Partitions of Graphs into Trees

Therese Biedl and Franz J. Brandenburg

[1] School of Computer Science, University of Waterloo, N2L3G1, Canada
biedl@uwaterloo.ca
[2] Lehrstuhl für Informatik, Universität Passau, 94030 Passau, Germany
brandenb@informatik.uni-passau.de

Abstract. In this paper, we study the *k-tree partition problem* which is a partition of the set of edges of a graph into k edge-disjoint trees. This problem occurs at several places with applications e.g. in network reliability and graph theory. In graph drawing there is the still unbeaten $(n-2) \times (n-2)$ area planar straight line drawing of maximal planar graphs using Schnyder's realizers [15], which are a 3-tree partition of the inner edges. Maximal planar bipartite graphs have a 2-tree partition, as shown by Ringel [14]. Here we give a different proof of this result with a linear time algorithm. The algorithm makes use of a new ordering which is of interest of its own. Then we establish the NP-hardness of the k-tree partition problem for general graphs and $k \geq 2$. This parallels NP-hard partition problems for the vertices [3], but it contrasts the efficient computation of partitions into forests (also known as arboricity) by matroid techniques [7].

1 Introduction

A *k-tree partition* of a graph $G = (V, E)$ is the partition of the set of edges E into k disjoint subsets which each induce a tree. Alternatively, the edges of G are colored by k colors and each color induces a tree. The trees are not necessarily spanning trees. The *k-tree partition problem* for a graph G and an integer k is whether or not G has a k-tree partition.

A relaxed version without connectivity is the *arboricity* $a(G)$ of a graph G, which is a partition of the edges of G into at most $a(G)$ forests. A well-known theorem by Nash-Williams states that a graph has arboricity c if and only if every non-trivial subgraph H has at most $c(|V(H)| - 1)$ edges [11,12]. In particular, this implies that every planar graph has arboricity at most 3, and every planar bipartite graph has arboricity 2. In fact, the two forests of the arboricity-decomposition must be "almost" trees: either one is a spanning tree and the other has $n-3$ edges, or both have $n-2$ edges. Ringel [14] proved that in fact, any maximal planar bipartite graph can be split into two trees, both with $n-2$ edges.

In this paper, we study algorithmic aspects of splitting a maximal planar bipartite graph into two trees. The proof by Ringel [14] is algorithmic in nature, but not particularly fast; it can be implemented in quadratic time. We give a different proof to show that every maximal planar bipartite graph can be

M. Kaufmann and D. Wagner (Eds.): GD 2006, LNCS 4372, pp. 430–439, 2007.
© Springer-Verlag Berlin Heidelberg 2007

split into two trees; the resulting algorithm is quite simple and can easily be implemented in linear time. As a side-effect, we develop a special vertex ordering for maximal planar bipartite graphs, which may be of interest of its own.

For general graphs with n vertices and m edges the arboricity can efficiently be computed by matroid techniques [7]. Here the relaxation to forests is crucial. We show the NP-hardness of the k-tree partition problem for every $k \geq 2$. For $k = 2$ this is proved by a reduction from the Not-All-Equal-3SAT problem, and for $k \geq 3$ there is a reduction from the k-coloring problem. These NP-hardness results complement common and extended versions of partition and coloring problems [2,3,8], which however are defined for the vertices.

Partitioning graphs into trees is also used in graph drawing. For example, the Schnyder realizers [15] are a partition of the inner edges of a maximal planar graph into three trees. They still yield the best known area bounds for straight-line grid drawings of maximal planar graphs.

In this paper, first we study maximal planar bipartite graphs and how they split into two trees, and then we show the NP-hardness results for the general case.

2 Maximal Planar Bipartite Graphs

Let $G = (V, E)$ be a maximal planar bipartite (mpb) graph. Thus, G has a vertex partition $V = W \cup B$ into *white vertices* W and *black vertices* B such that each edge connects a white vertex with a black vertex. Furthermore, G can be drawn in the plane without crossings such that every face has exactly four incident edges. It is well-known that G has $2n - 4$ edges (where $n = |V|$) and is bi-connected (i. e., cannot be disconnected by removing one vertex.)

