# FINDING THE CHROMATIC NUMBER BY MEANS OF CRITICAL GRAPHS

**Francine HERRMANN**
LITA, Université de Metz
Ile de Saulcy
57045 Metz Cedex France
Email : herrmann@lrim.sciences.univ-metz.fr

*and*

**Alain HERTZ**
Département de Mathématiques et de Génie Industriel
Ecole Polytechnique
CP 6079, succ. Centre-ville,
Montréal (QC) H3C 3A7, Canada
Email : Alain.Hertz@gerad.ca

**Abstract**

We propose a new exact algorithm for finding the chromatic number of a graph *G*. The algorithm attempts to determine the smallest possible induced subgraph *G'* of *G* which has the same chromatic number as *G*. Such a subgraph is said critical since all proper induced subgraphs of *G'* have a chromatic number strictly smaller than *G'*.
The proposed method is particularly helpful when a *k*-coloring of a non-critical graph is known, and it has to be proved that no (*k*-1)- coloring of *G* exists. Computational experiments on random graphs and on DIMACS benchmark problems demonstrate that the new proposed algorithm can solve larger problems than previous known exact methods.

*Key-words*
Chromatic number, exact algorithm, critical graphs.

## 1. INTRODUCTION

Given a graph $G=(V,E)$ with vertex set $V$ and edge set $E$, and given an integer $k$, a *k-coloring* of G is a function $c$: $V \rightarrow \{1,...,k\}$ such that $c(i) \neq c(j)$ for all edges $(i,j)$ in E. The *chromatic number* $\chi(G)$ of $G$ is the smallest integer $k$ such that there is a $k$-coloring of G.

Finding the chromatic number of a graph is an NP-hard problem [Garey 79]. Exact algorithms have been developed by several researchers [Brown 1972, Brélaz 1979, Peemöller 1983, Kubale 1985, Mehotra 1996], but all these exact methods can only be applied on small instances. This certainly explains why many heuristics methods have been developed for getting an upper bound on the chromatic number of a graph. A survey of the most famous heuristic graph coloring methods can be found in [de Werra 1990]. Nowadays, the most powerful heuristic algorithms are adaptations of metaheuristics (e.g., tabu search, genetic algorithms) [Fleurent 1996, Galinier 1998].

All known exact methods for finding the chromatic number of a graph $G$ suffer from the same difficulty : while it is often easy to exhibit a $\chi(G)$-coloring by means of an efficient heuristic method, it is however difficult to prove that there exists no ($\chi(G)$-1)-coloring of $G$. Typically, on random graphs with 85 vertices and edge density 0.5, exact methods only need a few seconds to determine a $\chi(G)$-coloring, while several minutes or hours are needed to prove that no better coloring exists.

A graph $G$ is said *critical* if all its proper induced subgraphs have a chromatic number strictly smaller than $\chi(G)$. All non-critical graphs $G$ contain a proper induced subgraph $G'$ which is critical and such that $\chi(G')=\chi(G)$. In order to compute the chromatic number of a graph $G$, we will try to determine the smallest possible critical subgraph $G'$ of $G$ such that $\chi(G')=\chi(G)$. While $G$ and $G'$ have the same chromatic number, $G'$ has hopefully less vertices than $G$. We will therefore apply any known exact method on $G'$ instead of $G$.

The new exact algorithm is described in details in the next section. Various possible strategies for implementing this algorithm are detailed in Section 3. Computational experiments on random graphs as well as on DIMACS benchmark problems [Johnson 1996] are reported in Section 4. We conclude by final remarks and possible extensions.

## 2. A NEW EXACT ALGORITHM

Let `EXACT` be any exact algorithm for the computation of the chromatic number of a graph $G$, and let `HEURISTIC` be any heuristic method which produces an upper bound, denoted $H(G)$, on the chromatic number of $G$. Our new exact algorithm for the computation of the chromatic number is based on a combination of `EXACT` and `HEURISTIC`, together with the concept of critical graphs.

