

Classical Coloring of Graphs

ADRIAN KOSOWSKI, KRZYSZTOF MANUSZEWSKI

Despite the variety of graph coloring models discussed in published papers of a theoretical nature, the classical model remains one of the most significant and widely applied in practice. The NP-hardness of the coloring problem gives rise to the necessity of using suboptimal methods in a wide range of practical applications. Moreover, the large range of problems solved by classical coloring, as well as the variety of graph families with practical significance in this field aids the evolution and development of new suboptimal algorithms. There exist several relatively simple methods, which are regarded as classical due to their date of creation or scope of practical application. As the implementation of a particular algorithmic solution requires the selection of at least one coloring method, it is essential to formulate criteria for the assessment of the suitability of coloring algorithms. Speed of operation, measured through computational complexity, is obviously one of the most important features which are taken into consideration when selecting a graph coloring approach. For suboptimal methods, the algorithm's performance guarantee is another characteristic feature, describing how accurate, or more precisely how inaccurate the obtained results may be. The analysis of the smallest hard to color graphs is yet another criterium, which in a certain sense complements the performance guarantee.

1.1. Basic terms and definitions

DEFINITION 1.1. A *graph* G is an ordered pair $G = (V, E)$, where V stands for a finite set of elements called *vertices*, while E – a finite set of unordered pairs of vertices called *edges*. The cardinality of the set of vertices V is denoted by the symbol $n = |V|$ and called the *order* of graph G . Likewise, the cardinality of the set of edges E is denoted by $m = |E|$ and called the *size* of graph G . The vertices $u, v \in V$ are called *adjacent* (or *neighbours*) if $\{u, v\} \in E$ and *nonadjacent* if $\{u, v\} \notin E$. The edges $e, f \in E$ are said to be *adjacent* if $e \cap f \neq \emptyset$ and *nonadjacent* if $e \cap f = \emptyset$.

DEFINITION 1.2. The *degree* $\deg(v)$ of *vertex* v in graph G is the number of edges incident to vertex v in graph G , that is $|\{e \in E: v \in e\}|$. The maximum degree of a vertex in graph G is denoted by $\Delta(G)$, while the minimum degree is denoted by $\delta(G)$. The number $g(G) = 2m/(n(n-1))$ is known as the *density* of graph G .

Only *simple graphs* (graphs with no directed edges, loops or multiple edges) will be taken into account in further considerations.

DEFINITION 1.3. A *path connecting vertices* v_1 and v_k in graph G is an ordered sequence of vertices v_1, v_2, \dots, v_k , in which every vertex appears at most once and for all values of i the following condition is fulfilled: $\{v_i, v_{i+1}\} \in E$.

DEFINITION 1.4. A graph G with at least 2 vertices is called *connected* if every pair of its vertices is connected by a path. We assume, that a 1-vertex graph is connected. The *distance between vertices* u and v in graph G , denoted by $d(u, v)$, is the length of the shortest path connecting vertices u and v in this graph. We will assume, that $d(u, u) = 0$.

DEFINITION 1.5. A *clique* V' in graph $G = (V, E)$ is a subset of V , for which the following condition is fulfilled: $u, v \in V' \implies \{u, v\} \in E$. The term clique is often used to describe not only the set V' , but also the subgraph of G induced by this set of vertices. The clique V' in graph G is called *maximal* if there does not exist any other clique V'' , such that $V' \subset V''$ (Fig. 1). By the *clique number* $\omega(G)$ we understand the size of the largest maximal clique in graph G .

DEFINITION 1.6. An *independent set* in graph G is any subset $V' \subseteq V$, such that $u, v \in V' \implies \{u, v\} \notin E$. The independent set V' in graph G is called *maximal* if it isn't a subset of any other independent set in G (Fig. 1). The *number of independence* $\alpha(G)$ is the size of the largest independent set in G .

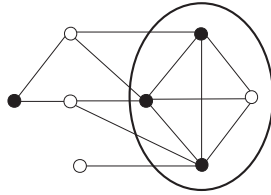


FIGURE 1. An example of a maximal clique (within black circle) and a maximal independent set (white vertices) in a graph

DEFINITION 1.7. A graph G is called *k-partite* if the set of all its vertices can be partitioned into k subsets V_1, V_2, \dots, V_k , in such a way that any edge of graph G connects vertices from different subsets. The terms *bipartite* graph and *tripartite* graph are used to describe k -partite graphs for k equal to 2 and 3, respectively (Fig. 2). A k -partite graph is called *complete* if any vertex $v \in V$ is adjacent to all vertices not belonging to the same partition as v . The symbol K_{n_1, n_2, \dots, n_k} is used to describe a complete k -partite graph, with partition sizes equal to $|V_i| = n_i$ for $i = 1, 2, \dots, k$. Moreover if $n_i = 1$ for all values of i , then the complete k -partite graph is denoted as K_k .

DEFINITION 1.8. The *core* of graph G is the subgraph of G obtained by the iterated removal of all vertices of degree 1 from G (Fig. 3).

DEFINITION 1.9. The *join* $G_1 + G_2$ of the pair of vertex disjoint graphs G_1 and G_2 is a graph containing all the vertices and edges from G_1 and G_2 , as well as all possible edges connecting a vertex from G_1 with a vertex from G_2 .

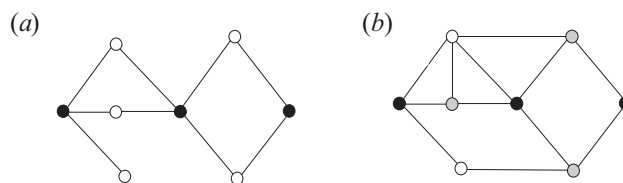


FIGURE 2. Examples of graphs: (a) bipartite, (b) tripartite

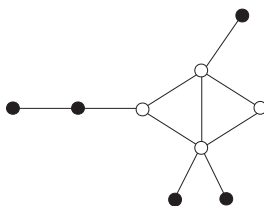


FIGURE 3. An example of a core of a graph (white vertices).

