Colorings With Few Colors: Counting, Enumeration and Combinatorial Bounds*

Petr A. Golovach[1], Dieter Kratsch[2], and Jean-Francois Couturier[2]

[1] School of Engineering and Computing Sciences, Durham University, South Road, Durham, DH1 3LE, United Kingdom.
petr.golovach@durham.ac.uk
[2] Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France.
(kratsch|couturier)@univ-metz.fr

Abstract. We provide exact algorithms for enumeration and counting problems on edge colorings and total colorings of graphs, if the number of (available) colors is fixed and small. For edge 3-colorings the following is achieved: there is a branching algorithm to enumerate all edge 3-colorings of a connected cubic graph in time $O^*(2^{5n/8})$. This implies that the maximum number of edge 3-colorings in an n-vertex connected cubic graph is $O^*(2^{5n/8})$. Finally, the maximum number of edge 3-colorings in an n-vertex connected cubic graph is lower bounded by $12^{n/10}$. Similar results are achieved for total 4-colorings of connected cubic graphs. We also present dynamic programming algorithms to count the number of edge k-colorings and total k-colorings for graphs of bounded pathwidth. These algorithms can be used to obtain fast exact exponential time algorithms for counting edge k-colorings and total k-colorings on graphs, if k is small.

1 Introduction

Graph coloring is one of the classical subjects in graph theory. The four color conjecture asking whether every planar graph can be vertex-colored using at most 4 colors has been triggering the research in graph theory for more than a century. From an algorithmic point of view, for many coloring type problems, like vertex coloring, edge coloring and total coloring, the existence problem asking whether the graph has a coloring with a given number of colors is NP-complete. Even more, these coloring problems remain NP-complete when the question is whether there is a coloring of the input graph with a fixed (and small) number of colors [12, 13, 20]. (For Definitions see Section 2.)

* The first author has been supported by EPSRC under project EP/G043434/1. The second and third author have been supported by ANR Blanc AGAPE (ANR-09-BLAN-0159-03).

Exact algorithms to solve NP-hard problems are a challenging research subject in graph algorithms. Many papers on exact exponential time algorithms have been published in the last decade. One of the major results is the $O^*(2^n)$ inclusion-exclusion algorithm to compute the chromatic number of a graph first presented at FOCS 2006 by Björklund, Husfeldt [2] and Koivisto [14]. [3] This approach can also be used to establish a $O^*(2^n)$ algorithm to count the k-colorings and to compute the chromatic polynomial of a graph. It also implies a $O^*(2^m)$ algorithm to count the edge k-colorings and a $O^*(2^{n+m})$ algorithm to count the total k-colorings of the input graph.

The existence problem asking whether a graph has a k-coloring for a fixed and small value of k also attracted a lot of attention. For vertex-colorability the fastest algorithm for $k = 3$ has running time $O^*(1.3289^n)$ and was proposed by Beigel and Eppstein [1], and the fastest algorithm for $k = 4$ has running time $O^*(1.7272^n)$ and was given by Fomin et al. [7]. They also established algorithms for counting vertex k-colorings for $k = 3$ and 4 [7]. The existence problem for an edge 3-coloring is considered in [1, 15] and the currently fastest algorithm with the running time $O(1.344^n)$ is due to Kowalik [15]. Very recently Björklund et al. showed how to detect whether a d-regular graph admits an edge d-coloring in time $O^*(2^{(d-1)n/2})$ [3].

Combinatorial bounds on the maximum number of combinatorial objects in any n-vertex graph, as e.g. maximal independent sets or k-colorings, are of interest in combinatorics. Such upper bounds can sometimes be achieved via algorithms to enumerate all these objects. A well-known example is a branching algorithm to enumerate all maximal independent sets of a graph that can be used to establish an $O^*(3^{n/3})$ upper bound for the number of maximal independent sets in an n-vertex graph. Originally in 1965 Moon and Moser showed by an inductive proof that the maximum number of maximal independent sets in an n-vertex graph is $3^{n/3}$ [18]. Another interesting example is the upper bound of $O(1.7159^n)$ on the number of minimal dominating sets of an n-vertex graph established via a branching enumeration algorithm and its Measure & Conquer analysis by Fomin et al. [10].

Our Results. For edge 3-colorings we achieve the following enumeration algorithm and related combinatorial bounds.

- There is a branching algorithm to enumerate all edge 3-colorings of a connected cubic graph of running time $O^*(2^{5n/8}) = O(1.5423^n)$ using polynomial space (Subsection 3.1).
- The maximum number of edge 3-colorings in an n-vertex connected cubic graph is at most $O^*(2^{5n/8}) = O(1.5423^n)$ (Subsection 3.1).
- The maximum number of edge 3-colorings in an n-vertex connected cubic graph is lower bounded by $12^{n/10} = \Omega(1.2820^n)$ (Subsection 3.3).