A description of the new exact algorithm is given in Table 1. We have used the following notations. For a graph $G=(V,E)$ and a vertex $x$ in $V$, we denote $G-x$ the subgraph of $G$ induced by $V\setminus\{x\}$. Similarly, given any induced subgraph $G'=(V',E')$ of $G=(V,E)$, and given any vertex $x$ in $V\setminus V'$, we denote $G'+x$ the subgraph of $G$ induced by $V'\cup\{x\}$.

*Initialisation phase*

1. Determine an upper bound $k=H(G)$ on $\chi(G)$ by means of `HEURISTIC`;

*Descending phase*

2. Set $G'=G$;
   Scan all vertices $x$ of $G$ in some order, and for each such $x$ do the following:
   (a) Determine an upper bound $\mu=H(G'-x)$ on $\chi(G'-x)$ by means of `HEURISTIC`;
   (b) If $\mu=k$ then set $G':=G'-x$;
3. Determine the chromatic number $k'=\chi(G')$ of $G'$ by means of `EXACT`;
   If $k'=k$ then STOP : $k$ is the chromatic number of $G$;

*Augmenting phase*

4. Set `List` to the empty set;
   For each vertex $x$ which is in $G$ but not in $G'$ do the following
   (a) Determine an upper bound $\mu=H(G'+x)$ on $\chi(G'+x)$ by means of `HEURISTIC`;
   (b) If $\mu>k'$ then
   - Determine the chromatic number $k''=\chi(G'+x)$ by means of `EXACT`;
   - If $k''=k'+1$ then introduce $x$ into `List`;
5. If `List` is not empty then choose a vertex $x$ in `List`, set $G'=G'+x$ and $k'=k'+1$;
   Otherwise choose a vertex $x$ which is in $G$ but not in $G'$, and set $G'=G'+x$;
6. If $G'=G$ or $k'=k$ then STOP : $k'$ is the chromatic number of $G$;
   Otherwise go to Step 4.

**Table 1.** *A new exact algorithm for the computation of the chromatic number of a graph G.*

We now give a detailed explanation of this new algorithm as well as a justification of its correctness. In the initialization phase, we compute an upper bound $k$ on the chromatic number of $G$ by means of HEURISTIC. We then start a *descending phase* that aims to determine a critical induced subgraph $G'$ of $G$ with $\chi(G')=\chi(G)$. This is done as follows. We first set $G'$ equal to $G$. Ideally, we should remove vertices from $G'$, as long as such removals do not change the chromatic number of $G'$. However, such a process would require the comparison between $\chi(G')$ and $\chi(G'-x)$, for each vertex $x$ in $G'$. In order to avoid such computations, we use upper bounds on the values of $\chi(G')$ and $\chi(G'-x)$. All these bounds are obtained by means of `HEURISTIC`. In other words, a vertex $x$ is removed from $G'$ only if $H(G')$ is equal to $H(G'-x)$.

**Property 1**
Let $k$ be an upper bound on the chromatic number $\chi(G)$ of a graph $G$, and let $G'$ be any subgraph of $G$. If $\chi(G')=k$, then $\chi(G)=\chi(G')$.

**Proof**
$\chi(G)\geq\chi(G')$ since $G'$ is a subraph of $G$, and $k\geq\chi(G)$ since $k$ is an upper bound on $\chi(G)$. It follows that $\chi(G)=\chi(G')$ when $\chi(G')=k$. ♦

**Property 2**
Let $H(G)$ be the upper bound computed at Step 1 of the above algorithm, and let $G'$ be the graph obtained from $G$ at the end of Step 2. If $\chi(G')=H(G)$, then $G'$ is critical and has the same chromatic number as $G$.

