate corrections for Christofides' and Brélaz's concepts in terms of the general algorithm. It should be noted that although both algorithms presented in the article differ from the originals, we use the appelations "Christofides' algorithm" and "Brélaz's algorithm." Nevertheless, the main ideas remain unchanged and, therefore, such terminology seems fair to the originators. In Section 4 we give computational results obtained by applying implicit enumeration algorithms to coloring an identical series of graphs generated at random. We conclude, in Section 5, with some remarks on the practical relevance of the improved graph coloring algorithms.

2. A GENERALIZED IMPLICIT ENUMERATION ALGORITHM

In principle, the generalized algorithm is based on Brown's original approach. Both basic steps, however, are more sophisticated and require an additional description. As we know, the forward step colors the vertices sequentially within a prescribed number of colors up to a vertex which cannot be colored because of a lack of feasible colors. Such a vertex is referred to as a *resumption point* for the backward step. More precisely, procedure FORWARDS successively associates the sets of *feasible colors* (FC) with the vertices $v_r, v_{r+1}, \ldots,$ and colors each of them with the first member of the respective set. Roughly, the invariant for this procedure is: "Graph G can be colored with ub colors and the vertices $v_1, \ldots, v_{r-1}$ are colored with colors less than ub" where ub is an upper bound on the *chromatic number* $\chi(G)$. This step terminates when either a vertex with the empty set of feasible colors has been encountered or all the vertices have been colored. The latter means that a new complete coloring with less than ub colors has been found and the upper bound can be diminished. Eventually, the value of variable r is changed to keep the following invariant for procedure BACKWARDS: "Graph G can be colored with ub colors, the vertices $v_1, \ldots, v_{r-1}$ can be colored with colors less than ub whereas v_r cannot be colored with a color less than ub." Procedure BACKWARDS searches for the highest numbered vertex among $v_1, \ldots, v_{r-1}$, say v_i, the recoloring of which might have an influence on the coloring of $v_r, v_{r+1}, \ldots$. The vertices which are not omitted by the checking process are called *current predecessors* (CP) of v_r. If such a vertex v_i is encountered, the FC(i) set is decreased by v_i's actual color to prevent it from obtaining the same partial coloring. (The vertex is referred to as a *resumption point* for the forward step.) Finally, the assignment $r \leftarrow i$ is performed to keep the invariant for the FORWARDS procedure, which ensues immediately. Otherwise, BACKWARDS is exited with $r = 0$. The algorithm terminates when either ub equals a lower bound on the chromatic number or a backtrack fails to find a vertex that can be recolored.

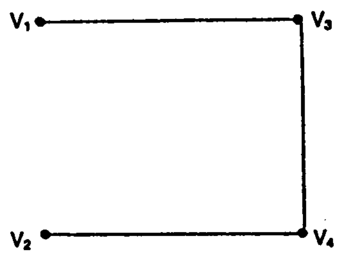

FIGURE 1. A Counterexample to the Christofides and Brelaz Algorithms

Here is a control abstraction for the general implicit algorithm written in SPARKS language [8].

```
procedure IMPLICIT_ENUMERATION (G, n)
// G is any n-vertex graph given implicitly. C(1),  //
// ..., C(n) are used to retain colors assigned to  //
// vertices in the best solution found so far. C'(1), //
// ..., C'(n) comprise a current coloring (partial  //
// or complete). //
 integer n,r,ub, lower_bound, upper_bound, C(1:n),
 C'(1:n)
 set CP, FC(1:n)
 initialize lower_bound, upper_bound and vertex
 order
 CP ← ∅; r ← 1
 ub ← upper_bound + 1
            // to get at least one //
            // complete coloring //
 loop
   call FORWARDS (r)
   if ub = lower_bound then exit endif
   call BACKWARDS (r)
   if r = 0 then exit endif
 repeat
 print('The chromatic number =', ub, 'coloring:', C)
end IMPLICIT_ENUMERATION
```