For the counting problem of edge k-colorings of graphs we achieve the following algorithms.

[3] As has recently become standard, we write $f(n) = O^*(g(n))$ if $f(n) \leq p(n) \cdot g(n)$ for some polynomial $p(n)$.

– The edge k-colorings of a graph given with a path decomposition of width at most p can be counted in time roughly $O^*(\binom{k}{\lfloor k/2 \rfloor}^{p+1})$ by a dynamic programming algorithm using exponential space (Section 4). [4]
– The number of edge 3-colorings of a graph can be counted in time $O^*(3^{n/6}) = O(1.201^n)$ and exponential space (Section 4).
– The number of edge 4-colorings of a graph can be counted in time $O^*(6^{n/3}) = O(1.8172^n)$ and exponential space (Section 4).

Note that our algorithm to count edge 3-colorings in time $O(1.201^n)$ improves upon the $O(1.344^n)$ time of Kowalik's polynomial space branching algorithm solving the decision problem [15].

For total k-colorings of graphs we achieve the following results.

– There is a branching algorithm to enumerate the total 4-colorings of a connected cubic graph in time $O^*(2^{13n/8}) = O(3.0845^n)$, implying that the maximum number of total 4-colorings in an n-vertex connected cubic graph is at most $O^*(2^{13n/8}) = O(3.0845^n)$ (Subsection 3.2).
– The number of total k-colorings of a graph given with a path decomposition of width at most p can be counted in time roughly $O^*(\left(k \cdot \binom{k-1}{\lfloor (k-1)/2 \rfloor}\right)^{p+1})$ (Section 4). [4]
– The number of total 4-colorings of a graph G can be counted in time $O(12^{n/6}) = O(1.5131^n)$ (Section 4).

Let us emphasize that edge 3-colorings and total 4-colorings exist only for graphs of maximum degree at most 3. Furthermore the largest number of such colorings for connected graphs is achieved by the n-vertex path. To avoid such trivial cases, it is natural to study these problems on connected cubic graphs; a well-known class of graphs for which upper bounding the number of colorings (of both types) is a non trivial and challenging task.

Furthermore we achieved similar results for the $L(2,1)$-labeling problem of graphs. Due to space restrictions only a short summary is given in Section 5. For the same reason some proofs will be omitted.

2 Preliminaries

We consider finite undirected graphs without loops or multiple edges. The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$, or simply by V and E if this does not create confusion. For a set of edges $S \subseteq E(G)$, $G-S$ is the graph obtained from G by removing the edges of S. We denote by $\deg_G(v)$ the *degree* of a vertex v. For a vertex v, $E_G(v)$ is the set of edges incident with v. We may omit indices if the graph under consideration is clear from the context. The maximum degree of a graph G is denoted by $\Delta(G)$. Let r be a positive integer. A graph G is called *r-regular* if all vertices of G have degree r. A *cubic* graph is a 3-regular graph.

[4] See Theorem 3 for a precise estimation of the running time.

Let k be a positive integer. A *vertex (edge) k-coloring* of a graph G is an assignment $c\colon V(G) \to \{1,\dots,k\}$ ($c\colon E(G) \to \{1,\dots,k\}$ respectively) of a positive integer (*color*) to each vertex (edge) of G such that adjacent vertices (edges) receive distinct colors. A *total k-coloring* of a graph G is a mapping $c\colon V(G) \cup E(G) \to \{1,\dots,k\}$ such that adjacent vertices and adjacent edges have different colors, and for any edge e incident to vertex v, $c(v) \neq c(e)$. The *chromatic number* of G (denoted by $\chi(G)$) is the minimum k such that there is a vertex k-coloring of G. The *chromatic index* (or *edge chromatic number*) is the smallest k for which an edge k-coloring of G exists. We denote the chromatic index by $\chi'(G)$. The *total coloring number* $\tau(G)$ is the minimum k for which there is a total k-coloring of G.

It is easy to see that $\chi'(G) \geq \Delta(G)$ and $\tau(G) \geq \Delta(G)+1$ for every graph G. Let us recall that by the well-known theorem of Vizing $\Delta(G) \leq \chi'(G) \leq \Delta(G)+1$ [21]. Also it is known that for any graph G with maximum degree at most three, there is a total 5-coloring of G [19].

3 Enumeration algorithms for cubic graphs

3.1 Edge 3-colorings

We present a branching algorithm to enumerate all edge 3-colorings of a given connected cubic graph. Our branching algorithm consists of a recursive procedure `EnumCol` and two auxiliary subroutines `Color` and `Extend`.