**Proof**
Assume that $\chi(G')=H(G)$. We know from Property 1 that $\chi(G)=\chi(G')$. Moreover, $G'$ is obtained from $G$ by removing all vertices that do not modify the value of the upper bound. We therefore have $H(G')=H(G)$ and $H(G'-x)<H(G')$ for all vertices $x$ remaining in $G'$. Hence, $\chi(G'-x)\leq H(G'-x)<H(G')=H(G)=\chi(G)$ for all $x$ in $G'$, which means that $G'$ is critical. ♦

In Step 3 of the *descending phase* we compute the chromatic number $\chi(G')$ of $G'$. It follows from Property 1 that if $\chi(G')$ is equal to the upper bound $H(G)$ on $\chi(G)$, then $\chi(G)=H(G)$. The algorithm can therefore be stopped.

**Property 3**
If all upper bounds produced by `HEURISTIC` in Steps 1 and 2 of the above algorithm are equal to the chromatic number of the graphs given as input, then the algorithm stops at Step 3.

**Proof**
Assume that the upper bound $H(G)$ computed at Step 1 is equal to $\chi(G)$, and that each time a vertex $x$ is removed from $G'$ at Step 2 we have $\chi(G'-x)=H(G'-x)$. Since a vertex $x$ is removed from $G'$ only if $H(G'-x)=H(G)$, we have $\chi(G')=H(G)=\chi(G)$ when entering Step 3, which means that the algorithm stops. ♦

Notice that if the algorithm stops at Step 3, this means that only one call to `EXACT` is required, and this call is performed on a graph $G'$ which has hopefully less vertices than the original graph $G$. It may however happen that $\chi(G')<H(G)$. In such a case, we start the *augmenting phase* in which vertices are added to $G'$ until either $\chi(G')=H(G)$ or $G'=G$. We now give details on this *augmenting phase*.

Given a subgraph $G'$ of $G$, we consider each vertex $x$ which is in $G$ but not in $G'$, and we compute an upper bound $H(G'+x)$ on $\chi(G'+x)$. If this upper bound is strictly larger than $\chi(G')$, we have good reasons to suspect that $\chi(G'+x)=\chi(G')+1$. We therefore apply `EXACT` on $G'+x$ in order to

confirm this fact. If $\chi(G'+x)$ is effectively equal to $\chi(G')+1$, we put $x$ into a set called `List`. At the end of Step 4, each vertex $x$ in `List` is such that $\chi(G'+x)=\chi(G')+1$, while each other vertex $x$ which is in $G$ but not in $G'$ is such that $\chi(G'+x)=\chi(G')$.

At Step 5, we add a vertex $x$ to $G'$. If `List` is not empty, then $x$ is chosen in it, and the chromatic number of $G'$ is increased by one unit. Otherwise, any vertex in $G$ but not in $G'$ is added to $G'$ without increasing the chromatic number of $G'$.

If during the *augmenting phase* we can exhibit a graph $G'$ such that $\chi(G')=H(G)$, then it follows from Property 1 that $\chi(G)=H(G)$, and the algorithm can therefore be stopped. Otherwise, vertices are iteratively added to $G'$ until $G'=G$, in which case $\chi(G)$ is equal to $\chi(G')$ (which has already been computed in the previous steps).

Notice that when a vertex $x$ from `list` is added to $G'$ in Step 5, it may happen that the augmented graph contains vertices which can be removed from it without modifying the chromatic number. A *desending phase* could therefore also be applied to the augmented graph. This can easily be done by putting Step 5' described in Table 2 in place of Step 5 of the original algorithm

5'. If `List` is not empty then

(a) choose a vertex $x$ in `List`, set $G'=G'+x$ and $k'=k'+1$;

*Descending phase*

(b) Scan all vertices $x$ of $G'$ in some order, and for each such $x$ do the following
- Determine an upper bound $\mu=H(G'-x)$ on $\chi(G'-x)$ by means of `HEURISTIC`;
- If $\mu=k'$ then
  - Determine the chromatic number $k''=\chi(G'-x)$ of $G'-x$ by means of `EXACT`;
  - If $k''=k'$ then set $G'=G'-x$;