Both basic procedures are described in the following form.

```
procedure FORWARDS (r)
 global integer n, ub, C(1:n), C'(1:n); global set FC(1:n)
 integer i, r
 for i ← r to n do
   rearrange v_i, ..., v_n
            // only if dynamic //
            // reordering is applied //
   determine FC(i)  // for r = 1 or r < i //
   if FC(i) = ∅ then r ← i; return endif
   C'(i) ← min(FC(i))
             // color v_i with smallest //
             // feasible color //
 repeat
 // a new complete coloring has been found //
 C ← C'  // store the current coloring //
 ub ← max(C)  // update the upper bound on χ(G) //
 r ← least i such that C(i) = ub
end FORWARDS
```

```
procedure BACKWARDS(r)
  global integer n, ub, C'(1:n); global set CP, FC(1:n)
  integer i, r
  determine initial CP
  while CP ≠ ∅ do
    i ← max(CP); CP ← CP − {i}
    update CP  // only if dynamic computation //
               // of CP is applied //
    FC(i) ← FC(i) − {C'(i)}
    if FC(i) ≠ ∅ then r ← i; return endif
  repeat
  r ← 0  // none of current predecessors //
         // can be recolored //
end BACKWARDS
```

The correctness of the general algorithm cannot be proved until the semantics of the statements initialize, determine, and update is defined. This is a crucial point of the algorithm, because by defining feasible colors, current predecessors, and a mode of initialization one can obtain a wide range of coloring algorithms, both exact and approximate. Hence, the general implicit algorithm should be regarded as a frame algorithm, any instance of which requires a separate analysis. Also, the time complexity of the algorithm depends on a particular definition of the above-mentioned notions. However, taking into account only explicit constraints on a solution space, we see that the number of backtracks never exceeds $n!$ regardless of the particular definitions of the FC and CP sets. Thus the worst-case time for an implicit enumeration algorithm will generally be $O((f(n) + b(n))n!)$ where $f(n)$ and $b(n)$ are polynomials determining the complexity of FORWARDS and BACKWARDS, respectively.

3. A SURVEY OF CONCRETE REALIZATIONS

There are four basic algorithms in the family of implicit enumeration methods.

1. Brown's algorithm
2. Christofides' algorithm
3. Brélaz's algorithm
4. Korman's algorithm

In this section we outline the main features of the algorithms in their corrected form with a special reference to the following:

(i) initialization, that is, stating bounds on the chromatic number, initial ordering of vertices, etc.,
(ii) rearrangement of yet uncolored vertices,
(iii) computation of the sets of feasible colors FC(i),
(iv) computation of the set of current predecessors CP.

3.1 Brown's Two Algorithms [2]

3.1.1 Brown's Ordinary Algorithm

(i) The vertices are preordered in a *greedy largest first* (GLF) manner. Namely, v_1 is a maximum degree vertex of G and remaining vertices are ordered by means of the following greedy principle: For every $i = 2, \ldots, n-1$, v_i is adjacent to more of the vertices $v_1, \ldots, v_{i-1}$ than any other vertex v_j, $j > i$ (ties are broken by choosing the vertex of greater degree). The GLF ordering can be accomplished in time $O(\min(n^2, m \log n))$. Default values for lower bound and upper bound are 1 and n, respectively.

(ii) No rearrangement is performed.

(iii) Let PC(i) denote a set of *prohibited colors* for v_i, that is, $\text{PC}(i) = \{c < ub$: there is an adjacent predecessor of v_i colored with $c\}$. Then

$$\text{FC}(i) = \{1, 2, \ldots, \max_{j<i} C'(j) + 1\} - \text{PC}(i) - \{ub\}.$$

To estimate the complexity of FORWARDS notice that determining of FC(i) requires deg(v_i) operations. Therefore, one pass of FORWARDS takes at most $O(m)$ time plus, eventually, $O(n)$ time to store coloration C' and actualize variables ub and r when a better solution has been found. Thus a total of $O(m + n)$ time must be used.

(iv) The definition of CP is trivial, namely

$$\text{CP} = \{1, 2, \ldots, r-1\},$$

where r is a value of parameter carried in to the BACKWARDS procedure. Since no updating is necessary, BACKWARDS requires constant effort for each i and $O(n)$ time in all.

Remarks. Brown's ordinary algorithm finds the lexicographically first exact coloration with respect to the GLF ordering of vertices in the following sense: Out of two different complete solutions $C = C(1), \ldots, C(n)$ and $C' = C'(1), \ldots, C'(n)$, the coloring C is said to be *prior* to C' under a given vertex order if either $C(1) < C'(1)$ or $C(1) = C'(1), \ldots, C(i-1) = C'(i-1)$ and $C(i) < C'(i)$ for some $i = 2, \ldots, n$.

3.1.2 Brown's Algorithm with Look-Ahead

(i) Same as in Section 3.1.1.

(ii) Same as in Section 3.1.1.

