Abstract

Defective coloring is a variant of the traditional vertex-coloring in which adjacent vertices are allowed to have the same color, as long as the induced monochromatic components have a certain structure. Due to its important applications, as for example in the bipartisation of graphs, this type of coloring has been extensively studied, mainly with respect to the size, degree, diameter, and acyclicity of the monochromatic components.

We focus on defective colorings with $\kappa$ colors in which the monochromatic components are acyclic and have small diameter, namely we consider $(edge, \kappa)$-colorings, in which the monochromatic components have diameter 1, and $(star, \kappa)$-colorings, in which they have diameter 2.

We prove that the $(edge, 3)$-coloring problem remains NP-complete even for graphs with maximum vertex-degree 6, hence answering an open question posed by Cowen et al. [10], and for planar graphs with maximum vertex-degree 7, and we prove that the $(star, 3)$-coloring problem is NP-complete even for planar graphs with bounded maximum vertex-degree. On the other hand, we give linear-time algorithms for testing the existence of $(edge, 2)$-colorings and of $(star, 2)$-colorings of partial 2-trees. Finally, we prove that outerpaths, a notable subclass of outerplanar graphs, always admit $(star, 2)$-colorings.

1 Introduction

Graph coloring is a fundamental problem in graph theory, which has been extensively studied over the years (see, e.g., [6] for an overview). Most of the
research in this area has been devoted to the *vertex-coloring problem* (or *coloring problem*, for short), which dates back to 1852 \[27\]. In its general form, the \(\kappa\)-coloring problem asks to label the vertices of a graph with a given number \(\kappa\) of colors, so that no two adjacent vertices have the same color. In other words, a \(\kappa\)-coloring of a graph partitions its vertices into \(\kappa\) color classes, each determining an independent set. A central result in this area is the so-called *four color theorem*, according to which every planar graph admits a \(\kappa\)-coloring with \(\kappa \leq 4\); see e.g. \[19\]. Note that the 3-coloring problem is NP-complete \[17\], even for graphs of maximum vertex-degree 4 \[11\].

Several variants of this problem have been proposed over the years; see, e.g., \[31\] for a survey. One of the most studied is the so-called *defective coloring*, independently introduced by Andrews and Jacobson \[2\], Harary and Jones \[21\], and Cowen et al. \[10\]. In this problem, edges between vertices of the same color class are allowed as long as the *monochromatic components*, which are the connected components of the subgraphs induced by vertices of the same color, maintain some special structure. In this respect, one can regard the classical vertex-coloring as a defective one in which every monochromatic component is an isolated vertex, given that every color class determines an independent set. Besides its theoretical interest, the defective coloring problem has interesting applications. For example, it can be regarded as the scheduling problem \[9\] where vertices represent tasks and edges represent conflicts between tasks in terms of shared resources. Here, a defect means tolerating some threshold of conflict: for example, each user running a task may find the maximum slowdown incurred for executing its task with one conflicting other task acceptable, and with more than one conflicting task unacceptable.

Over the years, models of defective colorings have been defined in terms of the maximum vertex-degree of each monochromatic component \[3, 8, 10, 26, 28\], of their size \[1, 15, 22\], of their acyclicity \[12, 14, 35\] (under the name of *tree-partition-width*), or of their diameter \[13\].

In this work we focus on the latter two aspects, namely we study defective colorings in which each monochromatic component is acyclic and has small diameter. In particular, we consider the cases in which the diameter is at most 1 (hence each component is either a vertex or an edge) or at most 2 (hence each component is a *star*, i.e. a tree with a central vertex connected to any number of leaves; see Figure 1d). We hence call the two corresponding problems *(edge, \(\kappa\))-coloring* and *(star, \(\kappa\))-coloring*, respectively. Figs. 1a-1c show a trade-off between number of colors and diameter of the monochromatic components. We present algorithmic and complexity results for these two problems when \(\kappa = 2\) and \(\kappa = 3\).

The model we study can be seen as a variant of the *bipartisation* of graphs, namely the problem of making a graph bipartite by removing a small number of elements (e.g. vertices or edges), which is a central graph problem with many applications \[20, 24\]. The bipartisation by removal of (a not-necessarily minimal number of) *non-adjacent* edges corresponds to the *(edge, 2)*-coloring problem. On the other hand, a *(star, 2)*-coloring also determines a bipartisation, in which *independent* stars are removed instead of vertices. Note that we do not ask
Figure 1: (a-c) Different colorings of the same graph: (a) a traditional 4-coloring, (b) an (edge, 3)-coloring (c) a (star, 2)-coloring; (d) a star with three leaves; its center has degree 3.

for the minimum number of removed edges or stars but for the existence of a solution.

We observe that the (edge, $\kappa$)-coloring setting also fits into the other models of defective colorings that have been studied so far. In fact, any monochromatic component with diameter at most 1 has also size at most 2 and vertex-degree at most 1. This observation implies that several results on the defective coloring with bounded maximum vertex-degree carry on to the (edge, $\kappa$)-coloring problem. More precisely, from a result of Lovász \[26\] it follows that all graphs of maximum vertex-degree 5 are (edge, 3)-colorable. However, Cowen et al. \[10\] prove that the (edge, 3)-coloring problem is already NP-complete for graphs of maximum vertex-degree 7 and for planar graphs \[10\] of maximum vertex-degree 10. In the same work, they prove that the (edge, 2)-coloring problem is NP-complete for graphs of maximum vertex-degree 4 and for planar graphs of maximum vertex-degree 5. Also, they prove that not all outerplanar graphs are (edge, 2)-colorable, while Eaton and Hull \[28\] prove that all triangle-free outerplanar graphs are.

On the other hand, (star, $\kappa$)-colorings have no direct relationship with the other defective coloring models, since their monochromatic components may have both the size and the maximum vertex-degree unbounded. Results on the complexity of the (star, 2)-coloring problem have been provided by Dorbec et al. \[13\], who proved that this problem is NP-complete even for planar graphs of maximum vertex-degree 4 and for triangle-free planar graphs.

Our contributions are:

- We prove that the (edge, 3)-coloring problem remains NP-complete for graphs with vertex-degree at most 6 (Section 2); this answers a question posed by Cowen et al. \[10\] in 1997 (recall that graphs of maximum vertex-degree 5 always admit such colorings \[26\]). We also show that this problem is NP-complete even for planar graphs of vertex-degree at most 7, which was already known only for planar graphs of vertex-degree at most 10 \[10\].

Also, we prove NP-completeness of the (star, 3)-colorings problem, even

---

1In \[10\] no explicit bound is given on the maximum vertex-degree; however, their proof can be adapted to work for planar graphs with maximum degree 10 by using planar graphs with maximum degree 4 in the reduction

2An outerplanar graph is a graph that admits a planar drawing in which all the vertices are on the same face
for (planar) graphs of bounded maximum vertex-degree, namely for graphs with vertex-degree at most 9 and for planar graphs with vertex-degree at most 16.

- We present efficient algorithms for testing the existence of (edge, 2)-colorings and (star, 2)-colorings of subclasses of planar graphs (Section 3). In particular, we describe linear-time algorithms for testing (edge, 2)-colorability and (star, 2)-colorability of partial 2-trees. We remark that partial 2-trees are a meaningful subclass of planar graphs that is of interest in computational complexity theory, since many NP-complete graph problems are solvable in linear time on these graphs (see, e.g., [34]).

We also recall that the partial \(k\)-trees are those graphs with treewidth at most \(k\), for any \(k \geq 1\). Intuitively, the treewidth \(\text{tw}\) is a parameter that measures how much a graph is similar to a tree, and it has applications in parameterized complexity (see, e.g., [16]). Since both types of colorability are expressible in the monadic second-order logic (MSO logic), they are decidable in linear time for graphs of bounded treewidth, as a consequence of Courcelle’s theorem [7]. However, the use of MSO logic and Courcelle’s theorem usually leads to algorithms having impractical running times, with high dependence on their parameters. This motivates the design of more efficient ad-hoc algorithms (see, e.g., [4, 25]).

- We provide a subclass of outerplanar graphs that is always (star, 2)-colorable, namely the class of outerpaths (Section 4). An outerpath is an outerplanar graph whose weak-dual is a path. Note that it is easy to construct an outerpath not admitting any (edge, 2)-coloring.

2 NP-completeness Results

In this section we study the computational complexity of the (edge, 3)-coloring and the (star, 3)-coloring problems.

As discussed above, problem (edge, 3)-coloring is NP-complete [10] for graphs of maximum vertex-degree 7, while graphs of maximum vertex-degree 5 always admit (edge, 3)-colorings [26]. We close this gap by proving in Theorem 1 that the problem remains NP-complete even for graphs of maximum vertex-degree 6. Then, in Theorem 2 we prove that the NP-completeness result extends to planar graphs of maximum vertex-degree 7. We leave as an open question the complexity of the problem for planar graphs of maximum vertex-degree 6.

For the (star, 3)-coloring problem we prove NP-completeness for graphs of maximum vertex-degree 9 and for planar graphs of maximum vertex-degree 16. While the (star, 2)-coloring problem was already known to be NP-complete, even for restricted graph classes [13], to the best of our knowledge these are the first NP-completeness results for the (star, 3)-coloring problem.

The weak-dual of a plane graph is the subgraph of its dual obtained by neglecting the face-vertex corresponding to its unbounded face.
From now on, given a defective coloring of a graph, we call colored edge an edge whose end-vertices have the same color.

**Theorem 1** Problem (edge, 3)-coloring is NP-complete for graphs of maximum vertex-degree 6.

Proof: Membership in NP is shown in [10]. To prove the NP-hardness, we employ a reduction from the Not-All-Equal 3-SAT (NAE3SAT) problem [30], p.187. An instance of NAE3SAT consists of a 3-CNF formula \( \phi \) with variables \( x_1, \ldots, x_n \) and clauses \( C_1, \ldots, C_m \). The task is to find a truth assignment satisfying \( \phi \) in which no clause has all its three literals equal in truth value (that is, not all are true). We show how to construct a graph \( G_\phi \) of maximum vertex-degree 6 admitting an (edge, 3)-coloring if and only if \( \phi \) is satisfiable.