Otherwise choose a vertex $x$ which is in $G$ but not in $G'$, and set $G'=G'+x$;

**Table 2.** *A possible variation on Step 5.*

Notice that the descending phase in the above Step 5' is slightly different from the descending phase of Step 2. Indeed, when `HEURISTIC` indicates that a vertex $x$ can probably be removed from $G'$ without modifying its chromatic number, we check this fact in Step 5' but not in Step 2. The reason for this difference is that we need to update the chromatic number of $G'$ each time $G'$ is modified in the *augmenting phase.*

In Tables 1 and 2, we have underlined some flexible points of the algorithm where different strategies can be implemented. We describe in the next section the choices we have made for these strategies.

## 3. IMPLEMENTATION STRATEGIES

The subgraph *G'* obtained from *G* at the end of Step 2 may depend on the ordering with which the vertices are scanned. We have chosen to order the vertices dynamically by choosing at each step the vertex in *G'* with smallest degree. We have also tested other strategies (e.g., random ordering, ordering according to non-decreasing degree in *G*, etc.), but these other orderings systematically produced worse results when compared to our choice. The chosen strategy is illustrated for the graph in Figure 1. We will assume in what follows that given any graph *G*, the upper bound *H(G)* delivered by `HEURISTIC` is equal to the chromatic number $\chi(G)$ of *G*.

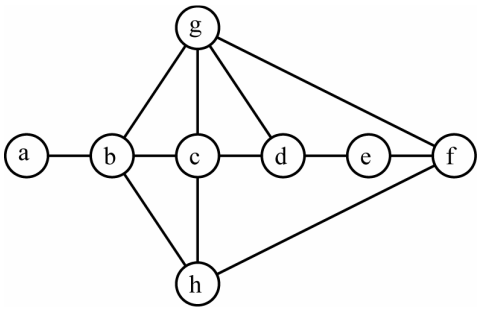


**Figure 1.** *A graph which helps to illustrate the strategy chosen for Step 2.*

The proposed algorithm first sets *H(G)* equal to 3 in Step 1. Then, when applying our strategy for Step 2, we first try to remove vertex *a* since it has the smallest degree in *G=G'*. Since such a removal does not change the chromatic number of *G'*, it is performed. Vertex *e* is now the vertex with smallest degree in *G'*, and it is also removed from *G'* since the remaining graph still has chromatic number 3. At this stage, both vertices *d* and *f* have degree 2 in *G'* and we choose one at random, say *f*. The removal of *f* induces a new graph *G'* that still has chromatic number 3. In the next step, either *d* or *h* is removed, say *d*, and finally either *g* or *h* is removed, say *g*. The resulting graph *G'* is a triangle made of *b*, *c* and *h*. No additional vertex can be removed from *G'* without modifying its chromatic number. We therefore enter Step 3 with this triangle.

For comparison, consider a similar strategy for Step 2, except that we choose a vertex with largest (instead of smallest) degree in *G'*. In such a case, there are three candidate vertices to be removed at the first iteration of Step 2. Indeed, vertices *b*, *c* and *g* all have degree 4, which is the largest one in *G=G'*. Assume that vertex *c* is chosen. `HEURISTIC` will find a 2-coloring *of G'-c*. Vertex *c* is therefore not removed from *G'*. If the next candidate is vertex *g*, then it is removed

from *G'* since $\chi(G'-g)$=3. There are then three vertices with largest degree 3, vertices *b*, *c* and *h*. Vertex *c* has already been considered in the previous iteration and is therefore not a candidate to be removed from *G'*. If vertex *h* is chosen as next candidate, it is not removed from *G'* since $\chi(G'-h)$=2. Since the removal of vertex *b* does not modify the chromatic number of *G'*, it is performed and the resulting graph is a pentagon on vertices *c*, *d*, *e*, *f*, and *h* plus an isolated vertex *a*. Vertices *c* and *h* have already been considered before and are therefore not candidates to be removed. Vertices *d*, *e* and *f* cannot be removed from *G'* since they induce a decrease of the chromatic number. The last vertex which is removed is therefore vertex *a*, and we enter Step 3 with *G'* equal to a pentagon.