Procedure `EnumCol`(S, c);
1 `Extend`(S, c);
2 **if** *the graph $H = G - S$ contains a component F s.t. F is not a cycle and not an isolated vertex* **then**
  choose a vertex $v \in V(F)$ s.t. $\deg_F(v) < 3$ and edge $e \in E(F)$ incident with v; let $\{\alpha, \beta\} = \{1,2,3\} \setminus \{c(S \cap E_G(v))\}$;set $S' = S, c' = c$;
  if *`Color`$(S, c, e, \alpha) = true$* **then** `EnumCol`(S, c);
  if *`Color`$(S', c', e, \beta) = true$* **then** `EnumCol`$(S', c')$;
3 **if** *the graph $H = G - S$ contains a component F s.t. F is an odd cycle and there is $\alpha \in \{1,2,3\}$ s.t. for all $v \in V(F)$, $c(S \cap E_G(v)) = \alpha$* **then** `Halt`;
4 **if** *there is an edge $\{u, v\}$ in $H = G - S$ s.t. $c(S \cap E_G(u)) \neq c(S \cap E_G(v))$* **then**
  let $\alpha \in \{1,2,3\} \setminus c(S \cap (E_G(u) \cup E_G(v))$;`Color`$(S, c, \{u, v\}, \alpha)$;
  `EnumCol`(S, c);
5 **if** *the graph $H = G - S$ contains a component F s.t. F is a cycle and there is $\alpha \in \{1,2,3\}$ s.t. for all $v \in V(F)$, $c(S \cap E_G(v)) = \alpha$* **then**
  let $e \in E(F)$ and let $\{\beta, \gamma\} = \{1,2,3\} \setminus \{\alpha\}$;set $S' = S, c' = c$;
  `Color`(S, c, e, β);`EnumCol`(S, c);
  `Color`(S', c', e, γ);`EnumCol`(S', c');
6 **if** $S = E(G)$ **then** `Output`(c)

The procedure `EnumCol` takes as input a connected cubic graph G and a set of colored edges $S \subseteq E(G)$, i.e. for each $e \in S$, the assigned color $c(e) \in \{1,2,3\}$ is given. The procedure enumerates all edge 3-colorings of G which are extensions

of the given edge coloring of S. Note that if v is an isolated vertex in $G-S$ then all edges incident to v are colored, and thus v is of no importance for extending the partial edge coloring c.

The subroutine `Color` takes as input a set of colored edges S with a partial edge coloring c, an edge $\{u,v\} \in E(G) \setminus S$ and a color $\alpha \in \{1,2,3\}$. It tries to extend c by assigning to edge $\{u,v\}$ the color α. If possible the subroutine returns $true$, otherwise it returns $false$.

Subroutine `Color`$(S, c, \{u,v\}, \alpha)$**;**
if $\alpha \notin c(S \cap (E_G(u) \cup E_G(v)))$ **then**
  set $c(\{u,v\}) = \alpha$, $S = S \cup \{e\}$;`Return(`*true*`);`
else `Return(`*false*`)`

The subroutine `Extend` tries to extend a given edge coloring c of S by reduction (i.e. without branching) if there is a vertex incident with two already colored edges.

Subroutine `Extend`(S, c)**;**
while *there is a vertex* $v \in V(G)$ *s.t.* $|S \cap E(v)| = 2$ **do**
  let $\{\alpha, \beta\} = S \cap E(v)$, $\gamma \in \{1,2,3\} \setminus \{\alpha, \beta\}$ *and* $e \in E(v) \setminus S$;
  if *Color*$(S, c, e, \gamma) = false$ **then Halt**

To enumerate all edge 3-colorings of a connected cubic G we choose any edge $e \in E(G)$, set $S = \{e\}$ and call the procedure `EnumCol` consecutively for $c(e) = 1$, $c(e) = 2$ and $c(e) = 3$. If the aim is to enumerate all edge 3-colorings up to permutations of colors then we choose two adjacent edges $e_1, e_2 \in E(G)$, set $S = \{e_1, e_2\}$, $c(e_1) = 1$, $c(e_2) = 2$ and call `EnumCol`.