(iii) Brown reduced the number of backtracks by introducing the so-called *look-ahead* (LA) procedure to the forward step. LA makes it possible to diminish the FC sets by enlarging the PC sets. The new definition of prohibited colors is $\text{PC}(i) = \{c < ub$: there is an adjacent predecessor of v_i colored with c or any of the adjacent successors of v_i can be colored only with $c\}$. The improved FORWARDS procedure can still run in linear time $O(m + n)$, but within a greater constant of proportionality.

(iv) Same as in Section 3.1.1.

Remarks. The concept of looking ahead can be refined further by ordering feasible colors by the number of *preventions*, that is, the number of adjacent successors of v_i which could possibly be assigned color c unless c is assigned to v_i. However, such a complex LA increases the complexity of FORWARDS to the order of $O(mn)$.

The backtracking algorithm with simple LA finds the first chromatic coloring [5] whereas the solution produced by the complex LA algorithm need not be the lexicographically first one.

3.2 Corrected Christofides Algorithm [3]

(i) No particular ordering is assumed; lower bound and upper bound are determined as in Section 3.1.

(ii) Same as in Section 3.1.

(iii) Same as in Section 3.1.

(iv) In an effort to decrease the number of backtracks, Christofides reduced the CP set too drastically and obtained a partially correct approximation algorithm. To correct his definition of CP we need a notion of path passing throughout increasingly indexed vertices. Such a path is said to be *monotonic*. By the *predecessor* set of v_i we mean $P(i) = \{j < i$: there is a monotonic path from v_j to v_i in $G\}$. Now, the initial CP set is simply defined as

$$\text{CP} = \text{CP} \cup P(r).$$

No updating of the set is performed.
The $P(i)$ sets can be precomputed just after the initial vertex order is stated. For this purpose one can use any efficient method for finding the transitive closure of a directed graph, for example, [12]. Hence BACKWARDS can be efficiently implemented in $O(n)$ time.

Remarks. It is possible to incorporate looking ahead into the Christofides algorithm. This can be done by restricting FC (as in Section 3.1.2) and updating CP according to the formula

$$\text{CP} = \text{CP} \cup \bigcup_{j \in B(i)} P(j) - \{i, \ldots, n\},$$

where $B(i)$ is the set of all uncolored neighbors of v_i which could be assigned the only color c unless c is assigned to v_i. It is convenient to construct the set $B(i)$ while searching for prohibited colors in FORWARDS.

Since the partial solutions of level i are lexicographically ordered, the method finds the first exact coloring (unless a refined version of LA is used).

3.3 Corrected Brélaz Algorithm [1, 11]

(i) The vertices are colored sequentially to obtain a suitable ordering of the vertex set and both bounds on the chromatic number of G. Brélaz suggests preordering the vertices by means of a sequential with interchange algorithm applied to a *saturation largest first* (SLF) ordering or the *Matula-Dsatur* algorithm (a combination of SLF and SL) as he calls it. The SLF algorithm, which is a basis for both heuristics, repeatedly colors an uncolored vertex of the largest saturation degree with the smallest possible color, where the *saturation degree* is the number of adjacent distinctly colored vertices. Ties are broken by choosing the vertex of greater degree. The SLF with interchange algorithm takes $O(mn)$ time and the Matula-Dsatur algorithm needs $O(\min(n^2, m \log n))$ time. Each of the preconditioning algorithms arranges the vertex set so that initial vertices constitute an *initial clique* (IC) of size at least 2 and the number of colors used is usually close to minimum.

(ii) Same as in Section 3.1.

(iii) Same as in Section 3.1.

(iv) Brélaz also tried to reduce the CP set but he made two errors. Herein we give Peemöller's correction to Brélaz's dynamic updating of current predecessors. The set of *adjacent predecessors* of v_i, AP(i), is partitioned into nonempty sets $\text{AP}_a(i), \ldots, \text{AP}_b(i)$ such that $\text{AP}_c(i) = \{j < i$: v_j is colored with $c\}$, $c < ub$. Let $r_c = \min \text{AP}_c(i)$. Then the set of *representatives* for AP(i) is defined as

$$R(i) = \begin{cases} \varnothing & \text{if } \text{AP}(i) = \varnothing, \\ \{r_a, \ldots, r_b\} - \text{IC} & \text{otherwise.} \end{cases}$$

Now the initial set of current predecessors is

$$\text{CP} = \text{CP} \cup R(r).$$

Updating of the set is accomplished in BACKWARDS as follows:

$$\text{CP} = \text{CP} \cup R(i).$$

BACKWARDS can be implemented to run in $O(m + n)$ time.