2.1 A Vertex Ordering for mpb Graphs

In this section, we present a vertex ordering for maximal planar bipartite graphs, which we will then use in the next section to obtain a split of an mpb graph into two trees.

Theorem 1. *Let G be a maximal planar bipartite graph with a fixed planar embedding and a fixed outer-face. Then there exists a vertex ordering $v_1, \ldots, v_n$ of G such that*

- *v_1 and v_n are the two black vertices on the outer-face.*
- *For all $i > 1$, vertex v_i is on the outer-face of the graph induced by $v_1, \ldots, v_i$.*
- *Every white vertex v_i has exactly one predecessor, i. e., neighbor with a smaller number.*
- *Every black vertex v_i, $i > 1$ has at least two predecessors.*

We will call such a vertex ordering an *mpb-ordering*. See Figure 1. To prove Theorem 1, we need an auxiliary graph. Let E_B be the *black diagonals*, i. e., for every face f in G (which has exactly two black vertices since G is maximal

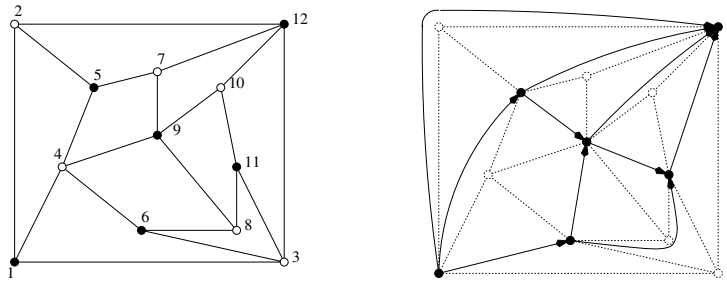

Fig. 1. An mpb-ordering of G, and the graph G_B (solid) with a bipolar orientation

planar bipartite), add an edge between the two black vertices on f to E_B. Let G_B be (B, E_B), i. e., take the black vertices and black diagonals only. See also Figure 1.

One can show that G_B is bi-connected [4]. Therefore, we can compute an *st-numbering* of G_B, i. e., an ordering $b_1, \ldots, b_l$ of the black vertices such that for any $1 < j < l$, vertex b_j has at least one predecessor and at least one successor, i. e., neighbor with a larger index [10]. Moreover, we can choose which vertices should be b_1 and b_l, and we can compute this order in linear time [6]. We choose here b_1 and b_l to be the two black vertices on the outer-face of G.

From this *st*-numbering, we can obtain a *bipolar orientation*, i. e., an acyclic orientation of the edges of G_B such that there is only one *source* (vertex without incoming edge) and only one *sink* (vertex without outgoing edge), simply by directing every edge from the lower-indexed to the higher-indexed vertex.

Let $G^+ = (V, E \cup E_B)$ be the graph resulting from G by adding the black diagonals. We now extend the bipolar orientation of G_B into one of G^+ as follows. For every white vertex w, let b_i be the neighbor of w (in G) that has the *smallest* index among the neighbors of w. Orient the edge (b_i, w) from b_i to w, and all other edges incident to w away from w. Clearly this orientation is bipolar: every white vertex must have degree 2 (by maximality), and has exactly one incoming edge by definition, and hence at least one outgoing edge. Furthermore, the orientation is acyclic since any directed path encounters increasingly larger indices in its black vertices.

From this bipolar orientation, we can recover a vertex ordering of all vertices of G, simply by computing a topological order in the acyclic graph. Let $v_1, \ldots, v_n$ be the resulting order; one can easily verify that it satisfies all conditions of Theorem 1.

Note that all steps of computing the mpb-ordering can easily be implemented in linear time.

The mbp-ordering is not a canonical ordering [5,9]: The black vertices act similar to the vertices of a canonical ordering, but white vertices have only one predecessor and violate the 2-connectivity property of a canonical ordering. E.g. the mbp-ordering of the graph in Figure 1 is not a canonical ordering.