DEFINITION 1.10. A legal *vertex-coloring* of graph $G = (V, E)$ is a function $c: V \rightarrow \mathbb{N}$, in which any two incident vertices $u, v \in V$ are assigned different colors, that is $\{u, v\} \in E \implies c(u) \neq c(v)$. The function c is called the coloring function. A graph G for which there exists a vertex-coloring which requires k colors is called *k-colorable*, while such a coloring is called a *k-coloring*. In this case, the coloring function induces a partition of graph G into independent subsets V_1, V_2, \dots, V_k , for which $V_i \cap V_j = \emptyset$ and $V_1 \cup V_2 \cup \dots \cup V_k = V$.

DEFINITION 1.11. The smallest number k for which there exists a k -coloring of graph G is called the *chromatic number* of graph G and is denoted by $\chi(G)$. Such a graph G is called *k-chromatic*, while any coloring of G which requires $k = \chi(G)$ colors is called *chromatic* or *optimal*.

In a partially colored graph G the term *saturation degree* $\rho(v)$ of vertex v in graph G is defined as the number of distinctly colored vertices adjacent to v .

1.1.1. Graph families. References to certain sets of graphs with a common structure or interesting features appear frequently in further parts of the book. This section contains assembled definitions of some of the more interesting or exotic graph families.

DEFINITION 1.12. An *r-regular* graph is a graph in which all vertices have a degree equal to r . A *cubic graph* is an equivalent term for a 3-regular graph. A hypercube Q_r is an r -regular graph with 2^r vertices (corresponding to all binary sequences n bits long) and $r2^{r-1}$ edges connecting only those vertices, whose binary sequences differ at exactly one position.

DEFINITION 1.13. The *cycle* C_n is a 2-regular connected graph of order n . The *wheel* W_n is the join $C_{n-1} + K_1$. The *path* P_n is the graph C_n without one edge. The *star* S_n is a complete bipartite graph, in which one of the partitions V_1, V_2 is a single vertex (Fig. 4). A *forest* is a graph, which does not contain a cycle. A *tree* is a connected forest.

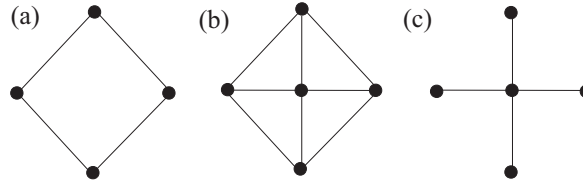


FIGURE 4. Examples of: (a) the cycle C_4 , (b) the wheel W_5 , (c) the star S_5 , the star is an example of a tree

DEFINITION 1.14. A *cactus* is a graph in which any two cycles have at most one vertex in common. A *polygon tree* is any member of a family of graphs defined recursively by the following procedure. Any cycle is a polygon tree. A new polygon tree G' can be created out of an existing polygon tree G by adding a cycle which shares exactly one edge with graph G (Fig. 4).

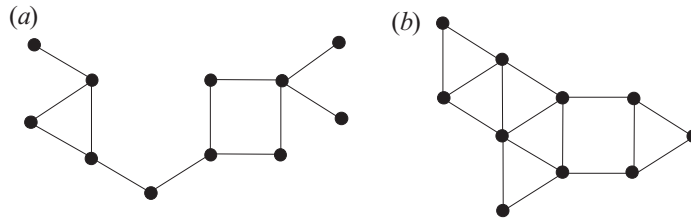


FIGURE 5. Examples of: (a) a cactus; (b) a polygon tree.

1.1.2. Analysis of heuristic methods. The development of heuristic graph coloring methods is necessary due to the computational complexity of optimal algorithms. Graph coloring is an NP-hard problem in the case of most non-trivial coloring models; in particular there are no known optimal polynomial-time solution. However, the large number of existing suboptimal algorithms necessitates the usage of tools and methods which enable the examination and assessment of performance of such heuristics.

Let us take into consideration a graph coloring algorithm A and let $A(G)$ stand for the number of colors used by A to color graph G .

DEFINITION 1.15. For algorithm A its *performance guarantee* $A(n)$ is defined by the following formula:

$$A(n) = \max\{A(G)/\chi(G) : G \text{ is a graph of order } n\}.$$

DEFINITION 1.16. The coloring algorithm A is called a *k-relative approximation algorithm* or simply a *k-approximation algorithm* if $A(G) \leq k \cdot \chi(G)$. The coloring algorithm A is called a *k-absolute approximation algorithm* if $|A(G) - \chi(G)| \leq k$.

The analysis of coloring methods may take either a quantity or quality oriented form. It may concern computational complexity and the closely connected algorithm run time, or the quality of generated solutions. The analysis of the quality of graph coloring methods is usually conducted for graphs of order tending to infinity.

Under such conditions the above defined performance guarantees may be regarded as a particularly appropriate tool for the analysis of algorithm effectiveness.

The performance guarantee enables the assessment of the behavior of an algorithm for the worst-case input data of a given size. Such analysis is usually of an asymptotic nature and can be regarded as an indicator of the outcome of graph coloring for graphs of order $n \rightarrow \infty$. Unfortunately, in the case of some of the more complex methods analysis of algorithm performance may turn out extremely difficult. What is more, even a known performance guarantee does not describe the behavior of the algorithm in the average case. Neither does the performance guarantee correspond to the behavior of the coloring method for relatively small graphs, which may undergo theoretical analysis. Consequently it may not be used to determine all the classes of graphs colored in a suboptimal way by a given method.

When judging the performance of heuristic methods, the relevant characteristics of a method include not only computational complexity and accuracy of generated solutions for large values of n , but also the level of complexity of graphs for which the method leads to suboptimal solutions. In the early 1990's Hansen and Kuplinsky [150] introduced the term of the smallest hard-to-color graph for a method.

DEFINITION 1.17. Graph G is called *slightly hard-to-color* for algorithm A if there exists at least one implementation of algorithm A which colors graph G in a suboptimal way.