Consider the graph of Fig.2a, which we denote by \( G_{4,5,5} \), as it contains one vertex \( u_1 \) of degree 4, two distinct vertices \( u_2 \) and \( u_3 \) of degree 5 (white in Fig.2a), and four vertices \( v_1, v_2, v_3, \) and \( v_4 \) (gray in Fig.2a) of degree 6, which form a \( K_4 \). Each of \( u_1, u_2, u_3 \) is connected to each of \( v_1, v_2, v_3, v_4, \) and there exists an edge \((u_2, u_3)\). We claim that in any (edge, 3)-coloring of \( G_{4,5,5} \) vertices \( u_1, u_2, \) and \( u_3 \) have the same color, even in the absence of edge \((u_2, u_3)\). We refer to this color as the color of \( G_{4,5,5} \). Suppose, for a contradiction, that \( u_1 \) and \( u_2 \) have different colors, say white and black. Since \( v_1, v_2, v_3, \) and \( v_4 \) form a \( K_4 \), at most two of them have the third color, say gray. Since \( u_1 \) (\( u_2 \)) is incident to all \( v_1, v_2, v_3, \) and \( v_4 \), at most one of them can be white (black). So, exactly two out of \( v_1, v_2, v_3, \) and \( v_4 \) are gray, one is white, and one is black. Further, each of them is incident to a colored edge. Since \( u_3 \) is adjacent to all of \( v_1, v_2, v_3, \) and \( v_4 \), one of them is incident to two colored edges, regardless of the color of vertex \( u_3 \); contradiction. A schematization of \( G_{4,5,5} \) is given in Fig.2b.

Now, consider the graph of Fig.2c denoted by \( G_{3,3,5,5} \), as it has two vertices \( s \) and \( t \) of degree 3, and two other vertices \( x \) and \( y \) of degree 5. Graph \( G_{3,3,5,5} \) contains five copies \( G_0, G_1, G_1', G_2, \) and \( G_2' \) of \( G_{4,5,5} \), connected by edges \( e_1, e_1', e_2, e_2', e_3, \) and \( e_3' \). We claim that in any (edge, 3)-coloring of \( G_{3,3,5,5} \) vertices \( s \) and \( t \) have the same color, say black, while one of \( x \) and \( y \) is white and the other one is gray. Namely, \( e_1, e_2, \) and \( e_3 \) guarantee that \( G_0, G_1, \) and \( G_2 \) have mutually different colors. Thus, \( x \) and \( y \) have different colors. Symmetrically, \( G_0, G_1', \) and \( G_2' \) have mutually different colors. Also, \( s \) and \( t \) are only incident to \( G_1, G_1', G_2, \) and \( G_2' \) (dotted edges in Fig.2c). Hence, both of them have the color of \( G_0 \), which is different from the ones of \( x \) and \( y \), completing the proof of the claim. We schematize \( G_{3,3,5,5} \) as in Fig.2d.

For \( k \geq 1 \), we form a **chain of length** \( k \), which contains \( 3k + 1 \) copies \( G_1, \ldots, G_{3k+1} \) of \( G_{3,3,5,5} \) connected to each other as follows (see Fig.2e). For \( i = 1, \ldots, 3k + 1 \), let \( s_i, t_i, x_i, y_i \) be the vertices of \( G_i \). Then, for \( i = 1, \ldots, 3k \), we introduce between \( G_i \) and \( G_{i+1} \) two vertices \( z_i \) and \( z_i' \) which form a \( K_4 \) with \( t_i \) and \( s_{i+1} \), as well as edges \((y_i, z_i)\) and \((z_i', x_{i+1})\) (dotted in Fig.2e). Assume that in \( G_i \) vertices \( s_i \) and \( t_i \) are black, \( x_i \) is gray, and \( y_i \) is white (see e.g. \( G_1 \) or \( G_4 \) in Fig.2e). Since \( t_i \) and \( s_{i+1} \) are incident to colored edges in \( G_i \) and \( G_{i+1} \) and are incident to each other due to edge \((t_i, s_{i+1})\), they have different
colors. Hence, \( z_i \) and \( z'_i \) have the third color. Since \( z_i \) is incident to \( y_i \), the color of \( z_i \) and \( z'_i \) is gray, and the one of \( s_{i+1} \) and \( t_{i+1} \) is white. Since \( z'_i \) is incident to \( x_{i+1} \), the color of \( x_{i+1} \) is black and the one of \( y_{i+1} \) is gray. So, the coloring of \( G_i \) uniquely determines the one of \( G_{i+1} \). Note that \( z_i \) and \( z_{i+3} \) have the same color, with \( 1 \leq i \leq 3k - 2 \), and that all vertices of the chain have degree 6, except for vertices \( z_i \) and \( z'_i \) (which have degree 4), and for vertices \( x_1 \) and \( y_{3k+1} \) (which have degree 5). We schematize a chain as in (Fig. 3a) (for \( k = 2 \)). Thus, the schematization of a chain \( C \) of length \( k \) is treated as a tripartite graph with partitions \( B[C] = \{ z_{3i+1}, z'_{3i+1}; 0 \leq i \leq k - 1 \} \), \( P[C] = \{ z_{3i+2}, z'_{3i+2}; 0 \leq i \leq k - 1 \} \) and \( N[C] = \{ z_{3i+3}, z'_{3i+3}; 0 \leq i \leq k - 1 \} \), each containing 2k vertices of degree 4 having the same color. The symbols \( B, P, \) and \( N \) stand for Black, Positive, and Negative, respectively.

Graph \( G_\phi \) contains a chain \( C \) of length \( \lfloor \frac{3n+2a}{4} \rfloor \), referred to as global chain (topmost chain in Fig. 3a). For each variable \( x_i \) of \( \phi \), graph \( G_\phi \) contains a chain \( C_{x_i} \) of length \( \lfloor \frac{n_i+1}{4} \rfloor \), where \( n_i \) is the number of occurrences of \( x_i \) in \( \phi \), \( 1 \leq i \leq m \) (variable-gadget; see Fig. 3a). For \( i = 1, \ldots, n \), we connect a vertex with degree 4 of partition \( B[C] \) to a vertex in \( P[C_{x_i}] \) and to a vertex in \( N[C_{x_i}] \) (solid gray edges in Fig. 3a). These connections guarantee that if partition \( B[C] \) is black, then neither \( P[C_{x_i}] \) nor \( N[C_{x_i}] \) is black and, hence, \( B[C_{x_i}] \) is black. Thus, \( P[C_{x_i}] \) and \( N[C_{x_i}] \) will act as the positive and negative partitions of chain \( C_{x_i} \).

For each clause \( C_i = (\lambda_j \lor \lambda_k \lor \lambda_\ell) \) of \( \phi \), \( 1 \leq i \leq m \), where \( \lambda_j \in \{ x_j, \neg x_j \}, \lambda_k \in \{ x_k, \neg x_k \}, \lambda_\ell \in \{ x_\ell, \neg x_\ell \} \) and \( j, k, \ell \in \{ 1, \ldots, n \} \), graph \( G_\phi \) contains a triplet of so-called clause-vertices that form a 3-cycle and so cannot have the same color (clause-gadget; see Fig. 3a). We connect each clause-vertex of clause \( C_i \) to a vertex of degree less than 6 of partition \( B[C] \) (dashed gray edges in Fig. 3a). These connections guarantee that if partition \( B[C] \) is colored black, then
no clause-vertex is also colored black. The length of the global chain guarantees that all connections can be made so that no vertex of partition $B[C]$ has degree larger than $6$.

If $\lambda_j$ is positive (negative), then we connect the clause-vertex corresponding to $\lambda_j$ in $G_\phi$ to a vertex of degree smaller than $6$ that belongs to the positive partition $P[C_{x_j}]$ (to the negative partition $N[C_{x_j}]$) of chain $C_{x_j}$. Similarly, we create connections for literals $\lambda_k$ and $\lambda_l$; see the solid-black edges leaving the triplets for clause $C_1$ and $C_2$ in Fig. 3a.

The length of $C_{x_i}$, for $i = 1, \ldots, n$ guarantees that all connections are accomplished so that no vertex of $P[C_{x_i}]$ and $N[C_{x_i}]$ has degree larger than $6$. Thus, $G_\phi$ has maximum vertex-degree $6$. Since $G_\phi$ is linear in the size of $\phi$, the construction can be done in $O(n + m)$ time.

We show that $G_\phi$ is (edge, $3$)-colorable if and only if $\phi$ is satisfiable. First assume that $\phi$ is satisfiable. We color partitions $P[C], N[C]$, and $B[C]$ of $C$ white, gray, and black, respectively. If $x_i$ is true (false), then we color $P[C_{x_i}]$ white (gray) and $N[C_{x_i}]$ gray (white), and $B[C_{x_i}]$ black. Hence, the tripartitions of $C_{x_i}$ are of different colors, as required by the construction. Further, if $x_i$ is true (false), then we color gray (white) all the clause-vertices of $G_\phi$ that correspond to positive literals of $x_i$ in $\phi$ and we color white (gray) those corresponding to negative literals. Thus, a clause-vertex of $G_\phi$ cannot have the same color as its two neighbors at the variable-gadget of $x_i$, with $1 \leq i \leq m$. Since in the truth assignment of $\phi$ no clause has all three literals true, no three clause-vertices belonging to the same clause have the same color.