2.2 Splitting into Two Trees

Now assume that we are given an mpb-ordering $v_1, \ldots, v_n$. We now show how to obtain a decomposition into two trees from it. We have two simple rules (see also Figure 2):

- For any white vertex, label the (unique) incoming edge with 1.
- For any black vertex $v_i \neq v_1$, label the leftmost incoming edge with 1 and the rightmost incoming edge with 2. Here, "leftmost" and "rightmost" are taken with respect to the planar embedding; recall that v_i is in the outer-face of the graph induced by $v_1, \ldots, v_{i-1}$, hence we can sort its incoming edges by the order in which these neighbors appear on the outer-face.
  All other (if any) incoming edges of v_i are called "middle incoming" and labeled with 2 as well. However, we will reverse the orientation of these edges, to make it easier to argue why the resulting structures must be trees.

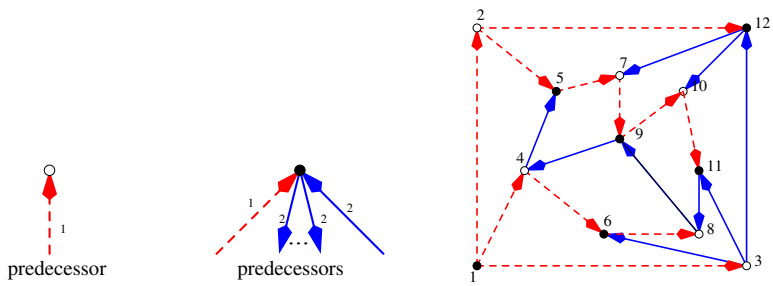

Fig. 2. Splitting the graph into two trees

From now on, let the 1-*edges* be the edges labeled 1, and the 2-*edges* be the edges labeled 2. We also use *1-path*, *1-cycle* and so on to mean a path/cycle of 1-edges. It is very easy to see that the 1-edges form a tree.

Lemma 1. *The* 1*-edges form a spanning tree.*

Proof. Since every vertex except v_1 has exactly one incoming 1-edge, there are $n-1$ 1-edges. v_1 has outgoing 1-edges, so the 1-edges span all vertices of the graph. No 1-edge had its orientation reversed, so the 1-edges form a directed acyclic graph. It is well-known that such a graph is a spanning tree.

Now we come to the significantly harder part of proving that the 2-edges form a tree. We first need observations about the order of edges around each vertex; see also Figure 3.

Claim. v_1 has only outgoing 1-edges. For any other black vertex, the incident edges are clockwise in the planar embedding as follows:

- One incoming 1-edge.
- Some number (possibly none) of outgoing 1-edges.

– One incoming 2-edge.
– Some number (possibly none) of outgoing 2-edges.

Proof. This follows directly from the way labels and directions were assigned to vertices, plus the fact that every successor of a black vertex is a white vertex and hence contributes an outgoing 1-edge.

Claim. Let w_1, w_2 be the white vertices on the outer-face. For every white vertex, the incident edges are ordered clockwise in the planar embedding as follows:

– One incoming 1-edge.
– Some number (possibly none) of outgoing 2-edges.
– One incoming 2-edge (except for w_1 and w_2).
– Some number (possibly none) of outgoing 1-edges.

Proof. Clearly each white vertex w has an incoming 1-edge. Now consider any successor b of w, which is a black vertex. The label of the edge (w, b) depends on whether w is a left, middle or right predecessor of b. If w is a right (left) predecessor, then (w, b) is labeled 2 (1), and its orientation is maintained. If w is a middle predecessor of b, then (w, b) is labeled 2 and turned around. Clearly the clockwise order of edges around w corresponds to whether w is a right, middle, left predecessor, so all that remains to argue is that if $w \neq w_1, w_2$, then it indeed must be the middle predecessor exactly once.

Consider the moment when we add the vertex b that makes w disappear from the outer-face. Then b must be black (white vertices have indegree 1), and adjacent to w by maximality, so w is a middle predecessor of b. It cannot be middle predecessor of anyone else, since it can disappear from the outer-face only once.