DEFINITION 1.18. For a given algorithm A the notation $SHC(A)$ stands for the set of all slightly hard-to-color graphs for algorithm A which have the smallest possible size among all graphs with the smallest possible order. If G is the only element of set $SHC(A)$, we will use the notation $SHC(A) = G$ and refer to graph G as SHC for algorithm A .

DEFINITION 1.19. Graph G is called *hard-to-color* for algorithm A if any implementation of algorithm A leads to a suboptimal coloring of G .

DEFINITION 1.20. For a given algorithm A the notation $HC(A)$ is used to describe the set of all hard-to-color graphs for algorithm A , which have the smallest possible size among all graphs with the smallest possible order. If G is the only element of set $HC(A)$, we will use the notation $SHC(A) = G$ and refer to graph G as HC for algorithm A .

The search for hard-to-color graphs is usually conducted simultaneously in manifold ways. Frequently, theoretical analysis of the properties and mechanism of a particular heuristic proves to be helpful. The first hard-to-color graphs were found in the 1990's. Unfortunately, the development and refinement of coloring methods and, most of all, their rising level of complexity make analysis more and more arduous. Therefore the usage of computer techniques to find the weak point of coloring algorithms appears to be a natural solution. An exhaustive search of the space of all colorings generated by a particular method gives absolute certainty of detecting any number of hard-to-color graphs for the considered method. Unfortunately, the duration time of such an operation may turn out a serious obstacle. One has to realize that even for a relatively small number of vertices n the number of possible graphs spanned on these vertices is very nearly astronomical, as it is expressed by a super-exponential function of n . In addition, for a single graph there may exist numerous

and essentially different legal implementations of a given method, yielding different colorings, which drastically increases the size of the search space. Hard-to-color graphs discovered by means of computer methods later undergo intensive theoretical research, aiming at the verification of the correctness of the obtained results. The theoretical search and analysis of hard-to-color graphs enable the estimation of the anticipated effectiveness of the method and — what is most important — often suggest improvements which contribute to the development of new, more effective graph coloring methods. Furthermore, they enable the comparison of graph coloring algorithms, as more effective algorithms as a rule have larger hard-to-color graphs than less optimal algorithms.

Many new algorithms for heuristic graph coloring use different approaches, often determined by the specific features of the problem, which leads to serious difficulties in testing the effectiveness of such algorithms. On the whole, there are two general, sensible approaches to testing algorithms, and they do not rule each other out. The first approach relies on the choice of such families of hard-to-color graphs which are connected with dedicated coloring methods used for certain classes of problems, or such families of hard-to-color graphs which are connected with characteristic input data sets. The second approach is based on a separate search and analysis of hard-to-color graphs for entire families of coloring methods. Such graphs are called benchmarks. Likewise, we define weak benchmarks as slightly hard-to-color graphs for many algorithms.

DEFINITION 1.21. Let $A = \{A_1, \dots, A_k\}$ be a family of graph coloring algorithms. Graph G is called a *benchmark* for family A if G is HC for all A_i , $i = 1, \dots, k$. Graph G is called a *weak benchmark* for family A if G is SHC for all A_i , $i = 1, \dots, k$.

It is interesting to observe that for certain families of algorithms benchmarks do not exist because there are no hard-to-color graphs for these families (slightly hard-to-color graphs, however, always exist).

The aforementioned approaches to the analysis of graph coloring methods also provide other supplementary information. By testing a heuristic method on general graphs or on a class of large graphs which are difficult for other methods, we can obtain an estimate of the method's behavior for such graphs. It is possible to measure the percentage of graphs from such a class which are colored suboptimally, and judge how inaccurate the generated colorings may turn out.

A survey of known benchmarks, smallest hard-to-color and slightly hard-to-color graphs for various coloring models and heuristic algorithms can be found in [222, 261].

1.2. Classical vertex-coloring

1.2.1. Problem complexity and simplest bounds. Classical graph coloring or, more precisely, the task of finding an optimal vertex-coloring as defined in the classical sense, is an NP-hard problem [97], that is — informally speaking — a problem with no known polynomial solution. The fact, that even the task of estimating the value of the graph classical chromatic number by means of any k -relative or k -absolute approximation algorithm is NP-hard, may be regarded as some measure of the true difficulty of the analyzed problem. Moreover, classical coloring remains NP-hard even in spite of strong restrictions imposed on the class

of graphs which are taken into consideration and on the number of colors used. Quite recently a proof of the NP-hardness of 4-coloring of tripartite graphs was presented [199]. The problem of determining whether a given planar graph with a degree not exceeding 4 is 3-colorable, is also NP-hard [98]. On the other hand, there exist entire classes of graphs whose chromatic number can be given through a simple formula, or whose optimal coloring can be determined in polynomial time (interval graphs, for instance, fall into this category [286]).

In order to determine a simple bound on the graph chromatic number in the general case, it is sufficient to notice that the chromatic number of an empty graph is equal to 1, while the chromatic number of a complete graph of order n equals n . A trivial bound for the chromatic number of any graph G comes as an immediate conclusion:

$$1 \leq \chi(G) \leq n.$$

An improvement of the lower bound may be reached through the observation that every clique of size k has to be colored with exactly k colors. Therefore

$$\omega(G) \leq \chi(G).$$

This bound is not accurate either. It is possible to construct graphs which do not contain a 3-vertex clique (a subgraph isomorphic with a triangle), and thus have $\omega(G) < 3$, and a chromatic number of an arbitrarily large value. A construction of a graph family with such a property was presented by Mycielski [281]. Furthermore, there is no known algorithm for determining $\omega(G)$ in the general case, as this problem is also NP-hard.