Suppose that $G_\phi$ is (edge, $3$)-colorable and, w.l.o.g., that partition $B[C]$ of global chain $C$ is black. Hence, $P[C_{x_i}]$ and $N[C_{x_i}]$ are white or gray, $i = 1, \ldots, n$. If $P[C_{x_i}]$ is white, then we set $x_i = true$; otherwise, we set $x_i = false$. Assume, for a contradiction, that there is a clause of $\phi$ whose literals are all true or all false. By construction, the corresponding clause-vertices of $G_\phi$ have the same color, which is a contradiction since they form a 3-cycle in $G_\phi$. 

\[\square\]
We now consider planar graphs of bounded vertex-degree.

**Theorem 2** Problem (edge, 3)-coloring is NP-complete for planar graphs of maximum vertex-degree 7.

**Proof:** NP membership is shown in \[10\]. To prove NP-hardness, we employ a reduction from the 3-coloring problem, which is NP-complete even for planar graphs of maximum vertex-degree 4 \[11\]. Let \(G\) be a graph of maximum vertex-degree 4; we construct a planar graph \(G'\) of maximum vertex-degree 7 admitting an (edge, 3)-coloring if and only if \(G\) is 3-colorable.

Consider the (edge, 3)-colorable graph of Fig.3b, which we call attachment gadget, with a distinguished vertex \(v\), which we call pole-vertex. We claim that, in any (edge, 3)-coloring of an attachment gadget, the pole-vertex is incident to exactly one colored edge. To prove this, we show that vertices \(v_1, v_2,\) and \(v_3\) have different colors. Considering the symmetry of the vertices, w.l.o.g. assume, for a contradiction, that \(v_1\) and \(v_2\) have the same color, say gray. Then, each of vertices \(v_1, v_2, v_3, v_2, v_3, v_1, v_2, v_3, v_1, v_2, v_3, v_1, v_2, v_3, v_1, v_2, v_3\) must be colored either black or white, since each of them is incident to the gray edge \((v_1, v_2)\). However, the subgraph of the attachment gadget induced by these vertices has no (edge, 2)-coloring, as shown in \[8\], which is a contradiction.

Graph \(G'\) is obtained from \(G\) by attaching a copy \(G_u\) of the attachment gadget at each vertex \(u\) of \(G\), identifying \(u\) with the pole-vertex of \(G_u\). Since \(u\) has three neighbors in the gadget and at most four in \(G\), graph \(G'\) has vertex-degree at most 7. In addition, since the size of \(G'\) is linear in the one of \(G\), the reduction can be performed in linear time.

If \(G\) admits a 3-coloring, then \(G'\) admits a (edge, 3)-coloring in which each pole-vertex in \(G'\) has the same color as the corresponding vertex of \(G\), while the colors of the vertices in each attachment gadget are determined based on the color of its pole-vertex, as in Fig.3b. For the other direction it is sufficient to prove that, in any (edge, 3)-coloring of \(G'\), every two adjacent pole-vertices \(v\) and \(w\) of \(G'\) have different colors, as in this case a 3-coloring of \(G\) can be obtained by coloring each vertex as the corresponding pole-vertex in \(G'\). Namely, since both \(v\) and \(w\) are incident to a colored-edge in their attachment gadgets, edge \((v, w)\) cannot be colored.

We conclude the section presenting our results on problem (star, 3)-coloring.

**Theorem 3** Problem (star, 3)-coloring is NP-complete for graphs of maximum vertex-degree 9 and for planar graphs of maximum vertex-degree 16.

**Proof:** The problem clearly belongs to NP; a non-deterministic algorithm only needs to guess a color for each vertex of the graph and then check whether the graphs induced by each color-set are forests of stars, which can be done in linear time. To prove that the problem is NP-hard, we employ a reduction from the 3-coloring problem that is the same as the one of Theorem 2 up to the choice of the attachment gadget.

To prove the first part of the statement we use as attachment gadget the complete graph \(K_6\) on six vertices, which is (star, 3)-colorable (see Figure 4a).
The proof is based on the fact that in any (star, 3)-coloring of $K_6$ each vertex is incident to exactly one colored edge. This is due to the fact that, if three vertices had the same color, then they would form a cycle of colored edges, given that $K_6$ is a complete graph. Hence, if we attach a copy of $K_6$ to each vertex of the original instance $G$ of the 3-coloring problem, using any of its six vertices as the pole-vertex, then we can prove as in Theorem 2 that $G$ admits a 3-coloring if and only if the resulting graph (which has vertex-degree 9, since $G$ has vertex-degree 4) admits a (star, 3)-coloring.

For the second part of the theorem, we use as attachment gadget the planar graph of Figure 4b, with its topmost vertex being the pole-vertex. Since the pole-vertex has degree 12, the resulting graph is planar and has vertex-degree 16. Also, it is possible to prove that in any (star, 3)-coloring of the attachment gadget the pole-vertex is adjacent to at least a colored edge. This completes the proof of the theorem.

\[ \square \]

3  Efficient Testing Algorithms

In this section we give linear-time algorithms for the (edge, 2)-coloring and for the (star, 2)-coloring problems, restricted to certain subclasses of planar graphs. We recall that both these problems are NP-complete, even for planar graphs of bounded vertex-degree [10, 13].

In particular, we consider (edge, 2)-colorings of partial 2-trees (i.e., graphs with threewidth at most 2 [29]) in Theorem 3 and (star, 2)-colorings of the same class of graphs in Theorem 5.

Both algorithms are based on an efficient dynamic programming technique that exploits the \textit{SPQ-tree} data structure (formally recalled below). From a high-level perspective, an \textit{SPQ-tree} $T$ of a partial 2-tree $G$ is a tree-like data structure such that each node $\mu$ corresponds to a subgraph $G_\mu$ of $G$, and $G_\mu$ can be obtained with one of two possible compositions (a “series” or a “parallel” composition) of the subgraphs corresponding to the children of $\mu$ in $T$. The leaves of $T$ correspond to the edges of $G$. We traverse $T$ bottom-up and, for each node $\mu$, we compute a small set of representative solutions for $G_\mu$. These representative solutions for node $\mu$ are obtained by suitably merging the representative solutions associated with the children of $\mu$. Since the graphs associated with the children of

![Figure 4: (a) The complete graph on six vertices $K_6$. (b) The attachment gadget for the planar case.](image)
Basic definitions. We first introduce some basic definitions and tools. Series-parallel graphs are graphs with two special vertices, called their poles, inductively defined as follows. An edge \((s, t)\) is a series-parallel graph with poles \(s\) and \(t\). Let \(G_0, G_1, \ldots, G_k\) be a sequence of series-parallel graphs \((k \geq 1)\) and let \(s_i\) and \(t_i\) be the poles of \(G_i\) \((i = 0, \ldots, k)\). A series composition of \(G_0, G_1, \ldots, G_k\) is a series-parallel graph with poles \(s = s_0\) and \(t = t_k\), containing each \(G_i\) as a subgraph, and such that \(t_i\) and \(s_{i+1}\) have been identified \((i = 0, 1, \ldots, k - 1)\). A parallel composition of \(G_0, G_1, \ldots, G_k\) is a series-parallel graph with poles \(s = s_0 = s_1 = \cdots = s_k\) and \(t = t_0 = t_1 = \cdots = t_k\), which contain each \(G_i\) as a subgraph.

An SPQ-tree \(T\) of a series-parallel graph \(G\) is a tree, rooted at some node, representing the series (\(S\)-nodes) and parallel (\(P\)-nodes) compositions of \(G\), as well as the single edges of \(G\) (\(Q\)-nodes) \([18]\). Figures 5(a) and 5(b) show a series-parallel graph and a corresponding SPQ-tree. The pertinent graph \(G_\mu\) of a node \(\mu\) of \(T\), is the series-parallel subgraph of \(G\) such that the subtree of \(T\) rooted at \(\mu\) is an SPQ-tree of \(G_\mu\). We will denote by \(s_\mu\) and \(t_\mu\) the poles of \(G_\mu\).

A 2-tree is a graph obtained by starting from an edge and iteratively attaching a new vertex per time to two already adjacent vertices. A partial 2-tree is any subgraph of a 2-tree. It is known that the class of 2-trees coincides with the class of maximal series-parallel graphs \([23]\), i.e., the series-parallel graphs that cannot be augmented with any edge while remaining series-parallel. Also, a graph is a partial 2-tree if and only if all its biconnected components are series-parallel graphs \([3]\).
3.1 Testing (edge, 2)-colorability of Partial 2-Trees

We first describe an algorithm, called SPColorer, to test in linear time whether a biconnected series-parallel graph $G$ admits an (edge, 2)-coloring; the extension to the case in which $G$ is not biconnected will be presented later. The idea is to incrementally compute such a coloring (if any) by visiting bottom-up an SPQ-tree $T$ of $G$. Since the number of all feasible colorings for a given subgraph of $G$ may be exponential in the size of the subgraph, we define an equivalence relation on the set of these colorings, and keep track of only one representative solution in each equivalence class during the visit. The relation is based on the following definitions.

Let $G$ be a biconnected series-parallel graph and assume an (edge, 2)-coloring $C$ exists for $G$. Call white and black the two colors of $C$, and for each vertex $v$ of $G$ we denote by $N(v)$ the set of its neighbors. If $v$ is white (black) and $N(v)$ contains no white (black) vertices (i.e., $v$ has no incident colored edges), we say that $v$ is of type $W_0$ (type $B_0$). If $v$ is white (black) and $N(v)$ contains one white (black) vertex (i.e., $v$ has one incident colored edge), $v$ is of type $W_1$ (type $B_1$). Let $T$ be an SPQ-tree of $G$ rooted at an arbitrary $Q$-node $ρ$, and let $μ$ be a node of $T$. We say that two (edge, 2)-colorings $C_1$ and $C_2$ of the pertinent $G_μ$ of $μ$ are equivalent if pole $s_μ$ (pole $t_μ$) is of the same type in $C_1$ and $C_2$. Since we have four possible types for a vertex, this relation yields up to 16 equivalence classes of (edge, 2)-colorings for $G_μ$. Each of these classes is represented in the following as a pair $(X, Y)$, where $X$ and $Y$ are the types of $s_μ$ and of $t_μ$, respectively. Each element (i.e., each (edge, 2)-coloring) of an equivalence class is called a solution for $G_μ$. An equivalence class is feasible if it contains at least one solution.