During the *augmenting phase*, there are several possible strategies for adding vertices to *G'*. If `List` is not empty we have decided to add to *G'* the vertex in `List` which has the largest number of neighbors in *G'*. If `List` is empty, then we choose the vertex which is in *G* but not in *G'*, and which has the largest number of neighbors in *G'*. Here also, we have tested different strategies, but the chosen one has systematically produced the best results.

## 4. COMPUTATIONAL EXPERIMENTS

There are many possible choices for `EXACT` and `HEURISTIC`. For `EXACT`, we have chosen to implement the branch-and-bound algorithm described in [Peemöller 1983] which is a corrected version of Brélaz's modification of Brown's algorithm [Brown 1972, Brélaz 1979]. For `HEURISTIC`, it is very important to make a good choice. Indeed, if the initial upper bound $H(G)$ on $\chi(G)$ produced by `HEURISTIC` is strictly larger than $\chi(G)$, then the algorithm will stop at Step 6 with *G'*=*G*. In such a case, there is no benefit to use the new algorithm instead of `EXACT`. It is therefore important to choose an efficient heuristic method for `HEURISTIC`. We have implemented the Tabu search method described in [Hertz 1987]. While better heuristic methods are available [e.g., Fleurent 1996, Galinier 1998], the chosen one always produced upper bounds equal to the chromatic number of the graphs given as input. It follows from Property 3 that our algorithm never entered the *augmenting phase.*

Since `HEURISTIC` is much faster than `EXACT` (their CPU-time differ by several orders of magnitude) the total computational time of the new algorithm is mainly due to Step 3 where the chromatic number of the critical graph *G*' is determined. Hence, in order to evaluate the benefit of using the new proposed algorithm instead of `EXACT`, we report for each graph instance $G=(V,E)$ the following information which does not depend on the computer used :

*size of the critical subgraph* : number of vertices in the critical subgraph *G*' of *G*.

*backtracks* `EXACT` : number of backtracks required by `EXACT` when computing the chromatic number of *G*

*backtracks New* : number of backtracks required by the new algorithm when computing the chromatic number of the critical subgraph *G'* of *G*.

We have performed tests on random graphs and on DIMACS benchmark problems [Johnson 1996]. Random graphs $G_{n,p}$ are graphs with $n$ vertices such that there exists an edge with a probability $p$ (called *edge density*) between any pair of vertices, independently of the existence or non-existence of any other edge. This family of graphs has been deeply studied with respect to coloring, especially for $p$=0.5. We have tested our new algorithm on random graphs having $n$=40, 50, 60, 70, 80, 85, 90, 95 and 100 vertices, and edge density 0.5. We have randomly generated 10 graphs $G_{n,p}$ for each couple $(n,p)$ and give average results in Table 3. All these instances can be obtained via the Web site *http://dmawww.epfl.ch/rose.mosaic/ah/RANDOM_GRAPHS*. We have imposed for each instance a time limit of 24 hours. When an algorithm is not able to solve all 10 instances of a same size, we only indicate, in parenthesis, the number of instances for which the chromatic number could be determined.

| *n* | $\chi(G_{n,0.5})$ | *size of the critical subgraph* | *backtracks New* | *backtracks* `EXACT` |
|---|---|---|---|---|
| 40 | 8.3 | 22.9 | $36.3\ 10^0$ | $97.1\ 10^0$ |
| 50 | 9.4 | 33.8 | $1.7\ 10^3$ | $2.0\ 10^3$ |
| 60 | 10.6 | 45.7 | $42.8\ 10^3$ | $24.4\ 10^3$ |
| 70 | 11.7 | 58.4 | $331.2\ 10^3$ | $551.1\ 10^3$ |
| 80 | 12.8 | 68.9 | $11.8\ 10^6$ | $13.1\ 10^6$ |
| 85 | 13.0 | 69.0 | $10.7\ 10^6$ | $26.0\ 10^6$ |
| 90 | 13.8 | 79.7 | $1632.7\ 10^6$ | (8) |
| 95 | 14.0 | 78.5 | $393.4\ 10^6$ | (2) |
| 100 | - | - | (2) | (0) |