Another lower bound (worse, but easier to determine than the previous one) was given by Geller [102] in the form of the inequality

$$\frac{n^2}{n^2 - 2m} \leq \chi(G).$$

Just as in the case of the lower bound, the trivial upper bound for the chromatic number of a graph can also be strengthened. The following relation holds

$$(1.1) \quad \chi(G) \leq \Delta(G) + 1$$

This bound may be easily proven through induction on the number of vertices. Obviously, the inequality is true for a 1-vertex graph. Let G be a graph with n vertices. If we remove vertex v with all connected edges from graph G , we will obtain a graph G' , which fulfills the relation: $\Delta(G') \leq \Delta(G)$. From the induction assumption we have: $\chi(G') \leq \Delta(G') + 1$. An extension of the coloring of the graph G' to the entire graph G can be performed by coloring vertex v . As the degree of this vertex does not exceed $\Delta(G)$, there must always exist a free color for vertex v in a palette of $\Delta(G) + 1$ colors, no matter what color the neighbors of v are assigned.

There exist only two classes of graphs, for which the bound (1.1) takes the form of an equality. Brooks [37] proved that the equality is true only for odd cycles C_{2k+1} and complete graphs. Therefore if G is not a complete graph and $\Delta(G) \geq 3$, then

$$(1.2) \quad \chi(G) \leq \Delta(G).$$

It is easy to show a class of graphs, for which bound (1.2) is very inaccurate. Such a situation occurs for stars $S_k = K_{1,k}$. All stars are obviously 2-chromatic, while the discussed bound implies only $\chi(S_k) \leq k$.

Another known upper bound for the chromatic number of G takes the form

$$(1.3) \quad \chi(G) \leq \sqrt{2m} + 1.$$

This bound is more accurate for the aforementioned class of stars $K_{1,k}$, however there are also classes of graphs for which the distance between the chromatic number and the value of the bound is arbitrarily large. Stars (or, more generally, complete bipartite graphs) may serve as an example.

Apart from bounds for the chromatic number of general graphs, there exist certain trivial or nontrivial bounds for the chromatic number of particular classes of graphs. The most famous result in the theory of graph coloring, namely the 4-color theorem [7, 8], stating that any planar graph can be colored with four colors, is an appropriate example.

In practice, the best upper bounds for the chromatic number of particular graphs are imposed by the results of graph coloring by using suboptimal methods.

1.2.2. Heuristic methods of greatest practical importance. The high computational complexity of the graph coloring problem necessitates the use of heuristic approximation methods for the determination of suboptimal solutions in polynomial time. Several criteria for the assessment of the effectiveness of a heuristic method were discussed in the previous section.

For a given graph G and the sequence of vertices $K = (v_1, v_2, \dots, v_n)$ we will use the term *greedy coloring* to describe the following procedure of color assignment:

```

algorithm Greedy-Color( $G, K$ );
begin
  for  $v := v_1$  to  $v_n$  do
    give vertex  $v$  the smallest possible color;
end;

```

DEFINITION 1.22. A *sequential coloring algorithm* of graph G is an algorithm operating in the following two stages:

- (1) Determine a coloring sequence K of vertices of G .
- (2) Greedy-Color(G, K).

Color interchange is a relatively simple method of improving the effectiveness of any sequential coloring algorithm. It was first presented in paper [264]. It is based on the following observation: if the next greedy coloring step requires the assignment of a new color to vertex v_k , then it may turn out possible to swap the colors in one of the bichromatic subgraphs of graph G , and thus enable the assignment of a hitherto forbidden color to vertex v_k .

```

algorithm Color-with-Interchange( $G, K$ );
begin
  for  $v := v_1$  to  $v_n$  do begin
    if  $v$  requires a new color then
      attempt to swap the colors in  $G$  in order to free one of them for  $v$ ;
      assign the smallest possible color to  $v$ ;
    end
  end;

```

1.2.2.1. *RS method.* The RS method, also known as the *naive* method, is a sequential coloring approach which lacks the sequence-building stage of the algorithm. In other words, the vertices of the graph are randomly ordered for coloring. The method's name is derived from the term *random sequential* (RS). The method may be treated as a general determinant, describing the behavior of sequential methods for a given class of graphs.

```

algorithm RS-Color( $G$ );
begin
   $K :=$  a random sequence of vertices of graph  $G$ ;
  Greedy-Color( $G, K$ );
end;

```

The RS method can be implemented with computational time directly proportional to the size of graph G , i.e. $O(m + n)$. The family of Johnson's bipartite graphs J_k [186] can be used for the determination of the performance guarantee of this method. For every graph J_k it is possible to find a coloring sequence which requires exactly k colors. Therefore, the performance guarantee for the RS method is linear, i.e. $RS(n) = O(n)$. The path P_4 (shown in Fig. 6), is the SHC graph for the discussed method (the sequence of vertices presented in the figure leads to a suboptimal graph coloring). An HC graph does not exist for the RS method. The proof of this fact is left to the reader as a simple exercise.



FIGURE 6. Path P_4 — smallest slightly hard-to-color graph for method RS.

1.2.2.2. *LF method.* The method, whose name is derived from the words *largest-first* (LF), was first put forward by Welsh and Powell [345]. It is one of the oldest and simplest sequential methods. The LF method is based on the observation that vertices of low degree usually allow for a more flexible choice of color. Therefore it is natural to color the other vertices (i.e. vertices of high degree) in first priority. The LF method has proved effective, despite its simplicity.

```

algorithm LF-Color( $G$ );
begin
   $K :=$  vertices of  $G$  arranged in non-increasing order of degree;
  Greedy-Color( $G, K$ );
end;

```

The LF method can be implemented to run in $O(m + n)$ time. The family of Johnson's bipartite graphs J_k can also be used to find the performance guarantee of this method. In the worst case, the graphs J_k are colored by the LF method with $k = n/2$ colors. In consequence, the performance guarantee for the LF method is linear, $LF(n) = O(n)$. The path P_6 , presented in Fig. 7a, is the SHC graph for the discussed method, while the $HC(LF)$ graph is known as the *envelope* graph, shown in Fig. 7(b).