Remarks. As previously, the LA procedure can be incorporated into the algorithm. In this case additional updating takes the form

$$\text{CP} = \text{CP} \cup \bigcup_{j \in B(i)} R(j).$$

(In constructing ($R(j)$ uncolored vertices are disregarded.) The Brélaz algorithm also finds the first exact coloring in the lexicographic order sense.

3.4 Korman's Dynamic Reordering Algorithm [9]

(i) Same as in Section 3.1.

(ii) Korman noticed that during the forward step some vertices appear to have fewer feasible colors than others and such vertices should be colored first. His *dynamic rearrangement* (DR) rule states simply: From the set $\{v_i, \ldots, v_n\}$ of uncolored vertices choose a vertex which can be colored with the smallest number of feasible colors and replace it with v_i. The complexity of FORWARDS with dynamic rearrangement is dominated by the DR procedure which can run in time $O(m + n)$, but at a cost of some increase of the complexity of BACKWARDS.

(iii) Same as in Section 3.1.

(iv) Same as in Section 3.1.

Remarks. The DR procedure plays a role similar to that of LA. However, empirical investigations show that, in

TABLE I. Computational Results from the Application of Implicit Enumeration Algorithms to Color Random Graphs

		Average time in CPU seconds				
n	*d*	Brown SSL	Christofides SSL	Brelaz SSL	Korman	
					LF	SL
10	0.1	.1	.1	.1	.0	.1
	0.3	.1	.1	.1	.1	.1
	0.5	.1	.1	.1	.1	.1
	0.7	.1	.1	.1	.1	.1
	0.9	.1	.1	.1	.1	.1
20	0.1	.3	.3	.3	.1	.2
	0.3	.4	.4	.4	.2	.2
	0.5	.4	.5	.4	.2	.3
	0.7	.4	.5	.4	.2	.3
	0.9	.4	.5	.5	.2	.3
30	0.1	.6	.7	.6	.3	.5
	0.3	.8	.9	.8	.6	.6
	0.5	5.0	5.2	5.2	1.1	1.9
	0.7	1.7	1.8	1.6	1.6	2.0
	0.9	1.1	1.2	1.1	.4	.6
40	0.1	1.2	1.4	1.2	.8	.9
	0.3	4.4	4.2	4.0	6.9	3.7
	0.5	13.5	12.6	9.5	10.1	5.5
	0.7	45.6	45.2	21.5	5.7	9.1
	0.9	2.2	2.5	2.1	6.9	1.4
50	0.1	1.7	2.0	2.1	1.0	1.3
	0.3	>207.8	>207.5	>207.3	93.8	>150.0
	0.5	∞	∞	∞	>222.5	∞
	0.7	>259.7	>261.4	>250.6	>283.3	>224.6
	0.9	20.1	60.4	11.1	1.1	2.2
60	0.1	3.1	3.3	3.1	1.7	1.9
	0.3	∞	>244.9	>136.4	100.0	>189.6
	0.5	∞	∞	∞	∞	∞
	0.7	∞	∞	∞	∞	∞
	0.9	>194.6	>195.7	>180.3	105.1	86.1

∞ means that all graphs required more than five CPU minutes
> means that at least one graph required more than five CPU minutes.

spite of the same complexity, the DR rule is much more powerful than the simple LA.

Korman's DR algorithm produces a chromatic coloration which is lexicographically first with respect to the final vertex order, that is, the vertex order just after the chromatic coloring had been found.

A modification to the DR rule is also possible. For instance, at restart after BACKWARDS coloring may be done according to the latest arrangement of vertices already colored or starting with the latest resumption point. In any case a particular definition of dynamic rearrangement affects only the efficiency of enumeration since any algorithm with a DR-like procedure leads to an exhaustive search regardless of the way uncolored vertices are ordered.

It is also possible to incorporate Peemöller's dynamic updating of the CP set into the algorithm.