**Table 3.** *Results for random graphs* $G_{n,0.5}$

This table requires some comments. First of all, it clearly appears that the new algorithm can solve larger problems than `Exact`. For example, while our method is able to find the chromatic number of all instances with 95 vertices, only 2 such graphs could be solved by `EXACT` within one day CPU-time.

Surprisingly, `EXACT` is faster than the new algorithm for random graphs having 60 vertices. We could expect the opposite since our algorithm has to determine the chromatic number of graphs having 45.7 vertices in average, and all these graphs are subgraphs of a $G_{60,0.5}$. The example depicted in Figure 2 shows that given a graph $G$ and a critical subgraph $G'$ with $\chi(G')=\chi(G)$, it may happen that $\chi(G)$ is easier to compute than $\chi(G')$. In this example, the triangle made of vertices $d$, $e$ and $h$ in $G$ is easily detected by `Exact` and it provides a proof that there is no 2-coloring of $G$. Such a proof is not so simple for the critical subgraph $G'$ of $G$.

Notice also that while the computational effort needed by `EXACT` increases with problem size, this is not always the case for the new algorithm which is, in average, slightly faster for $G_{85,0.5}$ than for $G_{80,0.5}$, and four times faster for $G_{95,0.5}$ than for $G_{90,0.5}$. This surprising behavior can be

explained as follows. Almost all tested random graphs $G_{80,0.5}$ (8 out of 10) have the same chromatic number as the tested $G_{85,0.5}$. The *descending phase* has therefore more flexibility in $G_{85,0.5}$ than in $G_{80,0.5}$ for choosing which vertices should be removed in order to get a critical subgraph *G'*. The same situation occurs with the tested $G_{90,0.5}$ which almost all have the same chromatic number as the tested $G_{95,0.5}$. To illustrate this phenomenon, consider once again the two graphs in Figure 2. Both have their chromatic number equal to 3. The smaller one *G'* is critical and can therefore not be reduced. There is however some flexibility for the removal of vertices in the larger graph *G*. One can for example remove vertex *h* and get the smaller graph *G'*. Another possibility is to remove vertices *a, b, c, f* and *g* and get a triangle which is easier to color than *G'*. This may explain why the critical subgraphs obtained from a $G_{95,0.5}$ are in average smaller than those obtained from a $G_{90,0.5}$.

The same phenomenon could be observed with random graphs having 100 vertices. Indeed, we could prove that two of the ten tested $G_{100,0.5}$ have a chromatic equal to 14 (which is the chromatic number of all tested $G_{95,0.5}$) while the chromatic number of the eight other instances seems to be equal to 15. The two solved instances have first been reduced to subgraphs having 77.5 vertices in average (to be compared with 78.5), and we then needed only $280\ 10^6$ backtracks (to be compared with $393\ 10^6$) to find their chromatic number. For comparison, the eight other 100 vertex instances have been reduced to sugraphs having 90.9 vertices.

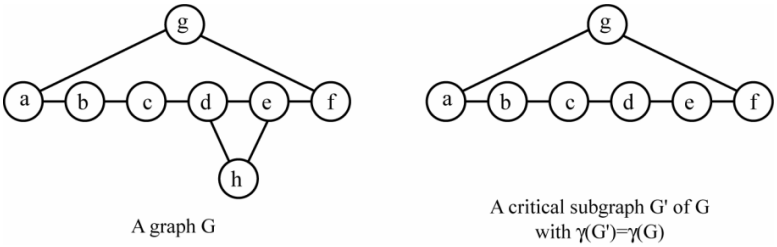


**Figure 2.** *A graph and one of its critical subgraphs.*

We have also tested our new algorithm on some DIMACS benchmark problems [Johnson 1996]. These problems can be divided into 5 categories (more details can be obtained on the Web site *http://mat.gsia.cmu.edu/COLOR/instances.html*).