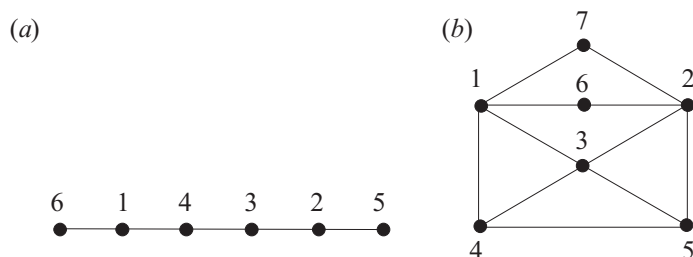


FIGURE 7. (a) path P_6 — smallest slightly hard-to-color graph for the LF method; (b) envelope graph — smallest hard-to-color graph for the LF method

1.2.2.3. *SL method.* The SL sequential coloring method was first suggested by Matula et al. [264] under the original name *smallest-last*. Like the LF method, the SL method is based on the observation that vertices with few neighbors ought to be colored as late as possible. Owing to a certain refinement in the process of creating sequences of vertices, the SL method does not have certain faults of the LF algorithm, for instance SL optimally colors trees, cycles, unicyclic graphs, wheels, complete bipartite graphs, Johnson's graphs [186] and Mycielski's graphs [281]. Moreover, the method also colors k -degenerate graphs using at most $k + 1$ colors. For planar graphs, the SL method will never use more than 6 colors.

```

algorithm SL-Color( $G$ );
begin
   $K := \emptyset$ ;
  while  $V \setminus K \neq \emptyset$  do
    append to  $K$  the vertex  $v$  with the smallest degree in the subgraph
    generated by  $V \setminus K$ ;
    Greedy-Color( $G, \overline{K}$ );  $\{\overline{K}$  denotes the inversion of order of sequence  $K\}$ 
end;

```

Just as the aforementioned methods, the SL method can also be implemented to run in $O(m + n)$ time. A family of bipartite graphs described by Coleman and More [57] can be used for the analysis of the performance guarantee of the SL method. These graphs, denoted as CM_k , are colored by SL with $k = n/6$ colors in the worst case. As a result, the performance guarantee for the LF method is linear, $SL(n) = O(n)$. The SHC graph for the SL method is called a *prism* graph, Fig. 8(a), while the HC is known as a *prismatoid*, Fig. 8(b).

1.2.2.4. *RSI method.* The RSI method was created as a simple modification of the RS method, achieved by introducing the earlier described mechanism of color interchange. Like RS, the RSI method can be treated as a general determinant describing the behavior of sequential methods for a given class of graphs. Because the color interchange procedure can be implemented to operate in $O(m)$ time, and greedy coloring in $O(n)$ time, it follows that the entire algorithm can be designed to work in $O(mn)$ time.

```

algorithm RSI-Color( $G$ );
begin

```

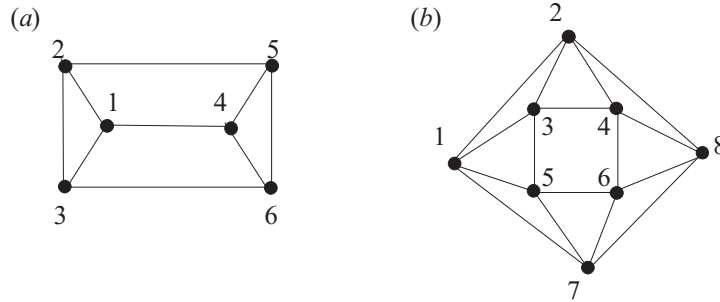


FIGURE 8. (a) Prism graph P_m — smallest slightly hard-to-color graph for the SL method; (b) Prismaoid P_d — smallest hard-to-color graph for the SL method.

```

     $K :=$  a random sequence of vertices of graph  $G$ ;
    Color-with-Interchange( $G, K$ );
end;

```

The performance guarantee of the method is $RSI(n) = O(n)$. The first slightly hard-to-color graph for the RSI method was given by Syso et al. [317]. The graph had 45 vertices and it was discovered through a search of random graphs, conducted by means of computer techniques. The SHC graph for the RSI method is shown in Fig. 9, while an HC graph does not exist, as in the case of the RS method.

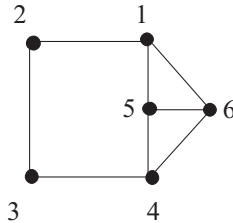


FIGURE 9. Smallest slightly hard-to-color graph for the RSI method.

1.2.2.5. LFI method. The LFI method is a modified version of the LF method, which includes the color interchange procedure. As in the case of RSI, the LFI algorithm can be implemented to operate in $O(mn)$ time.

```

algorithm LFI-Color( $G$ );
begin
     $K :=$  vertices of  $G$  arranged in non-increasing order of degree;
    Color-with-Interchange( $G, K$ );
end;

```

The smallest hard-to-color graphs for this method are presented in Fig. 10.

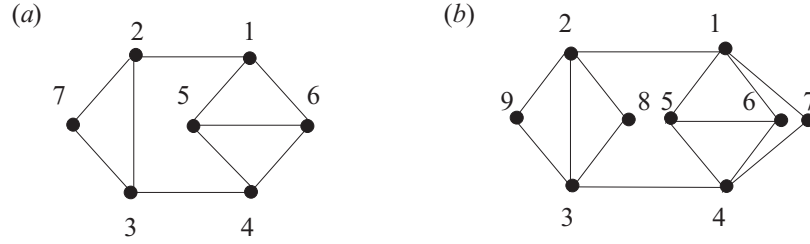


FIGURE 10. (a) Smallest slightly hard-to-color graph for the LFI method; (b) Smallest hard-to-color graph for the LFI method.

1.2.2.6. *SLI method.* The SLI method is a modification of algorithm SL, analogous to LFI. Just as the methods discussed earlier, it takes advantage of the color interchange mechanism and can run in $O(mn)$ time.

algorithm SLI-Color(G);
begin
 $K := \emptyset$;
while $V \setminus K \neq \emptyset$ **do**
 append to K the vertex v with the smallest degree in the subgraph
 generated by $V \setminus K$;
 Color-with-Interchange(G, \overline{K});
 $\{\overline{K}$ denotes the inversion of order of sequence K $\}$
end;

The *SHC*(SLI) and *HC*(SLI) graphs are shown in Fig. 11.