It is immediate to see that if $C$ is an (edge, 2)-coloring of $G$ and if $C_1$ is the restriction of $C$ to $G_μ$ (hence, $C_1$ is a solution for $G_μ$), then replacing $C_1$ in $C$ with any equivalent solution $C_2$ for $G_μ$ yields a valid (edge, 2)-coloring of $G$. Exploiting this property, algorithm SPColorer visits $T$ bottom-up and, for each visited node $μ$ of $T$, efficiently computes a set, called the feasible bag of $μ$ and denoted by $b_μ$, containing one (representative) solution from each feasible equivalence class of (edge, 2)-colorings of $G_μ$. Hence, $b_μ$ contains at most 16 solutions. If $b_μ$ is empty, the algorithm halts and returns false. If the algorithm reaches the root $ρ$ of $T$ and $b_ρ$ is not empty, then it returns true and one of these solutions as a witness.

To compute $b_μ$, we will make use of the following operation. Let $ν_1$ and $ν_2$ be two children of $μ$, and suppose $μ$ is a $P$-node. Recall that $s_μ = s_{ν_1} = s_{ν_2}$, and $t_μ = t_{ν_1} = t_{ν_2}$. Let $C_1$ and $C_2$ be a solution for $G_{ν_1}$ and $G_{ν_2}$, respectively, such that $s_{ν_1}$ and $s_{ν_2}$ ($t_{ν_1}$ and $t_{ν_2}$) have the same color in $C_1$ and $C_2$. Let $G^*$ be the (series-parallel) subgraph of $G_μ$ obtained by the parallel composition of $G_{ν_1}$ and $G_{ν_2}$. The union operation of $C_1$ and $C_2$ returns a solution $C^*$ for $G^*$, obtained by merging $C_1$ and $C_2$, provided that neither $s_μ$ nor $t_μ$ becomes incident to two colored edges in $G^*$. An analogous operation is defined if $μ$ is an $S$-node; in this case, $t_1$ and $s_2$ (which are identified in the composition) must have the same color in $C_1$ and $C_2$ and, after the merging, $t_1 = s_2$ must not be incident to two
colored edges. In both cases, the union operation can be easily performed in constant time by representing each solution as an SPQ-tree of the corresponding pertinent graph and storing the types of the two poles.

We now prove, for each type of node \( \mu \), that the feasible bag \( b_\mu \) can be computed efficiently.

**Lemma 1** Let \( \mu \) be a non-root Q-node of \( T \). The feasible bag \( b_\mu \) of \( \mu \) can be computed in \( O(1) \) time.

**Proof:** Node \( \mu \) is a leaf of \( T \) and \( G_\mu \) is an edge between the poles \( s_\mu \) and \( t_\mu \). All and only the following equivalence classes are feasible for \( G_\mu \): \( (W_1, W_1), (B_1, B_1), (W_0, B_0), (B_0, W_0) \). Each of these classes has only one possible solution, thus \( b_\mu \) is unique and can be computed in constant time. \( \square \)

**Lemma 2** Let \( \mu \) be a P-node of \( T \). Let \( v_1, \ldots, v_k \), be the \( k \geq 2 \) children of \( \mu \) in \( T \), whose corresponding feasible bags have been already computed. A feasible bag \( b_\mu \) of \( \mu \) can be computed in \( O(k) \) time.

**Proof:** Recall that \( s_\mu = s_{v_1} = \cdots = s_{v_k} \) and \( t_\mu = t_{v_1} = \cdots = t_{v_k} \). We first determine whether an equivalence class is feasible for \( G_\mu \), and then compute a solution for it.

We start with the equivalence classes where both poles \( s_\mu \) and \( t_\mu \) are not incident to colored edges. Let \( (X, Y) \in \{ (W_0, W_0), (W_0, B_0), (B_0, W_0), (B_0, B_0) \} \) be any of these classes. Clearly, \( (X, Y) \) is feasible for \( G_\mu \) if and only if it is feasible for each \( G_{v_i} (i = 1, \ldots, k) \). Also, if \( (X, Y) \) is feasible for \( G_{v_i} (i = 1, \ldots, k) \), then the union operation on the corresponding solutions provides a solution for \( G_\mu \) in class \( (X, Y) \).

Consider now the equivalence classes in which there is only one pole (not both) incident to a colored edge. Assume that \( s_\mu \) is the pole incident to a colored edge (the other case is symmetric). It is easy to see that \( (X, Y) \), with \( X \in \{ W_1, B_1 \} \) and \( Y \in \{ W_0, B_0 \} \), is feasible for \( G_\mu \) if and only if there exists one bag \( b_{v_i} \) such that bag \( b_{v_i} \) contains a solution in class \( (X, Y) \) and every bag different from \( b_{v_i} \) contains a solution in \( (W_0, Y) \), if \( X = W_1 \), or in \( (B_0, Y) \), if \( X = B_1 \). Again, if the condition is satisfied, the union operation on the corresponding solutions provides a solution for \( G_\mu \) in class \( (X, Y) \).

Finally, consider the equivalence classes with both poles incident to a colored edge. Consider the class \( (W_1, W_1) \): analogous arguments hold for the others. Class \( (W_1, W_1) \) is feasible for \( G_\mu \) if and only if there exist either one bag \( b_{v_i} \), with \( 1 \leq i \leq k \), such that (i) bag \( b_{v_i} \) contains a solution in class \( (W_1, W_1) \) and (ii) every bag different from \( b_{v_i} \) contains a solution in \( (W_0, W_0) \), or two bags \( b_{v_i} \) and \( b_{v_j} \), with \( 1 \leq i, j \leq k \), such that (i) bag \( b_{v_i} \) contains a solution in class \( (W_1, W_0) \), (ii) bag \( b_{v_j} \) contains a solution in class \( (W_0, W_1) \), and (iii) every bag different from \( b_{v_i} \) and \( b_{v_j} \) contains a solution in \( (W_0, W_0) \). If one of the two conditions holds, the union operation on the corresponding solutions gives a solution for \( G_\mu \) in \( (W_1, W_1) \).
Since for each of the 16 classes there are $k$ bags to check (each containing at most 16 representative solutions), and since the union operation takes constant time, bag $b_\mu$ is computed in $O(k)$ time.

**Lemma 3** Let $\mu$ be an $S$-node of $T$. Let $\nu_1, \ldots, \nu_k$, be the $k \geq 2$ children of $\mu$ in $T$, whose corresponding feasible bags have been already computed. A feasible bag $b_\mu$ of $\mu$ can be computed in $O(k)$ time.

**Proof:** Recall that $s_{\nu} = s_{\nu_1}, t_{\nu} = s_{\nu_2}, \ldots, t_{\nu_{k-1}} = s_{\nu_k}, t_{\nu_k} = t_{\mu}$. We determine whether a class is feasible for $G_\mu$ through $k - 1$ intermediate steps. For the illustration, consider the class $(W_0, W_0)$ (the argument is analogous for the other classes). Let $G_i$ denote the graph resulting from the series composition of $G_{\nu_1}, G_{\nu_2}, \ldots, G_{\nu_k}$.

At step $i$, for $1 \leq i \leq k - 1$, consider the graphs $G_i$ and $G_{\nu_{i+1}}$. Let $b_i$ be a bag containing one solution for each class $(W_0, Y)$ that is feasible for $G_i$ (where $Y$ can be any type). Such a bag is computed in the previous step if $i > 1$, or it coincides with $b_{i-1}$ if $i = 1$. Then, we compute a bag $b_{i+1}$ containing one solution for each class that is feasible for $G_{i+1}$, as follows. A class $(W_0, Z)$ (where $Z$ can be any type) is feasible for $G_{i+1}$ if there exist a class $(W_0, Y)$ that is feasible for $G_i$ and a class $(X, Z)$ that is feasible for $G_{\nu_{i+1}}$, such that either $X = W_0$ and $Y \in \{W_0, W_1\}$, or $Y = W_0$ and $X \in \{W_0, W_1\}$, or $X = B_0$ and $Y \in \{B_0, B_1\}$, or $Y = B_0$ and $X \in \{B_0, B_1\}$, that is, $X$ and $Y$ are such that $t_{\nu_i}$ and $s_{\nu_{i+1}}$ have the same color and at most one of them is incident to a colored edge. If $(W_0, Z)$ is decided as feasible, the union operation on the corresponding solutions gives a solution for $G_i$ in $(W_0, Z)$.

Finally, since $G_k$ corresponds to $G_\mu$, then $(W_0, W_0)$ is feasible for $G_\mu$ if and only if $b_k$ contains a representative solution in $(W_0, W_0)$. Computing $b_k$ for each class takes $O(k)$ time, and this computation is done 16 times. Hence, $b_\mu$ is computed in $O(k)$ time.

**Lemma 4** Let $\rho$ be the root of $T$. Let $\xi$ be the (only) child of $\rho$ in $T$, whose corresponding feasible bag has been computed. A feasible bag $b_\rho$ of $\rho$ can be computed in $O(1)$ time.

**Proof:** Consider each class $(X, Y) \in b_\xi$. If $X \in \{W_0, W_1\}$ and $Y \in \{B_0, B_1\}$, or vice versa, then add $(X, Y)$ to $b_\rho$. Otherwise, if $X = Y = W_0$, then add $(W_1, W_1)$ to $b_\rho$, while if $X = Y = B_0$, then add $(B_1, B_1)$ to $b_\rho$. In all the other cases, do not add any class to $b_\rho$.