Lemma 2. *The 2-edges form a tree.*

Proof. Let v_1, w_1, v_n, w_2 be the outer-face in clockwise order. By the claims, every vertex except v_1, w_1, w_2 has exactly one incoming 2-edge, so there are $n-3$ 2-edges. w_2 has an outgoing 2-edge (to vertex v_n), so the graph spanned by 2-edges has $n-3$ edges and $n-2$ vertices. To show that this graph is a tree it therefore suffices to show that it has no cycles.

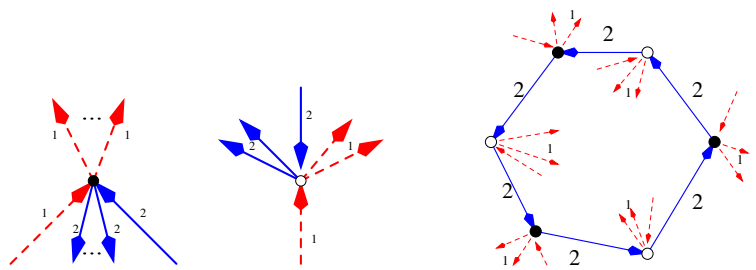

Fig. 3. Labeled edges around each vertex, and why no directed 2-cycle can exist

Assume we had a cycle of 2-edges C. Since every vertex has at most one incoming 2-edge, such a cycle must necessarily be directed. Assume that the cycle is directed counter-clockwise in the fixed planar embedding; the case of a clockwise directed cycle is very similar. By the first Claim, no black vertex on C can have 1-edges between the incoming 2-edge and the outgoing 2-edge of C. Hence, a black vertex on C has no incident 1-edges on the inside of C. Similarly by the second Claim a white vertex has no incident 1-edges on the outside of the cycle. See also Figure 3.

We know that the 1-edges form a rooted tree, and its root v_1 has no incident 2-edges and hence is not part of C. Now where is v_1 located? Assume that it is outside cycle C. Let w be a white vertex on C, and let b be its predecessor on the (unique) 1-path from v_1 to w. Vertex b is inside C by the above, hence there exists a directed 1-path from the outside of C to the inside of C. However, the order of edges around each vertex of C makes this impossible: any directed 1-path can reach C from the outside only at a black vertex, and is immediately directed back to the outside from there. Hence v_1 must be inside C. But now we can repeat the argument with a black vertex b on C; no directed 1-path can go from inside C to the neighbor of the incoming 1-edge of b (which is outside C). So we obtain a contradiction and no directed 2-cycle can exist.

Theorem 2. *Every maximal planar bipartite graph has a 2-tree partition. Furthermore, such a partition can be found in linear time.*

Not only gave we a split into two trees, we also obtained that the edges around each vertex are ordered in a special way when considering the incoming/outgoing edges of each tree. Note that this is similar to the edge-orderings obtained when splitting a triangulated planar graph into three trees [15]. An $\lfloor n/2 \rfloor \times \lceil n/2 - 1 \rceil$ grid drawing of planar bipartite graphs has been obtained in [1] using different techniques.

3 Tree Partitions of General Graphs

In this section we address the complexity of the tree partition problem. Recall that a k-tree partition of a graph is equivalent with a k-edge coloring such that each color induces a tree. Our problem is complementary to common and generalized partition and coloring problems of graphs which however address the vertices. Such problems have been studied in many versions, see, e.g., [2,3,8].

Clearly, a graph has a 1-tree partition if and only if it is a tree. This can be checked easily. All other cases are NP-hard. It turns out that connectivity is the crucial factor for the NP-hardness, since the partition into forests can be solved in polynomial time.

Theorem 3. *It is NP-hard to test whether a graph G has a 2-tree partition.*

Proof. We reduce from the Not-All-Equal-3SAT problem [8]. The construction extends the reduction to the 3-coloring problem in [13].