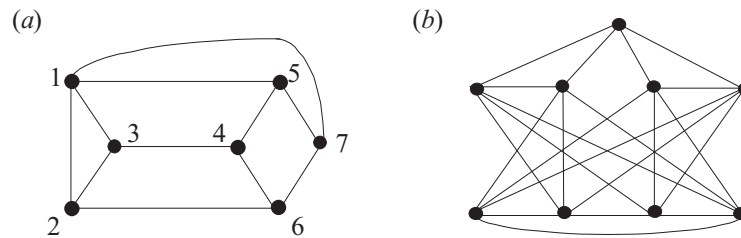


FIGURE 11. (a) Smallest slightly hard-to-color graph for the SLI method; (b) Smallest hard-to-color graph for the SLI method.

1.2.2.7. *CS method.* The CS method is a typical example of a sequential coloring algorithm. It is based on the observation that in each iteration of the algorithm only vertices adjacent to those already colored should be regarded as candidates for coloring (hence the method's full name — *connected sequential*). The presented approach helps to avoid suboptimal coloring in many situations. Trivial graphs, such as the path P_4 , are always colored in an optimal way by the CS algorithm.

algorithm CS-Color(G);
begin
 $K :=$ vertices of G ordered in such a way that each vertex (except
the first) one has at least one neighbour in the preceding part of the

```

sequence;
Greedy-Color( $G, K$ );
end;

```

It is possible to implement the CS method to run in $O(m+n)$ time, for instance with the aid of breadth-first or depth-first graph search techniques. Analysis of the performance guarantee for this method can be carried out in the same way as for the RS method. We obtain $CS(n) = O(n)$.

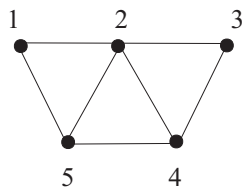


FIGURE 12. Fan F_5 — smallest slightly hard-to-color graph for the CS method.

The fan presented in Fig. 12 is the SHC graph for this coloring method, while the smallest hard-to-color graph is unknown. Babel and Tinhofer [11] proved that the graph named the *twin dragon* and denoted by the symbol TD , is a hard-to-color graph for CS (Fig. 13).

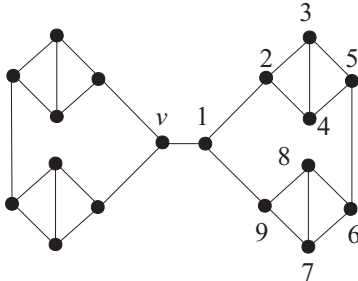


FIGURE 13. Twin dragon TD — smallest known hard-to-color graph for the CS method

1.2.2.8. *SLF method.* This method is not based on a typical sequential algorithm, but on the principle of dynamic vertex ordering. It was first presented by Brélaz [36] under the name DSATUR or *saturation LF* (SLF). It may be regarded as a sensible modification of the LF method based on the observation that the constraints in color assignment do not directly result from the degree of the colored vertex, but from the number of its uniquely colored neighbors.

```

algorithm SLF-Color( $G$ );
begin
  while not all vertices of  $G$  have been colored do begin

```

```

     $v :=$  the vertex with the highest saturation degree  $\rho(v)$  in  $G$ 
    {ties are broken by choosing the vertex with the highest possible
    degree  $\deg(v)$  in  $G$ };
    assign the smallest possible color to  $v$ ;
  end
end;

```

The SLF method can be implemented to run in $O((m+n)\log n)$ time [331]. The performance guarantee of the algorithm is linear, i.e. $\text{SLF}(n) = O(n)$. The method leads to optimal colorings of all bipartite graphs [36], cycles, mono- and bicyclic graphs, trees, necklaces and cacti, as well as all graphs, whose core is a member of one of the aforementioned families [176]. Moreover, the SLF algorithm optimally colors nearly all k -chromatic graphs [331]. The smallest hard-to-color graphs for the SLF method are shown in Fig. 14.

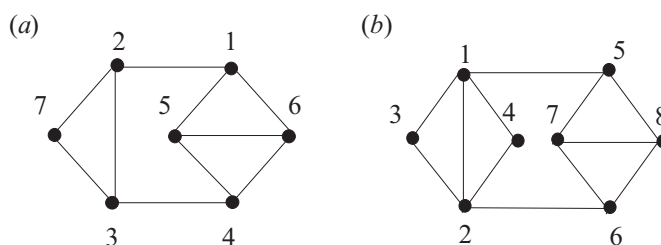


FIGURE 14. (a) Smallest slightly hard-to-color graph for the SLF method; (b) Smallest hard-to-color graph for the SLF method

1.2.2.9. *GIS method.* The GIS method was first suggested by Johnson [185]. It is an implementation of the maximum independent set algorithm, hence its name — *greedy independent sets* (GIS). The vertices of graph G are analyzed in a certain order and vertex v_i is assigned color c if v_i is not adjacent to any vertex already colored with c . Having assigned color c to all possible vertices, the algorithm removes them and repeats the same procedure for the remaining subgraph of G and color $c + 1$. This approach is the result of the observation that, for instance, the first color in the first place ought to be assigned to the vertices with fewest uncolored neighbors (thus blocking the possibility of using the considered color for the smallest possible number of vertices).

```

algorithm GIS-color( $G$ );
begin
   $c := 1$ ;
  while not all vertices of  $G$  have been colored do begin
    available := set of uncolored vertices of  $G$ ;
    while available  $\neq \emptyset$  do begin
       $v :=$  vertex with the minimal degree in the subgraph generated
      by available;
      assign color  $c$  to  $v$ ;
      remove vertex  $v$  and its neighbors from available;
    end;
     $c := c + 1$ ;
  end;
end;

```

end;
end;

The GIS method can be implemented to run in $O(mn)$ time. It is interesting to note that GIS is one of the few known heuristic algorithms with a sublinear performance guarantee, namely $\text{GIS}(n) = O(n/\log n)$.