4. COMPUTATIONAL RESULTS

In order to compare a relative speed-up of various improvements of Brown's coloring algorithm on empirical grounds, the implicit algorithms were coded in Pascal and run on an R-32 computer (equivalent to an IBM 360/65). All the algorithms were programmed in the versions of the previous section except the corrected Brélaz algorithm which was implemented in the form of [10]. Since the efficiency of vertex sequential algorithms is highly dependent on the way the vertices are ordered, the following preparatory procedures arranging the vertices by their degree were implemented: *largest first* (LF), *smallest last* (SL), *saturation LF* (SLF), and *saturation SL* (SSL). The saturation procedures tend to use the largest unavoidable color as soon as possible. The LF ordering procedure ran in linear time while the others required at most $O(n^2)$ additional time. All pro-

TABLE I. Computational Results from the Application of Implicit Enumeration Algorithms to Color Random Graphs (*Continued.*)

		Average number of backtracks				
n	d	Brown SSL	Christofides SSL	Brelaz SSL	Korman	
					LF	SL
10	0.1	1	1	1	1	1
	0.3	1	1	1	1	1
	0.5	1	1	1	1	1
	0.7	1	1	1	1	1
	0.9	1	1	1	1	1
20	0.1	1	1	1	1	1
	0.3	2	2	2	1	1
	0.5	4	4	3	2	1
	0.7	3	3	3	1	2
	0.9	2	1	2	1	1
30	0.1	1	1	1	1	1
	0.3	16	5	6	4	2
	0.5	229	228	175	7	13
	0.7	34	33	20	12	14
	0.9	5	4	4	1	1
40	0.1	3	2	3	1	1
	0.3	136	112	84	36	16
	0.5	528	463	238	50	24
	0.7	1379	1368	401	24	25
	0.9	20	19	8	1	2
50	0.1	12	5	12	2	2
	0.3	>5431	>7241	>4896	326	>493
	0.5	>5700	>8703	>5670	>751	>992
	0.7	>8872	>8927	>5691	>943	>774
	0.9	468	468	132	1	2
60	0.1	16	11	14	3	2
	0.3	>7199	>8285	>4775	288	>520
	0.5	>5751	>6911	>5672	>838	>734
	0.7	>4425	>5565	>3478	>714	>706
	0.9	>4245	>4183	>2355	229	192

∞ means that all graphs required more than five CPU minutes
> means that at least one graph required more than five CPU minutes.

grams were executed on identical samples of pseudo-random graphs generated according to the constant density model with vertex number $n = 10(10)60$ and graph density $d = .1(.2).9$. For each n and d three graphs were generated. Each was colored by each program within a five minute limit of CPU time. Table I contains the best computational results of the four orderings obtained for each program in question. Aside from average timing we give the average number of backtracks required to color a graph since this parameter seems to be a more objective measure of algorithmic complexity, especially if a program terminates before the deadline.

In practice, graphs to be colored are usually sparse in the sense that the number of edges m is much less than $n^2/2$. For this reason Figure 2 shows timing profiles of the best and worst variants of the analyzed algorithms plotted for graphs G with $d = .1$.

For more details on the extensive experimentally observed computing times comparing various backtracking algorithms, see [10].

5. CONCLUDING REMARKS

Results of the empirical investigations allow us to draw the following conclusions.

1. Small graphs with $n \leqslant 30$ can be colored efficiently by any backtracking algorithm preceded by any vertex ordering procedure. Nevertheless, simple methods, such as Brown's with LF, are preferable.

2. The effectiveness of this family of coloring algorithms decreases as the density of the graphs increases, except for very dense graphs.

3. The efficiency of implicit enumeration algorithms without dynamic reordering is highly dependent on the initial arrangement of vertices. In this case the SSL

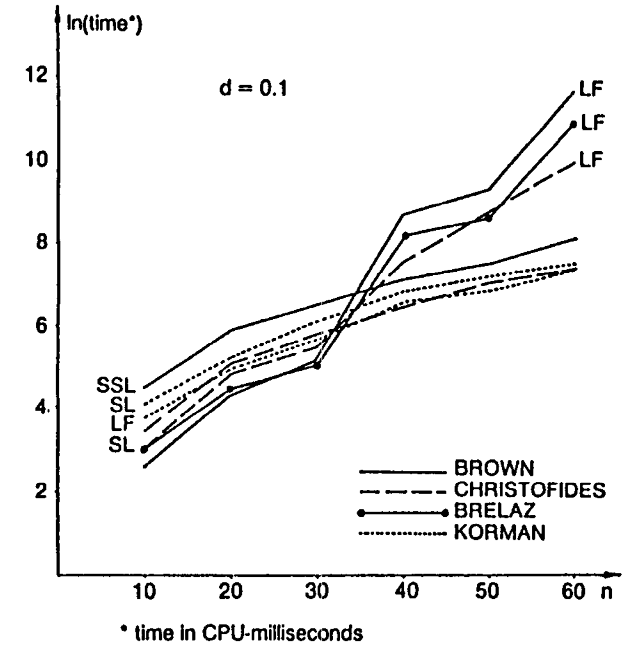

FIGURE 2. Timing Profiles from the Application of Backtracking Algorithms to Color Random Sparse Graphs

ordering performs best overall and saves more backtracking than any improvement of the forward or backward step.