Theorem 1. *Our algorithm enumerates all edge 3-colorings of a connected cubic graph G in time $O^*(2^{5n/8})$.*

Proof. To prove the correctness of the algorithm we consider Procedure `EnumCol`. By the step 1 we try to extend the coloring c of S without branching looking for vertices incident with two already colored edges. By the step 2 we choose a vertex v with exactly one incident colored edge, and then branch using an uncolored edge e incident with v. Clearly, we have two possibilities to color this edge. Notice that e is an edge of the component of $H = G - S$ which is not a cycle. If for the current input G, S, c the steps 1 and 2 are not applicable anymore, then since G is a connected graph, every vertex is incident to at least one edge of S. Hence the maximum degree of $G - S$ is at most two, which implies that $G - S$ is a union of paths, cycles and isolated vertices. Furthermore no component can be a path since such a component would be colored by applying step 1 successively to an end vertex of the path. Thus in steps 3–5 all non empty components of $H = G - S$ are cycles. Suppose F is a component of H which is a cycle. If F is an odd cycle and all vertices of F are incident with colored edges from S which are colored by the same color, then all edges of F have to be colored by the two remaining color, but this is impossible. This case is checked in the step 3. If F has at least two vertices which are incident with edges of S colored by different colors, then F contains two adjacent vertices u and v with the same property.

Since there is only one possibility to color $\{u, v\}$, all edges of F can be colored without branching, and it is done in step 4 and by a recursive call of `EnumCol`. In step 5 we consider the final case when all non empty components of H are even cycles, and for each cycle F, all vertices of F are incident with the edges of S colored by the same color. Then two edge colorings of F are possible and each is generated via a recursive call of `EnumCol`.

Now we estimate the running time. Consider the graph G_S which is the subgraph of G with edge set S and the set of all end vertices of S as vertex set. Notice that by each successive execution of step 2 and step 1 for the next recursive call of `EnumCol`, we add at least two vertices to the graph G_S by our choice of the edges for the branching in step 2. It follows from the fact that after a call of the subroutine `Extend` all non isolated vertices of $G - S$ have degree at least 2 in this graph, and then the `Extend` subroutine colors edges along two paths in $G - S$ starting at the vertex v chosen for the branching and ending at a degree 3 vertex in $G - S$ for each of them. Observe also that if these two paths reach the same vertex then the subroutine extends the coloring along a new path starting at this vertex until we reach a new vertex of degree 3. This means that the depth of the search tree generated by branchings in step 2 only, and not considering steps 3–6, is at most $\frac{n-2}{2}$ (recall that G_S has two vertices when `EnumCol` is called first). The branching on the steps 3–5 is done only when all non empty components of H are even cycles, and for each cycle, only one binary branching is done. Let us estimate the number of such cycles. Suppose that the set S constructed by `EnumCol` is such that all non empty components of $H = G-S$ are cycles. Since G is a connected graph and by the choice of the edges for the branching, G_S is a connected graph with the vertex set $V(G)$. Notice that G_S has only vertices of degree one or three. Let n_1 be the number of vertices of degree one, and let n_3 be the number of vertices of degree three. Clearly, G_S has $\frac{1}{2}(n_1 + 3n_3)$ edges, and since it is connected, $\frac{1}{2}(n_1 + 3n_3) \geq n_1 + n_3 - 1$. Hence, $\frac{n+2}{2} \geq n_1$ and therefore the number of vertices of degree 2 in $H = G - S$ is at most $\frac{n+2}{2}$ implying that H has at most $\frac{n+2}{8}$ even length cycles. Therefore, the depth of the overall search tree is at most $\frac{n-2}{2} + \frac{n+2}{8}$. Since for each branching, we consider two cases, the number of leaves in the search tree is at most $2^{\frac{n-2}{2}+\frac{n+2}{8}}$, and the running time of the algorithm is $O^*(2^{5n/8})$. □

Using the fact that edge 3-colorings correspond to leaves of the search tree, we have the following corollary.

Corollary 1. *Let G be a connected cubic graph with n vertices. Then the number of different edge 3-colorings of G is at most $3 \cdot 2^{(5n-6)/8}$.*

3.2 Total 4-colorings

Using an algorithm similar to the one of the previous subsection it is possible to enumerate the total 4-colorings of connected cubic graphs. The major ingredient to be added to the above algorithm is that a vertex is colored as soon as one of its incident edges is colored.

Theorem 2. *All total 4-colorings of a connected cubic graph G can be enumerated in time $O^*(2^{13n/8})$.*

Proof. We discuss those properties that are different from the algorithm in the previous subsection.

Notice that if $\{u, v\}$ is a colored edge and the vertex u is colored too, then there are at most two possibilities to color v. Thus whenever a vertex is colored (except the first two) at most two colors are possible. If a vertex v is colored and it is incident with one colored edge, then there are at most two possibilities to color the remaining edges incident to v. Also, if a vertex v is colored and it is incident with two colored edge, then there is at most one possibility to color the remaining edge incident to v.