GIS was the first algorithm for which a suboptimally colored graph was found (by Johnson [185]). The graph in question is the path P_4 . Both the SHC and HC graphs for the GIS algorithm are presented in Fig. 15.

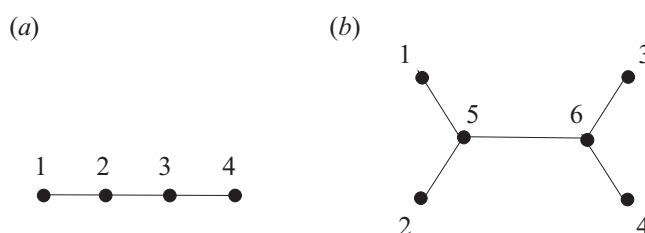


FIGURE 15. (a) path P_4 — smallest slightly hard-to-color graph for the GIS method; (b) double star $DS_{2,2}$ — smallest hard-to-color graph for the GIS method

1.2.3. Known benchmarks. The hard-to-color and slightly hard-to-color graphs for the three best known and most frequently used heuristic methods, i.e. SL, LF, SLF, may be regarded as the smallest nontrivial benchmarks. The graph presented in Fig. 16 is a benchmark for the family of algorithms LF, SL, whereas the graph in Fig. 17 is a benchmark for the family of algorithms LF, SL, SLF.

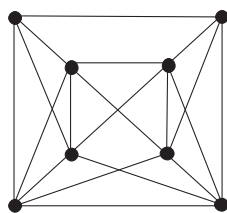


FIGURE 16. A benchmark for the family of algorithms LF and SL.

Graphs which are hard to color for sequential methods (of practical significance) and for methods without color interchange are also considered to be interesting benchmarks. The graph shown in Fig. 18 is a benchmark for the family of algorithms LF, SL, SLF, LFI and SLI, while the graph in Fig. 19 is a benchmark for the family of algorithms LF, SL, SLF, GIS.

The graph presented in Fig. 20 is a weak benchmark for the family of algorithms LF, SL, CS, SLF, SLI, LFI, GIS.

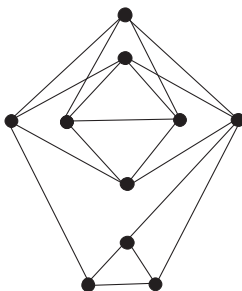


FIGURE 17. A benchmark for the family of algorithms LF, SL and SLF

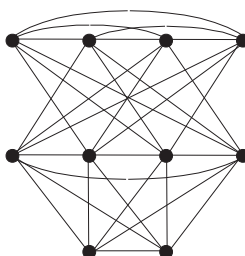


FIGURE 18. A benchmark for the family of widely used sequential algorithms

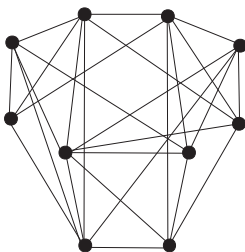


FIGURE 19. A benchmark for the family of widely used sequential algorithms without color interchange

1.3. Classical edge-coloring

The idea of coloring the edges of a graph has been investigated for many years due to the large number of its practical applications. The *classical edge-coloring* model leads to an NP-hard optimisation problem [165].

DEFINITION 1.23. An *edge-coloring* of graph $G = (V, E)$ is a function $c: E \rightarrow \mathbb{N}$, in which any two adjacent edges $e, f \in E$ are assigned different colors, i.e. $(e \neq f \wedge e \cap f \neq \emptyset) \implies c(e) \neq c(f)$. The function c is known as the *edge-coloring function*.

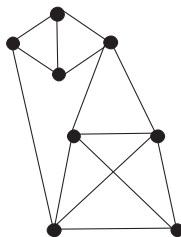


FIGURE 20. A weak benchmark for the family of widely used sequential algorithms

DEFINITION 1.24. A graph G for which there exists an edge-coloring which requires k colors is called *k -edge colorable*, while such a coloring is called a *k -edge-coloring*. The smallest number k for which there exists a k -edge-coloring of graph G is called the *chromatic index* of graph G and is denoted by $\chi'(G)$. Any edge-coloring of G which requires $k = \chi'(G)$ colors is called *chromatic* or *optimal*.

1.3.1. Problem complexity and simplest bounds. The problem of coloring the edges of a graph belongs to the family of NP-hard problems, just as the problem of vertex-coloring. In fact, the task of determining an optimal edge-coloring of graph G can be solved by finding an optimal vertex-coloring of a graph known as the line graph of G .

DEFINITION 1.25. The *line graph* of graph G , denoted by $L(G)$, is a graph in which every vertex corresponds to exactly one edge of G , and any two vertices of graph $L(G)$ are adjacent if and only if the corresponding edges of graph G are adjacent.

Very strong bounds for the chromatic index are known despite the high computational complexity of the edge-coloring problem. The lower bound for the chromatic index is an obvious conclusion from the definition of edge-coloring

$$(1.4) \quad \Delta(G) \leq \chi'(G).$$

On the other hand, in 1964 Vizing [339] proved a surprisingly sharp upper bound

$$(1.5) \quad \chi'(G) \leq \Delta(G) + 1.$$

It follows from inequalities (1.4) and (1.5) that the value of the chromatic index is nearly determined by the degree of the graph and can only take two values: $\Delta(G)$ or $\Delta(G) + 1$. This relation gave rise to the division of all graphs into two classes.

DEFINITION 1.26. Graph G belongs to *Class 1* if $\chi'(G) = \Delta(G)$. Likewise, graph G is called a graph of *Class 2*, if $\chi'(G) = \Delta(G) + 1$.