4. Regarding the expectation of the time complexity, the dynamic reordering of yet uncolored vertices establishes a great improvement even if compared to the methods with LA.

5. If graphs to be colored are sparse, then both the Christofides algorithm, applied to the vertices presorted by SL, and Korman's dynamic reordering, preceded by LF, can be recommended.

6. If graphs to be colored are not sparse, Korman's algorithm is decidedly superior to the others. In particular, Korman's algorithm with LF preordering runs faster on middle-density graphs whereas the method with SL is preferable for dense inputs.

Finally, we recall that our testing was carried out on a series of random graphs. However, graphs arising in practice may differ from our constant density samples. For example, if a graph is disconnected or separable, then Christofides' and Brélaz's algorithms may be superior to the others. This follows from the fact that these methods are, in a sense, the most sensitive to the connectedness of a graph among the implicit enumeration algorithms for graph coloring. Moreover, in real-life problems there are usually additional restrictions that must be taken into consideration. These restrictions may be reflected in some unavailability constraints imposed on the FC sets before coloring begins (e.g., such a situation takes place when constructing class-teacher timetables by the use of graph coloring). The unavailability constraints may be such that the initial ordering of vertices cannot be realized on a vertex degree principle any more. Consequently, the CP set may contain so few vertices that a considerable improvement on the backward step is made possible. Again, the Christofides and Brélaz–Peemöller algorithms seem to be well suited for coloring such graphs. However, the best implicit enumeration algorithm for graph coloring is presumably Korman's with Peemöller's updating of current predecessors.

Acknowledgments. The authors wish to thank Eugeniusz Kusz for stimulating discussions on the subject of this article.

REFERENCES

1. Brélaz, D. New methods to color the vertices of a graph. *Commun. ACM 22*, 4 (Apr. 1979), 251–256.
2. Brown, R.J. Chromatic scheduling and the chromatic number problem. *Manage. Sci. 19*, 4 (Dec. 1972), 451–463.
3. Christofides, N. *Graph Theory. An Algorithmic Approach.* Academic Press, London, 1975, pp. 70–71.
4. Dailey, D.P. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Math. 30*, (1980), 289–293.
5. Dürre, K. An algorithm for coloring the vertices of an arbitrary graph. In *Lecture Notes in Economics and Mathematical Systems.* Vol. 78, P. Deussen Ed., Springer-Verlag, Berlin, 1973, pp. 82–89.
6. Garey, M.R., and Johnson, D.S. The complexity of near-optimal graph coloring. *J. ACM 23*, 1 (Jan. 1976), 43–49.
7. Garey, M.R., Johnson, D.S., and Stockmeyer, L. Some simplified NP-complete graph problems. *Theor. Comput. Sci. 1*, (1976), 237–267.
8. Horowitz, E., and Sahni, S. *Fundamentals of Computer Algorithms.* Computer Science, Potomac, Md., 1978, pp. 614–621.
9. Korman, S.M. The graph-colouring problem. In *Combinatorial Optimization.* N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, Eds., Wiley, New York, 1979, pp. 211–235.
10. Kubale, M., and Kusz, E. Computational experience with implicit enumeration algorithms for graph coloring. In *Proceedings of the WG'83 International Workshop on Graphtheoretic Concepts in Computer Science.* M. Nagl and J. Perl, Eds., Trauner Verlag, Linz, 1983, pp. 167–176.
11. Peemöller, J. A correction to Brélaz's modification of Brown's coloring algorithm. *Commun. ACM 26*, 8 (Aug. 1983), 595–597.
12. Schmitz, L. An improved transitive closure algorithm. *Computing 30*, 4 (1983), 359–371.

CR Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*backtracking, graph and tree search strategies*

General Terms: Algorithms

Additional Key Words and Phrases: backtracking, chromatic number, computational complexity, graph coloring, implicit enumeration, partial correctness.

Received 3/84; revised 9/84; accepted 12/84

Authors' Present Addresses: Marek Kubale, Institute of Informatics, Technical University of Gdańsk, Gdańsk, Poland. Bogusław Jackowski, Institute of Hydroengineering, Polish Academy of Sciences, Gdańsk, Poland.