*Book graphs*. Given a work of literature, a graph is created where each vertex represents a character, and two vertices are connected by an edge if the corresponding characters encounter each other in the book. We have considered the graphs *anna* (from Tolstoy's Anna Karenina), *david* (from Dicken's David Copperfield), *huck* (from Twain's Huckleberry Finn) and *jean* (from Hugo's Les Misérables).

*Game graphs*. These graphs represent the games played in a college football season. The vertices represent each college team, and two teams are connected by an edge if they played each other during the season. We have considered the graph *games120* which

has been obtained from the 1990 college football season.

*Miles graphs.* The vertices represent a set of United States cities. Two cities are linked by an edge if they are close enough. We have considered the graphs *miles250* and miles 500 which differ on the threshold value used to create edges.

*Myc graphs*. These graphs are based on Mycielski's transformation. They are difficult to solve because they are triangle free but the chromatic number increases in problem size. We have tested the graphs *myciel3*, *myciel4*, *myciel5* and *myciel6*.

*Queen graphs*. Given an *n* by *m* chessboard, a queen graph *queen.n_m* is a graph with $n \cdot m$ vertices, each corresponding to a square of the board. Two vertices are connected by an edge when the corresponding squares are in the same row, column or diagonal. We have considered the graphs *queen.5_5*, *queen.6_6*, *queen.7_7*, *queen.8_8*, *queen.9_9* and *queen.8_12*.

*DSJC graphs*. These are benchmark random graphs used in [Johnson 1991]. We have considered *DSJC125.1* which has 125 vertices and an edge density 0.1.

The results are reported in Table 4. It can be observed that *Myc graphs* are critical and there is therefore no advantage of using our algorithm instead of `Exact` for this category of graphs. Some other graphs are however much easier to color using our new method. The best examples are provided by *games120* and *queen.8_12* for which `Exact` has not been able to determine the chromatic number in less than 24 hours while our new algorithm has reduced these graphs to a clique and has then needed only one backtrack to determine their chromatic number.

| ***name of the graph*** | ***number of vertices*** | ***number of edges*** | ***chromatic number*** | ***size of the critical subgraph*** | ***backtracks New*** | ***backtracks EXACT*** |
|---|---|---|---|---|---|---|
| anna | 138 | 493 | 11 | 11 | 1 | 8 |
| david | 87 | 406 | 11 | 11 | 1 | 1 |
| huck | 74 | 301 | 11 | 11 | 1 | 211 $10^3$ |
| jean | 80 | 254 | 10 | 10 | 1 | 8 $10^3$ |
| games120 | 120 | 638 | 9 | 9 | 1 | - |
| miles250 | 128 | 387 | 8 | 8 | 1 | 1 |
| miles500 | 128 | 1170 | 20 | 20 | 1 | 8 |
| myciel3 | 11 | 20 | 4 | 11 | 4 | 4 |
| myciel4 | 23 | 71 | 5 | 23 | 106 | 106 |
| myciel5 | 47 | 236 | 6 | 47 | 28 $10^3$ | 28 $10^3$ |
| myciel6 | 95 | 755 | 7 | 95 | 416 $10^6$ | 416 $10^6$ |
| queen.5_5 | 25 | 160 | 5 | 5 | 1 | 1 |
| queen.6_6 | 36 | 290 | 7 | 26 | 43 | 410 |
| queen.7_7 | 49 | 476 | 7 | 7 | 1 | 2 $10^3$ |
| queen.8_8 | 64 | 728 | 9 | 54 | 246 $10^3$ | 597 $10^3$ |
| queen.9_9 | 81 | 2112 | 10 | 72 | 170 $10^6$ | 81 $10^6$ |
| queen.8_12 | 96 | 1368 | 12 | 12 | 1 | - |
| DSJC125.1 | 125 | 1472 | 5 | 11 | 3 | 227 |