Through extensive computer search and theoretical investigation it has been shown that the number of n -vertex graphs of Class 2 is much smaller than the number of graphs of Class 1. Examples of Class 1 graphs include bipartite graphs, complete graphs of even order K_{2k} and wheels. By contrast, cycles of odd order, complete graphs of odd order K_{2k+1} and snarks are examples of Class 2 graphs.

1.3.2. Typical heuristic methods — known results. Because the determination of the value of the chromatic index remains an NP-hard problem despite the strength of known bounds, and — more importantly — because numerous applications of edge-coloring require not only the value of the chromatic index but also an assignment of colors, effective heuristics solving the problem of edge-coloring remain indispensable. On the whole, there are two different approaches to determining suboptimal edge-colorings of a graph. The first approach relies on the use of a wide variety of heuristic methods for coloring line graphs. The alternative approach concentrates on dedicated methods, designed specially for the edge-coloring of graphs. The naive (greedy) edge-coloring algorithm as well as Vizing’s widely used method belong to the latter group.

The greedy method requires fewer than $2\chi'(G)$ colors to color graph G , while Vizing’s method never uses more than $\chi'(G) + 1$ colors. Therefore, edge-coloring methods based on the principle of vertex-coloring of line graphs are not widely used in practice.

1.3.2.1. NC naive coloring method. The NC method, also referred to as the *greedy* method, is based on the principle of coloring graph edges in a random order and assigning the smallest possible color to the currently colored edge [216]. It may be regarded as a counterpart of the RS method for vertex-coloring. Despite its simplicity, NC is a 2-relative approximation algorithm. The proof of this fact can be found e.g. in [216]. The NC method can be implemented to run in $O(mn)$. The path P_5 is the SHC graph for the NC algorithm, whereas no HC graph exists for this method (as in the case of RS and RSI vertex-coloring methods).

1.3.2.2. NTL method. The name of the NTL method is derived from the first letters of the names of its creators [285]. The NTL algorithm is based on the Recolor procedure, which brings about an improvement of edge-coloring through a mechanism of color swapping. It is useful to introduce the following definitions in order to facilitate the description of the NTL method.

DEFINITION 1.27. The term *missing color for vertex* $v \in V$ of graph $G = (V, E)$ is used to describe a color which has not been assigned to any edge incident to v . The symbol $M(v)$ will be used to denote the set of all missing colors for vertex v .

DEFINITION 1.28. If every vertex $v \in V$ of graph $G = (V, E)$ is associated with exactly one selected missing color, denoted as $m(v)$, then the term *fan* F around vertex v beginning with edge $\{v, u\}$ is used to refer to the sequence of edges $(\{v, w_0\}, \{v, w_1\}, \dots, \{v, w_s\})$, where $w_0 = u$ and $\{v, w_i\}$ is assigned color $m(w_{i-1})$. The number s is known as the *span* of the fan.

PROPERTY 1.29. *Let us assume, that all the edges of graph G with the exception of $\{u, v\}$ were colored with $\Delta(G) + 1$ colors. Obviously, both u and v have at least two missing colors. If F is the maximal fan around v , then either the color $m(w_s)$ is missing for vertex v , i.e. $m(w_s) \in M(v)$, or a certain edge of F has been assigned the color $m(w_s)$. Of course, when $m(u) \in M(v)$, fan F consists of only one edge $\{u, v\}$.*

Procedure Recolor. In order to color the edge $\{u, v\}$ we first determine the maximal fan F around vertex v . If $m(w_i) \in M(v)$, then for every $0 \leq i \leq s$ (where s is the span of the fan) we assign the color $m(w_i)$ to edge $\{u, w_i\}$. Otherwise, let P be a path in graph G , beginning in w_s and colored alternately with colors $m(v)$ and $m(w_s)$. If P does not reach v , we can swap the colors used in P and

cyclically shift the colors of edges $\{v, w_i\}$, painting them with color $m(v_i)$ for all $0 \leq i \leq s$, and then painting $\{v, w_s\}$ with color $m(v)$. If, on the other hand, path P reaches vertex v , then let us denote by w_j ($0 \leq j \leq s - 2$) the vertex which fulfills the equation $m(w_{j-1}) = m(w_s)$. Next, let P be a path beginning with vertex w_j and painted with colors $m(v)$ and $m(w_s)$. In this case it is necessary: to shift all the colors of the fan cyclically by painting each edge $\{v, w_i\}$ with color $m(w_i)$ for $0 \leq i \leq j - 2$, to swap the colors along path P , and finally to color edge $\{v, w_{j-1}\}$ with color $m(v)$.

An implementation of the NTL method which uses the described Recolor procedure is presented below.

```

algorithm NTL-color( $G$ );
begin
  if  $\Delta(G) \leq 2$  then
    color  $G$  greedily, traversing paths and cycles
  else begin
     $q := \Delta(G) + 1$ ;  $G' := (V, \emptyset)$ ;
    for each  $e \in E$  do begin
       $G' := G' + e$ ;
      if  $e = \{u, v\}$  cannot be assigned a common missing color for
        vertices  $u$  and  $v$  then Recolor( $u, v$ );
      color  $e$ ;
    end
  end
end;

```

The Recolor procedure can be performed in linear time, therefore the computational complexity of algorithm NTL is $O(mn)$. NTL is a 1-absolute approximation algorithm, therefore at most four colors will be used to paint a graph with a chromatic index of 3. Thus NTL can be regarded as a 4/3-relative approximation algorithm. Such an algorithm yields optimal colorings of an extensive family of graphs, including cycles, wheels, stars, double stars, planar graphs ($\Delta \geq 9$) and nearly all random graphs [285]. The SHC graph for the NTL method is known as the *bull* and is presented in Fig. 21. As in the case of greedy algorithms, no HC graph exists for the NTL method.

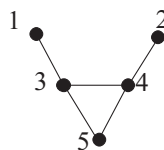


FIGURE 21. Bull — smallest slightly hard-to-color graph for the NTL method

The bull is at the same time a weak benchmark for the family of algorithms NC, NTL.