**Lemma 5** Let $G$ be an $n$-vertex biconnected series-parallel graph. There exists an $O(n)$-time algorithm that decides whether $G$ admits an (edge, 2)-coloring.

**Extension to non-biconnected graphs.** We now extend the result of Lemma 5 to every partial 2-tree $G$. Let $B$ be the set of blocks (i.e., biconnected components)
of $G$. As already pointed out, each block in $B$ is a biconnected series-parallel graph. Let $C$ be the set of cutvertices of $G$. Construct a tree $T$ with vertex set $B \cup C$ in which the edges are defined as follows: $c \in C$ is adjacent to $b \in B$ if and only if the block $b$ contains $c$. Tree $T$ is called the black-cutvertex tree of $G$, or simply the BC-tree of $G$. Furthermore, observe that the union operation on two solutions can be naturally extended to the case when the two corresponding subgraphs $G_{v_1}$ and $G_{v_2}$ share only one (cut) vertex.

**Theorem 4** Let $G$ be an $n$-vertex partial 2-tree. There exists an $O(n)$-time algorithm that decides whether $G$ admits an (edge, 2)-coloring.

**Proof:** If $G$ is biconnected, then the statement follows by Lemma 5. Otherwise, let $T$ be the BC-tree of $G$, rooted at an arbitrary cutvertex. We visit $T$ bottom-up and apply SPColorer to each block of $G$. In order to obtain a consistent and valid coloring for the cutvertices, we extend SPColorer so to handle additional inter-block constraints, as described below for the different types of nodes of $T$.

Let $b$ be a leaf of $T$ (associated with a block of $G$) and let $c_b$ be the cutvertex corresponding to the parent of $b$ in $T$. Root the SPQ-tree $T_b$ of $b$ at a Q-node having $c_b$ as a pole, and apply SPColorer to $b$. If SPColorer returns false, then the algorithm returns false. If SPColorer returns true, let the block bag $B_b$ of $b$ coincide with the feasible bag associated with the root node of $T_b$.

Let $c$ be a cutvertex of $T$ and let $b_1, b_2, \ldots, b_k$ be the $k \geq 1$ blocks that are children of $c$ in $T$. Let the cutvertex bag $B_c$ of $c$ be a set of solutions computed as follows. If all the block bags $B_{b_i}$, for $1 \leq i \leq k$, contain at least one solution where $c$ is of type $W_0$ (resp., $B_0$), then apply the union operation on these solutions and add it to $B_c$. If all the block bags $B_{b_i}$, for $1 \leq i \leq k$, contain at least one solution where $c$ is of type $W_0$ (resp., $B_0$), except for one block bag which contains a solution where $c$ is of type $W_1$ (resp., $B_1$) then apply the union operation on these solutions and add it to $B_c$. Clearly, $B_c$ contains at most four solutions. If $B_c$ is empty, then the algorithm halts and returns false. If $c$ is the root of $T$ and $B_c$ contains at least one solution $C$, then the algorithm returns true and $C$ as a witness.

Let $b$ be a block of $G$ that is not a leaf of $T$, and let $c_b$ be the cutvertex corresponding to the parent of $b$ in $T$. Let $c_1, \ldots, c_k$ be the $k \geq 1$ cutvertices that correspond to the children of $b$ in $T$. Root the SPQ-tree $T_b$ of $b$ at a Q-node having $c_b$ as a pole. We apply SPColorer to $b$ with the following modification. Every time a node $\mu$ of $T_b$ is visited such that one of the two poles is a cutvertex $c_i$ ($1 \leq i \leq k$), we need to ensure that $c_i$ receives a valid color with respect to all the other blocks attached to $c_i$, as follows.

Assume that $c_i$ coincides with pole $s_\mu$, the other case is symmetric. For each class $(X, Y)$ such that the feasible bag $b_\mu$ contains a solution in this class, we perform the following operation. Suppose that $X = W_0$ (that $X = B_0$). Remove the solution corresponding to $(X, Y)$ from $b_\mu$; then, for each $Z \in \{W_0, W_1\}$ (for each $Z \in \{B_0, B_1\}$), if the cutvertex bag $B_{c_i}$ contains a solution in which $c_i$ is of type $Z$, then add to $b_\mu$ a solution for class $(Z, Y)$, obtained by applying the union between the solution in which $c_i$ is of type $Z$ and the one corresponding to $(X, Y)$.
Suppose that \( X = W_1 \) (that \( X = B_1 \)). Remove the solution corresponding to \((X, Y)\) from \( b_\mu\); then, if \( B_{c_i} \) contains a solution in which \( c_i \) is of type \( W_0 \), then add to \( b_\mu \) a solution for class \((X, Y)\), again obtained by applying the union between the corresponding solutions. At the end of this procedure there may exist two solutions that belong to the same class; we pick one arbitrarily. Again, if \( \text{SPColorer} \) returns false, then a solution does not exist for \( G \) as well. If \( \text{SPColorer} \) returns true, then let the block bag \( B_b \) of \( b \) be the feasible bag of the root of \( T_b \).

The time complexity of the algorithm is \( O(n) \), as for each block \( b \) with \( n_b \) vertices we spend \( O(n_b) \) time to compute its block bag by Lemma 3.1 (including the operations at cutvertices); for each cutvertex \( c \) with \( n_c \) children in \( T \) we spend \( O(n_c) \) time to compute its cutvertex bag; and \( \sum_{\forall b \in B} n_b + \sum_{\forall c \in C} n_c = O(n). \)

### 3.2 Testing (star, 2)-colorability of Partial 2-Trees

We now turn our attention to the (star, 2)-coloring problem by extending the test for partial 2-trees provided in Section 3.1 to this problem.

Let \( T \) be an \( SPQ \)-tree of \( G \) rooted at an arbitrary \( Q \)-node \( \rho \), and let \( \mu \) be a node of \( T \). We maintain the same definition of equivalence between two colorings of \( C_1 \) and \( C_2 \) of \( G_\mu \) as in the (edge, 2)-coloring case, with the only difference that the set of possible types of the poles has to be larger in this case, in order to encompass all the possible configurations that may arise in a (star, 2)-coloring. Namely a pole of \( \mu \), say \( s_\mu \), in a (star, 2)-coloring can be of the following types:

(i) \( W_0 \): pole \( s_\mu \) is white and no vertex in \( N(s_\mu) \) is white (the monochromatic component containing \( s_\mu \) is an isolated vertex)

(ii) \( W_1 \): pole \( s_\mu \) is white, there exists exactly one white vertex \( w \) in \( N(s_\mu) \), and \( N(w) \) contains at least one white vertex different from \( s_\mu \) (the monochromatic component containing \( s_\mu \) is a star with center \( w \), and \( s_\mu \) is one of its leaves)

(iii) \( W_2 \): pole \( s_\mu \) is white and either there exists more than one white vertex in \( N(s_\mu) \) or there exists exactly one white vertex \( w \) in \( N(s_\mu) \), which is not \( t_\mu \) and has no white neighbor different from \( s_\mu \) (the monochromatic component containing \( s_\mu \) is a star and \( s_\mu \) is its center)

(iv) \( W_3 \): pole \( s_\mu \) is white and the only white vertex in \( N(s_\mu) \) is \( t_\mu \), which has no white neighbor other than \( s_\mu \) (the monochromatic component containing \( s_\mu \) only consists of edge \((s_\mu, t_\mu)\), and it is still undecided which of them is the center).

Types \( B_0, \ldots, B_3 \) are defined analogously, with color black instead of white.

Note that there exist eight possible types for a vertex; however, a pole is of type either \( W_3 \) or \( B_3 \) if and only if the other pole is of the same type. This implies that the total number of equivalence classes of (edge, 2)-colorings for \( G_\mu \) is 38. Recall that an equivalence class of a (star, 2)-coloring is represented by a pair \((X, Y)\), where \( X \) and \( Y \) are the types of \( s_\mu \) and \( t_\mu \), respectively. In the following, a pair of the form \((X, \cdot)\) will be used to denote an equivalence class in which \( s_\mu \) is of type \( X \), while \( t_\mu \) can be of any type. Pair \((\cdot, Y)\) is defined analogously.

Based on the property that the number of equivalence classes is still bounded
by a constant, we provide a linear-time algorithm to test the existence of a "edge, 2"-coloring of a partial 2-tree that works along the same lines as algorithm SPColorer. In the following we show how to compute in constant time the feasible bag $b_\mu$ of a node $\mu \in T$ starting from the feasible bags of the children of $\mu$.

**Lemma 6** Let $\mu$ be a non-root $Q$-node of $T$. The feasible bag $b_\mu$ of $\mu$ can be computed in $O(1)$ time.

**Proof:** Node $\mu$ is a leaf of $T$ and $G_\mu$ is an edge between the poles $s_\mu$ and $t_\mu$. Only the following equivalence classes are feasible for $G_\mu$: $(W_0, B_0)$, $(B_0, W_0)$, $(W_3, W_3)$, $(B_3, B_3)$. Each of these classes has only one possible solution, thus $b_\mu$ is unique and can be computed in constant time.

**Lemma 7** Let $\mu$ be a $P$-node of $T$. Let $\nu_1, \ldots, \nu_k$, be the $k \geq 2$ children of $\mu$ in $T$, whose corresponding feasible bags have been already computed. A feasible bag $b_\mu$ of $\mu$ can be computed in $O(k)$ time.

**Proof:** Recall that $s_\mu = s_{\nu_1} = \cdots = s_{\nu_k}$ and $t_\mu = t_{\nu_1} = \cdots = t_{\nu_k}$. We first determine whether an equivalence class is feasible for $G_\mu$, and then compute a solution for it.