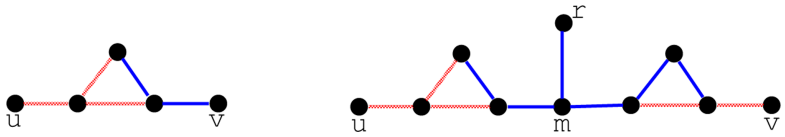

Fig. 4. Switch and Double-Switch

Let $\alpha = c_1, \ldots, c_m$ be an expression with clauses $c_1, \ldots, c_m$ and variables $x_1, \ldots, x_n$ and such that there is a clause $(x, x, \bar{x})$ for every variable x. The assignment must be such that there is a true and a false literal in each clause. Let $m > 1$. Construct a graph $G(\alpha)$ which has a 2-tree partition into a blue and a red tree if and only if α has a Not-All-Equal assignment. $G(\alpha)$ has a distinguished root r. The reduction is based on two gadgets, a *switch* and a *double-switch* between two vertices u and v, as shown in Fig. 4. In a double-switch there is a direct edge to the root. In both gadgets, only u and v (and r for the double-switches) may have edges incident to other vertices.

We immediately observe:

Claim. In a 2-tree partition,

- the edges of a triangle cannot belong to a single tree and must be colored with different colors,
- the edges at the endpoints u and v of a switch are colored by different colors, and
- the edges at the endpoints u and v of a double-switch are colored by the same color, which is different from the color of the edges at m.

We can now give the reduction.

For every variable x construct a switch $S(x)$ with the vertices x and $\bar{x}$, and directly connect these vertices to the root with an edge. For every clause c_i construct a triangle with vertices $c_i(1), c_i(2), c_i(3)$ which are identified with the literals. Connect each variable x in $S(x)$ with its occurrences in the clauses by paths of length two. Each such path p_j has a middle vertex m_j and two edges $e_j = (x, m_j)$ and $f_j = (m_j, x_j)$, where x_j is the j-th occurrence of x in some clause. Finally, connect any two such vertices m_i and m_j by a double-switch.

Hence each variable x has a gadget $H(x)$ consisting of the switch $S(x)$ and the triangle for the clause $(x, x, \bar{x})$ with the vertices x', x'' and $\bar{x}'$. There are paths between x and x', x and x'' and $\bar{x}$ and $\bar{x}'$, and there is a double-switch between the first two paths, see Fig. 5.

Claim. α has a Not-All-Equal truth assignment if and only if $G(\alpha)$ has a 2-tree partition.

Proof. First, suppose there is a Not-All-Equal truth assignment. Then color the edges from the root to each vertex representing a true literal blue, and the edges towards the false literals red, and for the edges between the root and the double-switches take the opposite color of the edges at the endpoints. Complete the

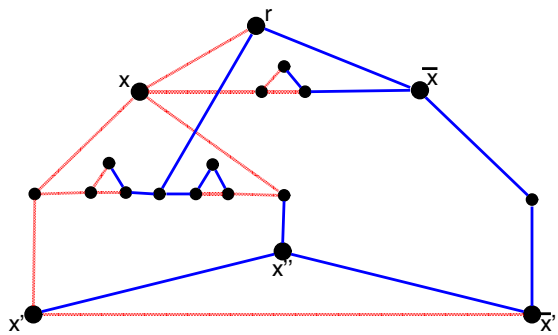

Fig. 5. The gadget of a variable x

coloring of the remaining edges of the switches and the double-switches. They separate edges with the same color at the endpoints. Finally, color the edges in the triangles of the clauses such that there are no monochrome cycles. The latter is doable, since there is a Not-All-Equal assignment with a true and a false literal for each clause. Thus there is a blue and a red path from the root to each triangle. Now all edges are properly colored, and each color induces a tree.

Conversely, if there is a 2-tree partition, then there is a Not-All-Equal assignment of the variables by the color of the edges on the monochrome paths in the gadgets $H(x)$, and there is a recoloring of the edges such that blue edges correspond to true and red edges to false.