Suppose that S is the set of colored edges. Let F be a component of $G - S$ and assume that F is a cycle. Since we color vertices as soon as at least one incident edge is colored, all vertices of F are colored. If two adjacent vertices are colored by the same color we stop since there is no total 4-coloring extending the current partial total coloring. Assume that F is an odd cycle. Thus there is a vertex $u \in V(G)$ which is adjacent to vertices $v, w \in V(G)$ which are colored by different colors. In this case there is at most one possibility to extend the total coloring by coloring either $\{u, v\}$ or $\{u, w\}$, and therefore there is at most one possibility to color the edges of F. Assume that F is an even cycle. Then there are at most two possibilities to color the edges of F, and two possibilities may exist only if the vertices of the cycle are colored alternately by two colors.

The time analysis is similar to the one in the proof of Theorem 1. The number of possibilities to color the edges (using 4 colors) is $O^*(2^{5n/8})$ since the same recurrences apply. Furthermore there are at most two possibilities to color a vertex which contributes a factor of 2^n and implies the stated running time. □

3.3 Lower bounds for edge 3-colorings

Let us note that the enumeration algorithm of Subsection 3.1 actively uses the fact that the considered graphs are connected and have no vertices of degree one or two. Particularly, our upper bound for the number of edge 3-colorings does not apply to all graphs of maximum degree 3. For example, for an n-vertex path P_n, the number of edge 3-colorings is $3 \cdot 2^{n-2}$, and for the disjoint union of $\frac{n}{6}$ copies of $K_{3,3}$, the number of edge 3-colorings is $12^{n/6}$. Now we give lower bound for the number of edge 3-colorings of connected cubic graphs.

We consider a complete bipartite graph $K_{3,3}$. Let e_1 and e_2 be edges of this graph. We replace these edges by paths of length 3 with middle vertices a_1, a_2 and b_1, b_2 respectively. Denote the obtained graph by H. We call vertices a_1, a_2, b_1, b_2 *roots* of H. Let $n = 10r$ be a positive integer. We construct r copies of H denoted by $H_1 \ldots, H_r$, and denote by $a_1^{(i)}, a_2^{(i)}, b_1^{(i)}, b_2^{(i)}$ the roots of H_i for $1 \leq i \leq r$. Assume that $b_1^{(0)} = b_1^{(r)}$ and $b_1^{(0)} = b_1^{(r)}$. For $1 \leq i \leq r$, we add edges $\{b_1^{(i-1)}, a_1^{(i)}\}$ and $\{b_2^{(i-1)}, a_2^{(i)}\}$. Let us call the resulting connected cubic graph G. It is possible to prove the following proposition.

Proposition 1. *The connected cubic graph G has n vertices and at least $12^{n/10}$ different edge 3-colorings.* [5]

4 Dynamic programming counting algorithms

We establish dynamic programming algorithms (needing exponential space) to count the number of edge k-colorings and total k-colorings on graphs of bounded pathwidth. This allows the design of exact algorithms to count the edge and total k-colorings of graphs of bounded degree. It also implies a faster edge 3-coloring algorithm. Notice that since $\chi'(G) \geq \Delta(G)$ and $\tau(G) \geq \Delta(G)+1$, it is sufficient to consider our problems for graphs of bounded maximum degree: $\Delta(G) \leq k$ for edge k-colorings and $\Delta(G) \leq k-1$ for total k-colorings.

First we summarize a few fundamentals on treewidth and pathwidth.

A *tree decomposition* of a graph G is a pair (X,T) where T is a tree whose vertices we will call *nodes* and $X = (\{X_i \mid i \in V(T)\})$ is a collection of subsets of $V(G)$ (called *bags*) such that

1. $\bigcup_{i \in V(T)} X_i = V(G)$,
2. for each edge $\{v,w\} \in E(G)$, there is an $i \in V(T)$ such that $v,w \in X_i$, and
3. for each $v \in V(G)$ the set of nodes $\{i \mid v \in X_i\}$ forms a subtree of T.

The *width* of a tree decomposition $(\{X_i \mid i \in V(T)\}, T)$ equals $\max_{i \in V(T)}\{|X_i|-1\}$. The *treewidth* of a graph G, denoted $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G.

A tree decomposition (X,T) of a graph G with T being a path is called a *path decomposition* of G. The *pathwidth* of G is the minimum width over all path decompositions of G. The pathwidth is denoted by $\mathrm{pw}(G)$. For a path decomposition (X,P), we assume that the path P has nodes $1,\ldots,r$ in the given order. For $1 \leq i \leq r$, by G_i we denote the graph induced by the set $X_1 \cup \cdots \cup X_i$. It is well known that every path decomposition (X,P) can be easily converted (in linear time) to a *nice* path decomposition of same width (and with a linear size of P), such that nodes of P are of two types:

1. *Introduce* nodes i with $X_i = X_{i-1} \cup \{v\}$ for some vertex $v \in V(G)$.
2. *Forget* nodes i with $X_i = X_{i-1} \setminus \{v\}$ for some vertex $v \in V(G)$.