**Table 4.** *Results for some DIMACS benchmark problems*

## 5. FINAL REMARKS AND CONCLUSION

We have developed a new exact algorithm for finding the chromatic number of a graph $G$. The proposed algorithm first determines an upper bound $k$ on $\chi(G)$ by means of a heuristic method, and then tries to find a critical subgraph $G'$ such that $\chi(G')=k$. We have seen that if the initial upper bound $k$ is strictly larger than $\chi(G)$, then there is no benefit to use the new proposed algorithm. It is therefore important to choose an efficient heuristic method for the computation of the initial upper bound $k$. Also, when the given graph $G$ is critical, the *descending phase* is not able to remove any vertex from $G$, and the proposed algorithm therefore enters Step 3 with the original graph. In all other cases, the critical subgraph $G'$ determined by our method is smaller than the original graph $G$, and hopefully easier to color.

We have performed computational experiments on random graphs and on DIMACS benchmark problems. These experiments clearly demonstrate that our method can solve larger problems than previous known exact methods.

It is to be mentioned that the new algorithm can be implemented in various ways. First of all, any known exact coloring algorithm can be used for `EXACT`, and any known heuristic coloring method can be used for `HEURISTIC`. Moreover, the order in which the vertices are removed from $G$ during the *descending phase* may influence the size of the resulting critical subgraph $G'$ of $G$. We have decided to order the vertices dynamically by choosing at each step the vertex with smallest degree in the current subgraph $G'$. Other strategies may be preferred and can possibly generate smaller critical subgraphs.

**References**


[Brélaz 1979] Brélaz D., "New Methods to Color the Vertices of a Graph", *Communications of the ACM* 22/4, 1979, 251-256.

[Brown 1972] Brown J.R., "Chromatic Scheduling and the Chromatic Number Problem", *Management Science* 19/4, 1972, 456-463.

[Fleurent 1996] Fleurent C. and Ferland J.A., " Genetic and Hybrid Algorithms for Graph Coloring", *Annals of Operations Research* 63, 1996, 437-461.

[Galinier, 1998] Galinier P. and Hao J.K., "Hybrid evolutionary algorithms for graph coloring". A paraître dans *Journal of Combinatorial Optimization*.

[Garey 1979] Garey M.R. and Johnson D.S., *Computers and Intractabilitiy : A Guide to the Theory of NP-Completeness,* W.H. Freman and Company, New York, 1979

[Hertz 1987] Hertz A. and de Werra D., "Using Tabu Search for Graph Coloring", *Computing* 39, 1987, 345-351.

[Johnson 1991] Johnson D.S., Aragon C.R., McGeoch L.A. and Schevon C., "Optimization by Simulated Annealing: an Experimental Evaluation. Part II, Graph Coloring and Number Partitioning", *Operations Research* 39, 1991, 378-406.

[Johnson 1996] Johnson D.S. and Trick M.A., "Proceedings of the 2nd DIMACS Implementation Challenge", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 26, American Mathematical Society, 1996.

[Peemöller 1983] Peemöller J., "A Correction to Brélaz's Modification of Brown's Coloring Algorithm", *Communications of the ACM* 26/8, 1983, 593-597.

[Kubale 1985] Kubale M. and Jackowski B., "A Generalized Implicit Enumeration Algorithm for Graph Coloring", *Communications of the ACM* 28/4, 1985, 412-418.

[Mehotra 1996] Mehrotra A. and Trick M.A., "A Column Generation Approach for Exact Graph Coloring", *INFORMS Journal on Computing* 8, 1996, 344-354.

[de Werra 1990] de Werra D., "Heuristics for Graph Coloring", *Computing* 7, 1990, 191-208.