In the following, we consider a pole of $\mu$, say $s_\mu$, and for each possible type $T_0, \ldots, T_3$, with $T \in \{W, B\}$, we discuss which are the conditions on the classes contained in the feasible bags of $\nu_1, \ldots, \nu_k$ that have to be satisfied in order for $s_\mu$ to be of that type. Testing whether a certain equivalence class $(X, Y)$ is feasible for $G_\mu$ can be done by combining the conditions that have to be satisfied in order for $s_\mu$ to be of type $X$ and $t_\mu$ to be of type $Y$, in the same way as in the last part of the proof of Lemma 2.

- Pole $s_\mu$ can be of type $T_0$ (that is, it is not incident to any colored edge) if and only if each bag $b_{\nu_i}$, with $1 \leq i \leq k$, contains a solution in class $(T_0, \cdot)$.
- Pole $s_\mu$ can be of type $T_1$ (that is, it is a leaf of a colored star) if and only if at least one of the following conditions is satisfied:
  - there exists a child $\nu_h$ of $\mu$ such that bag $b_{\nu_h}$ contains a solution in a class $(T_1, \cdot)$ and each bag $b_{\nu_i}$, with $1 \leq i \neq h \leq k$, contains a solution in a class $(T_0, \cdot)$; or
  - there exist two children $\nu_h$ and $\nu_z$ of $\mu$ such that bag $b_{\nu_h}$ contains a solution in a class $(T_3, T_3)$, bag $b_{\nu_z}$ contains a solution in a class $(T_0, T_2)$, and each bag $b_{\nu_i}$, with $1 \leq i \neq h, z \leq k$, contains a solution in a class either $(T_0, T_0)$ or $(T_0, T_2)$.
- Pole $s_\mu$ can be of type $T_2$ (that is, it is the center of a colored star) if and only if at least one of the following conditions is satisfied:
  - there exists a child $\nu_h$ of $\mu$ such that bag $b_{\nu_h}$ contains a solution in class $(T_2, \cdot)$ and each bag $b_{\nu_i}$, with $1 \leq i \neq h \leq k$, contains a solution in a class either $(T_0, \cdot)$ or $(T_2, \cdot)$; or
there exist two children νₜ and ν₂ of µ such that bag bₜ contains a solution in a class (T₃, T₃), bag b₂ contains a solution in class (T₂, T₀), and each bag bᵢ, with 1 ≤ i ≠ h, z ≤ k, contains a solution in a class either (T₀, T₀) or (T₂, T₀).

- Pole sₜ can be of type T₃ (that is, it is still undecided whether it is a leaf or a center of a colored star) if and only if there exists a child νₜ of µ such that bag bₜ contains a solution in a class (T₃, T₃) and each bag bᵢ, with 1 ≤ i ≠ h ≤ k, contains a solution in class (T₀, T₀).

Once an equivalence class (X, Y) has been positively tested by checking the corresponding conditions to hold at the same time for both the poles, the union operation on the solutions for the children of µ provides a solution for Gµ in class (X, Y).

Since for each of the 38 classes there are k bags to check (each containing at most 38 representative solutions), and since the union operation takes constant time, bag bₚ is computed in O(k) time.

**Lemma 8** Let µ be an S-node of T. Let ν₁, ..., νₖ, be the k ≥ 2 children of µ in T, whose corresponding feasible bags have been already computed. A feasible bag bₚ of µ can be computed in O(k) time.

**Proof:** Recall that sₚ = sᵣ₁, tᵣ₁ = tᵣ₂, ..., tᵣₖ₋₁ = sᵣₖ, tᵣₖ = tₚ. We first determine whether an equivalence class is feasible for Gµ, and then compute a solution for it.

For each equivalence class (X, Y) we test whether it is feasible for Gµ by selecting a class (Xᵢ, Yᵢ) in the feasible bag of each child νᵢ, with i = 1, ..., k, such that: (A) X = X₁. (B) Y = Yᵦ. (C) For any two consecutive children µᵢ and µᵢ₊₁, we have that tᵢ and sᵢ₊₁ have the same color and, if Yᵢ = W₁ then Xᵢ₊₁ = W₀, if Yᵢ = B₁ then Xᵢ₊₁ = B₀, if Xᵢ₊₁ = W₁ then Yᵢ = W₀, and if Xᵢ₊₁ = B₁ then Yᵢ = B₀. (D) For any three consecutive children µᵢ₋₁, µᵢ, and µᵢ₊₁, we have that tᵢ₋₁ and sᵢ, tᵢ and sᵢ₊₁ have the same color, and, if Xᵢ = Yᵢ = W₃, then at least one of Yᵢ₋₁ and Xᵢ₊₁ is W₀, while the other one is W₀ or W₂ or W₃; analogously, if Xᵢ = Yᵢ = B₃, then at least one of Yᵢ₋₁ and Xᵢ₊₁ is B₀, while the other one is B₀ or B₂ or B₃.

Note that the first two conditions ensure that the coloring belongs to (X, Y), while the other two ensure that it is a valid (star, 2)-coloring. The test can be performed in O(k) time using a technique similar to the one used in Lemma 2. Repeating the test for all the 38 classes yields a total O(k) running time to compute bₚ.

**Lemma 9** Let ρ be the root of T. Let ξ be the (only) child of ρ in T, whose corresponding feasible bag has been computed. A feasible bag bₜ of ρ can be computed in O(1) time.

**Proof:** Let sₜ = sₜ and tₜ = tₜ be the poles of ξ. Consider any class (X, Y) ∈ bₜ. Note that X, Y ≠ W₃, B₃, since Gₜ does not contain an edge between the
poles. If \( X \in \{W_0, W_1, W_2\} \) and \( Y \in \{B_0, B_1, B_2\} \), or vice versa, then add \((X, Y)\) to \( b_\mu \). Otherwise, suppose that \( X, Y \in \{W_0, W_1, W_2\} \), the case in which \( X, Y \in \{B_0, B_1, B_2\} \) being analogous. If \( X = Y = W_0 \), then add \((W_3, W_3)\) to \( b_\mu \). If \( X = W_0 \) and \( Y = W_2 \), then add \((W_1, W_2)\) to \( b_\mu \). If \( X = W_2 \) and \( Y = W_0 \), then add \((W_2, W_1)\) to \( b_\mu \). If \( X = W_0 \) and \( Y = W_2 \), then add \((W_1, W_2)\) to \( b_\mu \). In all the other cases, do not add any class to \( b_\mu \).

\( \Box \)

Lemmas 6–9 and the fact that an SPQ-tree \( T \) of \( G \) has \( O(n) \) testing algorithm for biconnected series-parallel graphs. In order to extend the algorithm to every (non-biconnected) partial 2-tree, we can apply the same procedure as in Theorem 4 with the following modifications.

The first modification is in the computation of the cutvertex bag \( B_c \) of a cut-vertex \( c \). Namely, if all the block bags \( B_{(a,b)} \), for \( 1 \leq i \leq k \), contain at least one solution where \( c \) is of type \( W_0 \) (resp., \( B_0 \)), then apply the union operation on these solutions and add it to \( B_c \). If all the block bags \( B_{(a,b)} \), for \( 1 \leq i \leq k \), contain at least one solution where \( c \) is of type \( W_0 \) (resp., \( B_0 \)), except for one block bag which contains a solution where \( c \) is of type \( W_1 \) (resp., \( B_1 \)) then apply the union operation on these solutions and add it to \( B_c \). Finally, if all the block bags \( B_{(a,b)} \), for \( 1 \leq i \leq k \), contain at least one solution where \( c \) is of type either \( W_0 \), or \( W_2 \), or \( W_3 \) (resp., \( B_0 \), or \( B_2 \), or \( B_3 \)), with at least one of them being different from \( W_0 \) (from \( B_0 \)), then apply the union operation on these solutions and add it to \( B_c \). Clearly, \( B_c \) contains at most eight solutions.

The other modification is on the operations that have to be performed when a node \( \mu \) of \( T_b \) is visited in which one of the two poles is a cutvertex \( c_i \) (\( 1 \leq i \leq k \)). Assume w.l.o.g. that \( c_i \) coincides with pole \( s_\mu \). For each class \((X, Y)\) such that the feasible bag \( b_\mu \) contains a solution in this class, we perform the following operation.

Suppose that \( X = W_0 \) (that \( X = B_0 \)). Remove the solution corresponding to \((X, Y)\) from \( b_\mu \); then, for each \( Z \in \{W_0, W_1, W_2\} \) (for each \( Z \in \{B_0, B_1, B_2\} \)), if the cutvertex bag \( B_{c_i} \) contains a solution in which \( c_i \) is of type \( Z \), then add to \( b_\mu \) a solution for class \((Z, Y)\), obtained by applying the union between the solution in which \( c_i \) is of type \( Z \) and the one corresponding to \((X, Y)\). Suppose that \( X = W_1 \) (that \( X = B_1 \)). If \( B_{c_i} \) does not contain any solution in which \( c_i \) is of type \( W_0 \) (of type \( B_0 \)), then remove the solution corresponding to \((X, Y)\) from \( b_\mu \). Suppose that \( X = W_2 \) (that \( X = B_2 \)). If \( B_{c_i} \) does not contain any solution in which \( c_i \) is of type either \( W_0 \) or \( W_2 \) (of type either \( B_0 \) or \( B_2 \)), then remove the solution corresponding to \((X, Y)\) from \( b_\mu \). Suppose that \( X = W_3 \) (that \( X = B_3 \)). Remove the solution corresponding to \((X, Y)\) from \( b_\mu \); then, if \( B_{c_i} \) contains a solution in which \( c_i \) is of type \( W_0 \) (of type \( B_0 \)), then add a solution in \((X, Y)\) to \( b_\mu \), while if \( B_{c_i} \) contains a solution in which \( c_i \) is of type \( W_2 \) (of type \( B_2 \)), then add a solution in \((W_2, W_1)\) (in \((B_2, B_1)\)) to \( b_\mu \). In both cases, the solutions are obtained by applying the union operation. At the end of this procedure there may exist two solutions that belong to the same class; we pick one arbitrarily. All the rest of the algorithm works as for SPColorer.