We assume here that $X_0 = \emptyset$. Nice path decompositions are used in the design of our dynamic programming algorithm.

Nowadays dynamic programming algorithms on path or tree decompositions are often used to establish exact exponential time algorithms (see e.g. [8, 9, 11]). We discuss this approach for for graphs of maximum degree $d \geq 3$. This approach relies on upper bounds for the pathwidth. Fomin and Høie [11] proved

[5] It was pointed to us by Artem Pyatkin that it is possible to improve this lower bound, if we consider the graph with $n = 2r$ vertices obtained by joining two cycles C_r by a perfect matching (vertices joined in the cyclical order). This graph has at least $\frac{3}{4} \cdot 2^{n/2}$ different edge 3-colorings.

that for any $\varepsilon > 0$, there exists an integer n_ε such that for every graph G with maximum vertex degree at most three and with $|V(G)| > n_\varepsilon$, $\mathrm{pw}(G) \leq (\frac{1}{6} + \varepsilon)|V(G)|$. It should be noted that the proof of this fact is constructive and a path decomposition of width at most $(\frac{1}{6}+\varepsilon)|V(G)|$ can be constructed in polynomial time. Fomin and Høie pointed out in [11] that such path decompositions can be used for constructing fast exact algorithm for the graphs of maximum degree three. They demonstrated it for the problems Maximum Independent Set, Max-Cut and Minimum Dominating Set. This technique was also used for the vertex 3- and 4-coloring problems in [7]. The best known upper bound is the following.

Proposition 2 ([8]). *For any $\varepsilon > 0$, there exists an integer n_ε such that for every graph G with $|V(G)| = n > n_\varepsilon$,*

$$\mathrm{pw}(G) \leq \frac{1}{6}n_3 + \frac{1}{3}n_4 + \frac{13}{30}n_5 + \frac{23}{45}n_6 + n_{\geq 7} + \epsilon n,$$

where n_i is the number of vertices of degree i in G for any $i \in \{3,4,5,6\}$ and $n_{\geq 7}$ is the number of vertices of degree at least 7. Moreover, a path decomposition of the corresponding width can be computed in polynomial time.

This bound can be combined with dynamic programming algorithms for edge k-coloring and total k-coloring on graphs of bounded pathwidth.

Theorem 3. *For an n-vertex graph with pathwidth at most p,*

1. *all edge k-colorings can be counted in time $O(\binom{k}{\lfloor k/2 \rfloor}^{p+1} \cdot k \cdot k! \cdot \log k \cdot n^2)$,*
2. *all total k-colorings can be counted in time $O(\left(k \cdot \binom{k-1}{\lfloor (k-1)/2 \rfloor}\right)^{p+1} \cdot k \cdot k! \cdot \log k \cdot n^2)$,*

if the path decomposition is given.

Proof. To describe the dynamic programming algorithms, we describe what we store in the tables corresponding to the bags of the path decomposition. We consider a nice path decomposition of a graph G with maximum degree at most k and pathwidth at most p with bags $X_1, \ldots, X_r$. For $1 \leq i \leq r$, we denote by G_i the subgraph of G induced by $X_1 \cup X_2 \cup \cdots X_i$. For $1 \leq i \leq r$ and $0 \leq j \leq k$, let $Z_i^{(j)} \subseteq X_i$ be the set of vertices in the bag X_i having degree j in G_i. Assume that $Z_i^{(j)} = \{z_1^{(j)}, \ldots, z_{p_j}^{(j)}\}$.

At first we consider edge colorings. The table of data for a bag X_i stores entries often called characteristics which contain collections of sets $\{S_1^{(j)}, \ldots, S_{p_j}^{(j)}\}$ for $1 \leq j \leq k$ and an integer σ such that

- $S_t^{(j)} \subseteq \{1, \ldots, k\}$ and $|S_t^{(j)}| = j$ for $1 \leq t \leq p_j$ and $1 \leq j \leq k$, and
- there are σ edge k-colorings of G_i such that edges of G_i incident with $z_t^{(j)}$ are colored by the colors from $S_t^{(j)}$ for $1 \leq t \leq p_j$ and $1 \leq j \leq k$.

The first claim of the proposition follows from the observation that the table contains at most

$$\prod_{j=1}^{k}\binom{k}{j}^{p_j} \leq \binom{k}{\lfloor k/2\rfloor}^{p+1}$$

entries. Constructions of the tables for introduce and forget nodes are straightforward, and we omit these descriptions here. It remains to notice that the number of all edge k-colorings of G equals to the sum of all integers σ over all entries of the table for the node r.

The proof of the second claim is similar. For a node i, we keep collections of pairs $\{(\alpha_1^{(j)}, S_1^{(j)}), \ldots, (\alpha_{p_i}^{(j)}, S_{p_j}^{(j)})\}$ for $1 \leq j \leq k-1$ and an integer σ such that

- $\alpha_t^{(j)} \in \{1, \ldots, k\}$,
- $S_t^{(j)} \subseteq \{1, \ldots, k\} \setminus \{\alpha_t^{(j)}\}$ and $|S_t^{(j)}| = j$ for $1 \leq t \leq p_j$ and $0 \leq j \leq k-1$, and
- there are σ total k-coloring of G_i such that $z_t^{(j)}$ is colored by $\alpha_t^{(j)}$ and edges of G_i incident with $z_t^{(j)}$ are colored by the colors from $S_t^{(j)}$ for $1 \leq t \leq p_j$ and $0 \leq j \leq k-1$.

It remains to note that the table contains at most

$$\prod_{j=0}^{k-1}\left(k\cdot\binom{k-1}{j}\right)^{p_j} \leq \left(k\cdot\binom{k-1}{\lfloor (k-1)/2\rfloor}\right)^{p+1}$$

entries.

This implies that the overall number of entries stored in tables of bags is $O(\binom{k}{\lfloor k/2\rfloor}^{p+1}\cdot n)$ in the first and $O(\left(k\cdot\binom{k-1}{\lfloor (k-1)/2\rfloor}\right)^{p+1}\cdot n)$ in the second algorithm.

The additional factors in the stated running times (though of little importance for our further applications in which $k > 0$ is a small integer) are discussed here for edge coloring only. When computing the entries for an insert node X_i obtained from a fixed entry of X_{i-1} with $v \in X_i \setminus X_{i-1}$, there are at most $k!$ possibilities to color the edges with endpoint v and another endpoint in X_{i-1}, and it takes time $O(k)$ to compute and verify the validity of an entry. Furthermore the algorithm stores in each entry the number of valid partial edge colorings in G_i. In a unit-cost RAM model the necessary arithmetic operations can be done in time $O(1)$. In a more realistic log-cost RAM model there is another factor $n \log k$ since the number of edge k-colorings in an n-vertex graph is at most k^n. □

The above dynamic programming algorithm is simpler and has a better running time for a small number of colors than the known dynamic programming algorithms for edge colorings on graphs of bounded treewidth [4, 22, 23].

Using our algorithms of Theorem 3 and upper bounds on the maximum pathwidth of graphs of a given maximum degree, we can obtain exact algorithms for counting edge k-colorings and total $(k+1)$-colorings on graphs of maximum degree k. Combined with the fact that graphs of maximum degree larger than k have neither edge k-colorings nor total $(k+1)$-colorings this implies algorithms for all graphs. To mention a few:

Theorem 4. *For any $\varepsilon > 0$,*

1. *all edge 3-colorings of a graph can be counted in time $O^*(3^{(1/6+\varepsilon)n})$,*
2. *all edge 4-colorings of a graph can be counted in time $O^*(6^{(1/3+\varepsilon)n})$,*
3. *all total 4-colorings of a graph can be counted in time $O^*(12^{(1/6+\varepsilon)n})$.*

Theorem 4 implies a $O^*(3^{n/6}) = O(1.2010^n)$ time exponential space algorithm to count the edge 3-colorings of graphs improving upon the $O(1.344^n)$ running time of Kowalik's polynomial space algorithm for the edge 3-coloring decision problem [15].

5 Conclusion

Results similar to those for edge and total k-colorings presented in previous sections can be obtained for $L(2,1)$-labelings of graphs. Due to space restrictions, we give only a short summary of our results.

An $L(2,1)$-*labeling* of a graph G of span k is a function $f\colon V(G) \to \{0,\ldots,k\}$ such that for any adjacent vertices u and v, $|f(u) - f(v)| \geq 2$, and for any two vertices u and v at distance two, $f(u) \neq f(v)$. Fiala et al. proved that it is NP-complete to decide whether a given graph has an $L(2,1)$-labeling of span k for any $k \geq 4$ [6]. Exact algorithms for the $L(2,1)$-labeling problem with span k are given by Král [16] (for the more general Channel Assignment problem) and Kratochvíl et al. [17]. For $k = 4$, an algorithm with running time $O(1.3161^n)$ was given in [17].