We summarize the result in the following.
Theorem 5 Let \( G \) be an \( n \)-vertex partial 2-tree. There exists an \( O(n) \)-time algorithm that decides whether \( G \) admits a (star, 2)-coloring.

4 Outerpaths

In this section we aim at determining notable classes of graphs that always admit a (star, 2)-coloring. In particular, we consider outerplanar graphs, since this class is known to have good properties with respect to coloring; in fact, every outerplanar graph admits a 3-coloring \([32]\), and every triangle-free outerplanar graph admits an (edge, 2)-coloring. On the other hand, it is known that not all outerplanar graphs admit an (edge, 2)-coloring \([10]\).

We hence ask whether allowing stars instead of edges as monochromatic components can help, namely whether every outerplanar graph admits a (star, 2)-coloring. We answer this question in the negative, by providing in Lemma 10 an example of a small outerplanar graph that is not (star, 2)-colorable. On the other hand, we prove in Theorem 6 that there exists a notable subclass of outerplanar graphs, called outerpaths, for which this property holds.

Lemma 10 There exist outerplanar graphs that are not (star, 2)-colorable.

Proof: We prove that the outerplanar graph of Figure 6a is not (star, 2)-colorable. In particular, we show that in any 2-coloring of this graph there exists a monochromatic path of four vertices. Assume w.l.o.g. that vertex \( u \) has color gray. Then, at least two vertices out of \( u_1, \ldots, u_8 \) are gray, as otherwise there would be a path of four white vertices. Hence, \( u \) is the center of a gray star.

Next, we observe that either \( u_2 \) is white or the path \( u_2, \ldots, u_8 \) must consist of only white vertices. Similarly, we observe that either \( u_3 \) is white or the path \( u_3, \ldots, u_8 \) must consist of only white vertices. If both \( u_2 \) and \( u_3 \) are white, then either one of paths \( u_2, \ldots, u_8 \) and \( u_3, \ldots, u_8 \) consists only of gray vertices, or there exists a path from one of \( u_2, \ldots, u_8 \) via \( u_2 \) and \( u_3 \) to one of \( u_3, \ldots, u_8 \), that consists only of white vertices. Clearly, all aforementioned cases lead to a monochromatic path of four vertices.

While not all outerplanar graphs admit a (star, 2)-coloring, as we observed in Lemma 10 we prove in Theorem 6 that every outerpath admits a coloring of this type, by giving a linear-time constructive algorithm. Note that the example provided in Lemma 10 is “almost” an outerpath, meaning that the weak-dual of this graph is “almost” a path, given that it contains only degree-1 and degree-2 vertices, except for one specific vertex, which has degree 3 (see the face of the graph in Figure 6a highlighted in gray).

Before describing the (star, 2)-coloring algorithm, we observe that it is easy to construct an outerpath not admitting any (edge, 2)-coloring. In fact, consider a graph \( G \) composed of a path \( P \) with six vertices and of an additional vertex \( v \) connected to all the vertices of \( P \). Since the weak dual of \( G \) is a path with five vertices, \( G \) is an outerpath. Also, in any (edge, 2)-coloring of \( G \) there exists at most one vertex of \( P \) with the same color as \( v \); hence, at least three consecutive
vertices of $P$ must have the same color, and hence $G$ does not admit any (edge, 2)-coloring.

We now describe the (star, 2)-coloring algorithm. Let $G$ be an outerpath. We assume that $G$ is inner-triangulated. This is not a loss of generality, as any (star, 2)-coloring of a triangulated outerpath induces a (star, 2)-coloring of any of its subgraphs. We first give some definitions; refer to Figure 6b. We call spine vertices the vertices $v_1, v_2, \ldots, v_m$ with degree at least 4 in $G$. We consider an additional spine vertex $v_{m+1}$, which is the (unique) neighbor of $v_m$ along the cycle delimiting the outer face that is not adjacent to $v_{m-1}$. Note that the spine vertices of $G$ induce a path, that we call spine of $G$. The fan $f_i$ of a spine vertex $v_i$ consists of the set of neighbors of $v_i$ in $G$, except for $v_{i-1}$ and for those following and preceding $v_i$ along the cycle delimiting the outer face; note that $|f_i| \geq 1$ for each $i = 1, \ldots, m$, while $|f_{m+1}| = 0$. For each $i = 1, \ldots, m+1$, we denote by $G_i$ the subgraph of $G$ induced by the spine vertices $v_1, \ldots, v_i$ and by the fans $f_1, \ldots, f_{i-1}$. Note that $G_{m+1} = G$. We denote by $c_i$ the color assigned to spine vertex $v_i$, and by $c(G_i)$ a coloring of graph $G_i$. Recall that an edge of $G$ is called colored if its two endpoints have the same color.

**Theorem 6** Every outerpath admits a (star, 2)-coloring, which can be computed in linear time.

**Proof:** Let $G$ be an outerpath with spine $v_1, \ldots, v_k$. We describe an algorithm to compute a (star, 2)-coloring of $G$. At each step $i = 1, \ldots, k$ of the algorithm we consider the spine edge $(v_{i-1}, v_i)$, assuming that a (star, 2)-coloring of $G_i$ has already been computed satisfying one of the following conditions (see Figure 7):

$Q_0$: The only colored vertex is $v_1$.

---

Note that the spine of $G$ coincides with the spine of the caterpillar obtained from the outerpath $G$ by removing all the edges incident to its outer face, neglecting the additional spine vertex $v_{m+1}$.

Fan $f_i$ contains all the leaves of the caterpillar incident to $v_i$, plus the following spine vertex $v_{i+1}$.
Figure 7: Schematization of the algorithm. Each node represents the (unique) condition satisfied by $G_i$ at some step $0 \leq i \leq k$. An edge label $0, 1, e, o$ represents the fact that the cardinality of a fan $f_i$ is 0, 1, even $\neq 0$, or odd $\neq 1$. If the label contains two characters, the second one describes the cardinality of $f_{i+1}$. An edge between $Q_j$ and $Q_h$ with label $x \in \{1, e, o\}$ (with label $xy$, where $y \in \{0, 1, e, o\}$) represents the fact that, if $G_i$ satisfies condition $Q_j$ and $|f_i| = x$ (resp. $|f_i| = x$ and $|f_{i+1}| = y$), then $f_i$ is colored so that $G_{i+1}$ satisfies $Q_h$.

$Q_1$: $c_i \neq c_{i-1}$, vertex $v_{i-1}$ is the center of a star with color $c_{i-1}$, and no colored edge is incident to $v_i$;

$Q_2$: $c_i = c_{i-1}$, and no colored edge other than $(v_{i-1}, v_i)$ is incident to $v_{i-1}$ or $v_i$;

$Q_3$: $c_i \neq c_{i-1}$, vertex $v_{i-1}$ is a leaf of a star with color $c_{i-1}$, and no colored edge is incident to $v_i$;

$Q_4$: $c_i \neq c_{i-1}$, vertex $v_{i-1}$ is the center of a star with color $c_{i-1}$, and vertex $v_i$ is the center of a star with color $c_i$; further, $i < k$ and $|f_i| > 1$;

$Q_5$: $c_i = c_{i-1}$, vertex $v_{i-1}$ is the center of a star with color $c_{i-1}$, and no colored edge other than $(v_{i-1}, v_i)$ is incident to $v_i$; further, $i < k$ and $|f_i| = 1$.

Next, we color the vertices in $f_i$ in such a way that $c(G_{i+1})$ is a (star, 2)-coloring satisfying one of the conditions; refer to Figure 7 for a schematization of the case analysis. In the first step of the algorithm, we assign an arbitrary color to $v_1$, and hence $c(G_1)$ satisfies $Q_0$. For $i = 1, \ldots, k$ we color $f_i$ depending on the condition satisfied by $c(G_i)$.

**Coloring $c(G_i)$ satisfies $Q_0$:** Independently of the cardinality of $f_i$, we color its vertices with alternating colors so that $c_{i+1} \neq c_i$. In this way, the only possible colored edges are incident to $v_i$ and not to $v_{i+1}$. So, $c(G_{i+1})$ satisfies condition $Q_1$.

**Coloring $c(G_i)$ satisfies $Q_1$:** In this case we distinguish the following subcases, based on the cardinality of $f_i$.

---

21
Figure 8: Graph $G_{i+1}$ after coloring $f_i$ when $c(G_i)$ satisfies: $Q_1$ and (a) $|f_i| = 1$ or (b) $|f_i| > 1$; $Q_2$ and (c) $|f_i| = 0$, or (d) $|f_i| = e$ and (e) $|f_{i+1}| = 1$, or (f) $f_{i+1} > 1$. Shaded regions represent $G_i$. Bold edges connect vertices with the same color, while spine edges are dashed.

- If $|f_i| = 0$, we have that $i = k$ and hence $G_k = G$. It follows that $c(G_k)$ is a (star, 2)-coloring of $G$.

- If $|f_i| = 1$ (that is, $f_i$ contains only $v_{i+1}$; see Figure 8a), we set $c_{i+1} = c_i$. Since the only neighbor of $v_{i+1}$ in $G_{i+1}$ different from $v_i$ is $v_{i-1}$, whose color is $c_{i-1} \neq c_i$, and since $v_i$ has no neighbor with color $c_i$ other than $v_{i+1}$, by condition $Q_1$, coloring $c(G_{i+1})$ is a (star, 2)-coloring satisfying condition $Q_2$.