Since cubic graphs have no $L(2,1)$-labeling of span k for $k \leq 4$, we are interested in $L(2,1)$-labelings of span 5. One of the basic observations is the asymmetry within the colors. Coloring a vertex x by 0 or 5 makes only two colors unavailable for a neighbor of x. Coloring a vertex x by 1, 2, 3 or 4 makes three colors unavailable for each neighbor of x.

By a classical branching algorithm we can prove the following theorem.

Theorem 5. *All $L(2,1)$-labelings of span 5 for a given connected cubic graph with n vertices can be enumerated in time $O(1.8613^n)$, and the number $L(2,1)$-labelings of span 5 of any connected cubic graph is $O(1.8613^n)$.*

We can also show that the maximum number of $L(2,1)$-labelings of span 5 in a connected cubic graph is lower bounded by $2^{n/6}$.

Finally for any $\varepsilon > 0$, the number of $L(2,1)$-labelings of span 4 of n-vertex graphs can be counted in time $O^*(6^{(1/15+\varepsilon)n})$.

References

1. R. Beigel and D. Eppstein, *3-coloring in time $O(1.3289^n)$*, Journal of Algorithms, 54 (2005), pp. 168–204.
2. A. Björklund and T. Husfeldt, *Inclusion-exclusion algorithms for counting set partitions*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE, 2006, pp. 575–582.

3. A. Björklund, T. Husfeldt, P. Kaski and M. Koivisto *Narrow sieves for parameterized paths and packings*, arXiv:1007.1161v1, 2010.
4. H. L. Bodlaender, *Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees*, J. Algorithms 11 (1990), pp. 631–643.
5. D. Eppstein, *Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction*, Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2001, pp. 329–337.
6. J. Fiala, T. Kloks, and J. Kratochvíl, *Fixed-parameter complexity of lambda-labelings*, Discrete Applied Mathematics, 113 (2001), pp. 59–72.
7. F. V. Fomin, S. Gaspers, and S. Saurabh, *Improved exact algorithms for counting 3- and 4-colorings*, in COCOON 2007, vol. 4598 of Lecture Notes in Computer Science, Springer, 2007, pp. 65–74.
8. F. V. Fomin, S. Gaspers, and S. Saurabh, *On two techniques of combining branching and treewidth*, Algorithmica, 54 (2009), pp. 181–207.
9. F. Fomin, F. Grandoni, D. Kratsch, *Some new techniques in design and analysis of exact (exponential) algorithms*, Bulletin of the EATCS, 87 (2005), pp. 47–77.
10. F. V. Fomin, F. Grandoni, A. Pyatkin, and A. Stepanov, *Combinatorial bounds via Measure and Conquer: Bounding minimal dominating sets and applications*, ACM Transactions on Algorithms, Vol. 5, No. 1, Article 9, 2008.
11. F. V. Fomin and K. Høie, *Pathwidth of cubic graphs and exact algorithms*, Inf. Process. Lett., 97 (2006), pp. 191–196.
12. M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the Theory of NP-completeness*, Freeman, New York, 1979.
13. I. Holyer, *The NP-completeness of edge-coloring*, SIAM J. Comput., 10 (1981), pp. 718–720.
14. M. Koivisto, *An $O(2^n)$ Algorithm for graph coloring and other partitioning problems via inclusion-exclusion*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE, 2006, pp. 583–590.
15. L. Kowalik, *Improved edge-coloring with three colors*, Theoret. Comp. Sci., 410 (2009), pp. 3733–3742.
16. D. Král, *An exact algorithm for the channel assignment problem*, Discrete Applied Mathematics, 145 (2005), pp. 326–331.
17. J. Kratochvíl, D. Kratsch, and M. Liedloff, *Exact algorithms for $L(2,1)$-labeling of graphs*, in MFCS, L. Kucera and A. Kucera, eds., vol. 4708 of Lecture Notes in Computer Science, Springer, 2007, pp. 513–524.
18. J. W. Moon and L. Moser, *On cliques in graphs*, Israel J. Math., 3 (1965), pp. 23–28.
19. M. Rosenfeld, *On the total coloring of certain graphs*, Israel Journal of Mathematics, 9 (1971), pp. 396–402.
20. A. Sánchez-Arroyo, *Determining the total colouring number is NP-hard*, Discrete Mathematics, 78 (1989), pp. 315–319.
21. V. G. Vizing, *On an estimate of the chromatic class of a p-graph (in russian)*, Diskret. Anal., (1964), pp. 25–30.
22. X. Zhou and T. Nishizeki, *Optimal parallel algorithm for edge-coloring partial k-trees with bounded degrees*, Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, IEEE, 1994, pp. 167–174.
23. X. Zhou, S. Nakano, and T. Nishizeki, *Edge-coloring partial k-trees*, J. Algorithms 21 (1996), pp. 598–617.