- If $|f_i| > 1$ (see Figure 8b), we color the vertices in $f_i$ with alternating colors so that $c_{i+1} \neq c_i$. This implies that every colored edge of $G_{i+1}$ not belonging to $G_i$ is incident either to $v_i$, if its color is $c_i$, or to $v_{i-1}$, if its color is $c_{i-1}$; the latter case only happens if $|f_i|$ is odd. Thus, $v_i$ (resp. $v_{i-1}$) is the center of a star of color $c_i$ (resp. $c_{i-1}$) in $G_{i+1}$. Since $v_i$ has no neighbor with color $c_i$ in $G_i$, while $v_{i-1}$ is a center also in $G_i$, coloring $c(G_{i+1})$ is a (star, 2)-coloring. Finally, since $v_{i+1}$ has no neighbors with color $c_{i+1} \neq c_i$, by construction, $c(G_{i+1})$ satisfies condition $Q_1$.

**Coloring $c(G_i)$ satisfies $Q_2$:** We again distinguish subcases based on $|f_i|$.

- If $|f_i| = 0$, we have that $i = k$ and hence $c(G_k)$ is a (star, 2)-coloring of $G = G_k$.

- If $|f_i|$ is odd, including the case $|f_i| = 1$ (see Figure 8c), we color the vertices of $f_i$ with alternating colors in such a way that $c_{i+1} \neq c_i$. By construction, $c(G_{i+1})$ is a (star, 2)-coloring satisfying condition $Q_1$.

- If $|f_i|$ is even and different from 0, instead, we have to consider the cardinality of $f_{i+1}$ in order to decide the coloring of $f_i$. We distinguish three subcases:

  - $|f_{i+1}| = 0$: Note that in this case $i = k$ holds (see Figure 8d). We color the vertices of $f_i$ with alternating colors so that $c_{i+1} = c_i$. Note that the unique neighbor of $v_{i-1}$ in $f_i$ has color different from $c_{i-1}$, since $|f_i|$ is even. Hence, all the new colored edges are incident to $v_i$, which implies that $c(G_{i+1})$ is a (star, 2)-coloring satisfying condition $Q_2$. 

  - $|f_{i+1}| > 0$: We need to decide the coloring of $f_i$. If $|f_{i+1}| = 1$, we set $c_{i+1} = c_i$. Otherwise, we color $f_i$ with alternating colors so that $c_{i+1} \neq c_i$. This implies that every colored edge of $G_{i+1}$ not belonging to $G_i$ is incident either to $v_i$, if its color is $c_i$, or to $v_{i-1}$, if its color is $c_{i-1}$; the latter case only happens if $|f_i|$ is odd. Thus, $v_i$ (resp. $v_{i-1}$) is the center of a star of color $c_i$ (resp. $c_{i-1}$) in $G_{i+1}$. Since $v_i$ has no neighbor with color $c_i$ in $G_i$, while $v_{i-1}$ is a center also in $G_i$, coloring $c(G_{i+1})$ is a (star, 2)-coloring. Finally, since $v_{i+1}$ has no neighbors with color $c_{i+1} \neq c_i$, by construction, $c(G_{i+1})$ satisfies condition $Q_1$.

  - $|f_{i+1}| > 0$: We need to decide the coloring of $f_i$. If $|f_{i+1}| = 1$, we set $c_{i+1} = c_i$. Otherwise, we color $f_i$ with alternating colors so that $c_{i+1} \neq c_i$. This implies that every colored edge of $G_{i+1}$ not belonging to $G_i$ is incident either to $v_i$, if its color is $c_i$, or to $v_{i-1}$, if its color is $c_{i-1}$; the latter case only happens if $|f_i|$ is odd. Thus, $v_i$ (resp. $v_{i-1}$) is the center of a star of color $c_i$ (resp. $c_{i-1}$) in $G_{i+1}$. Since $v_i$ has no neighbor with color $c_i$ in $G_i$, while $v_{i-1}$ is a center also in $G_i$, coloring $c(G_{i+1})$ is a (star, 2)-coloring. Finally, since $v_{i+1}$ has no neighbors with color $c_{i+1} \neq c_i$, by construction, $c(G_{i+1})$ satisfies condition $Q_1$. 

22
Figure 9: Graph $G_{i+1}$ after coloring $f_i$ when $c(G_i)$ satisfies: $Q_3$ and $|f_i| = 1$, or $|f_i| = e$; $Q_4$ or $Q_5$ (d). Shaded regions represent $G_i$. Bold edges connect vertices with the same color, while spine edges are dashed.

$|f_{i+1}| = 1$: Note that $i < k$ and $f_{i+1}$ only contains $v_{i+2}$ (see Figure 8e). We color the vertices of $f_i$ with alternating colors so that $c_{i+1} = c_i$. Since (i) all the new colored edges are incident to $v_i$, (ii) $v_i$ and $v_{i-1}$ have no neighbor with their same color in $G_i$ (apart from each other), (iii) $c_{i+1} = c_i$, and (iv) $i < k$, we have that $c(G_{i+1})$ is a (star, 2)-coloring satisfying condition $Q_5$.

$|f_{i+1}| > 1$: Note that $i < k$ (see Figure 8f). Independently of whether $|f_{i+1}|$ is even or odd, we color the vertices of $f_i$ so that $c_{i+1} \neq c_i$, the unique neighbor of $v_{i+1}$ different from $v_i$ has also color $c_{i+1}$, and all the other vertices have alternating colors. Since each new colored edge is incident to either $v_i$ or $v_{i+1}$, since $c_{i+1} \neq c_i$, and since $i < k$, coloring $c(G_{i+1})$ is a (star, 2)-coloring satisfying condition $Q_4$.

**Coloring $c(G_i)$ satisfies $Q_3$:**

- If $|f_i| = 0$, we have that $i = k$ and hence $c(G_k)$ is a (star, 2)-coloring of $G = G_k$.

- If $|f_i| = 1$ (that is, $f_i$ contains only $v_{i+1}$; see Figure 9a), we set $c_{i+1} = c_i$. As in the analogous case in which $c(G_i)$ satisfies condition $Q_1$, we can prove that $c(G_{i+1})$ is a (star, 2)-coloring which satisfies condition $Q_2$.

- If $|f_i|$ is even and different from 0 (see Figure 9b), we color the vertices of $f_i$ with alternating colors in such a way that $c_{i+1} \neq c_i$. By construction, $c(G_{i+1})$ is a (star, 2)-coloring which satisfies condition $Q_1$.

- If $|f_i|$ is odd and different from 1, we again consider the cardinality of $f_{i+1}$ in order to decide the coloring of $f_i$. For the four possible classes of values of $|f_{i+1}|$, the coloring strategy and the condition satisfied by the resulting coloring $c(G_{i+1})$ are the same as for the analogous case in which $c(G_i)$ satisfies $Q_2$ and $|f_i|$ is even.

**Coloring $c(G_i)$ satisfies $Q_4$:** Note that $|f_i| > 0$, given that $i < k$, and $|f_i| \neq 1$, by condition $Q_4$. Independently of whether $|f_i|$ is even or odd (see Figure 9c), we color the vertices in $f_i$ with alternating colors so that $c_{i+1} \neq c_i$. In this way, the only possible colored edges are incident to $v_{i-1}$ and to $v_i$, which are
both centers of a star already in $G_i$, and not to $v_{i+1}$. Hence, $c(G_{i+1})$ is a \((\text{star, 2})\)-coloring satisfying condition $Q_1$.

**Coloring $c(G_i)$ satisfies $Q_5$:** Note that $|f_i| = 1$, by condition $Q_5$ (that is, $f_i$ only contains $v_{i+1}$; see Figure 9d). We set $c_{i+1} = c_i$; clearly, $c(G_{i+1})$ is a \((\text{star, 2})\)-coloring satisfying condition $Q_3$.

Observe that the running time of the algorithm is linear in the number of vertices of $G$. In fact, at each step $i = 1, \ldots, k$, the condition $Q_j$ satisfied by $c(G_i)$ and the cardinalities of $f_i$ and $f_{i+1}$ are known (the cardinality of all the fans can be precomputed in advance), and the coloring strategy to obtain $c(G_{i+1})$ and the condition satisfied by this coloring are uniquely determined by these information in constant time.

\[ \square \]

5 Conclusions

In this work we presented algorithmic and complexity results for the \((\text{edge, } \kappa)\)-coloring and the \((\text{star, } \kappa)\)-coloring problems, with $\kappa \in \{2, 3\}$, which ask for the existence of defective $\kappa$-colorings of given graphs in which every monochromatic component is an edge and is a star, respectively. There exist several open questions raised by our work.

- What is the complexity of the \((\text{edge, 3})\)-coloring problem for planar graphs of maximum vertex-degree 6?
- What is the complexity of the \((\text{star, 3})\)-coloring problem for (planar) graphs of maximum vertex-degree 6, 7, 8?
- Is it possible to extend our linear-time testing algorithms from Section 3 to efficiently test \((\text{edge, 2})\)-colorability and \((\text{star, 2})\)-colorability of every graph of bounded treewidth? We recall that Courcelle’s theorem \[7\] already provides a linear-time testing algorithm for these graphs, which is however unusable in practice due to the large constant factors.
- Are there other classes of graphs, besides the outerpaths, that are always \((\text{star, 2})\)-colorable, e.g., outerplanar graphs with maximum vertex-degree 4?
- One possible way to expand the class of graphs that admit defective colorings is to allow larger values on the diameter of the monochromatic components.
- The variant of the problem that minimizes the total number of defective edges is of interest; it is related to the max-cut and the $k$-partization problems.

24
Acknowledgement: This work started at the Graph and Network Visualization (GNV) session of IISA ’15. Research supported in part by the MIUR project AMANDA, prot. 2012C4E3KT_001, by NSERC Canada, and by DFG grant Ka812/17-1.

We thanks the anonymous reviewers of this paper for their valuable suggestions.

References