Claim. For every gadget $H(x)$ the path from $\bar{x}$ to $\bar{x}'$ is monochrome, say blue, and there is blue path between $\bar{x}$ and the root. There is a monochrome red path between x and x' or x and x'' and a red path between x and the root.

Proof. By the first claim both colors are present in the triangles for the clause. Suppose the path between $\bar{x}$ and $\bar{x}'$ is not monochrome. Then it is a dead end and does not connect the edges in the triangle of the clause to the root. Now both colors in the triangle for the clause must come through x. Then there must be a blue path and a red path from x to x' and to x''. This is impossible by the double-switch, whose ends have the same color. Thus the path from $\bar{x}$ to $\bar{x}'$ is monochrome, and there must be a monochrome path of the other color to x' or x'' through x. All red (blue) edges must be connected by monochrome paths, and this connection can only be established through to root.

For every variable we color the vertex x in $S(x)$ by the color of the monochrome path to x' or x'' and accordingly for $\bar{x}$, and we assign x the value true if x is blue, and false if x is red. By the previous claim this is consistent for a pair $x, \bar{x}$.

Finally, to achieve a coloring which agrees with an assignment we may recolor some edges.

First, for every x all paths between x in $S(x)$ and x in the triangles of the clauses can be colored with the color of x.

Therefore, observe that by the double-switches two such paths cannot be monochrome and with different colors. Suppose that x is blue and let $p = (e, f)$ be a path from x with different colors for e and f. Then e cannot be red by the connectivity of the tree. If f is red, then it is a dead end. The two vertices of the triangle which are not incident with f are attached to a blue and a red edge, and these edges have monochromatic paths to the root. Now the edge f can be recolored blue. Subsequently, the edge in the triangle between the two vertices with an attached blue edge must be colored red, and one of the other edges in the triangle must be red.

Finally, suppose x is blue and the edge from x to the root is red. Then recolor this edge blue. This may induce blue cycles from the root to x, via a blue path to a triangle and back to the root. We break each such cycle in the triangle by recoloring the edge between the two attached blue edges and let another edge in the triangle be red. This is consistent with the 2-tree partition.

Now all edges incident with a vertex x of a literal are single-colored, and every clause has a red and a blue edge. Hence, there is a consistent Not-All-Equal truth assignment.

For $k \geq 3$ we reduce from the k-coloring problem. Let G be an instance of k-coloring. Add a new root r to G and connect all vertices of G with the root. Replace each edge (u, v) of G by a switch as shown in Fig. 4. Then the coloring of G one-to-one corresponds to the tree partition of the constructed graph. In any k-tree partition of the resulting graph if the edge from the root to a vertex v is in the i-th tree, then v is assigned the i-th color. By the switches, adjacent vertices are colored differently. Conversely, color the edges from a vertex v to the root according to the given color of v, color the remaining edges in the switches appropriately.

We summarize:

Theorem 4. *For every $k \geq 2$ is NP-hard whether a graph G has a k-tree partition.*

Acknowledgements

We thank an anonymous referee for the valuable suggestions.

References

1. T. Biedl and F. J. Brandenburg. Drawing planar bipartite graphs with small area. In *Proc. 17th Canadian Conference on Computational Geometry, CCCG'05, University of Windsor, Ontario, Canada, August 10-1, 2005*, volume 17, 2005.
2. A. Brandstädt, V. B. Le, and T. Symczak. The complexity of some problems related to graph 3-colorability. *Disc. Appl. Math.*, 89(1):59–73, 1998.
3. H. Broersma, F. V. Fomin, J. Kratochvil, and G. J. Woeginger. Planar graph coloring avoiding monochromatic subgraphs: Trees and paths make it difficult. *Algorithmica*, 44:343–361, 2006.

4. H. de Fraysseix, P. O. de Mendez, and J. Pach. Representation of planar graphs by segments. *Intuitive Geometry*, 63:109–117, 1991.
5. H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.
6. S. Even and R. E. Tarjan. Computing an *st*-numbering. *Theoretical Computer Science*, 2:436–441, 1976.
7. H. N. Gabow and H. H. Westermann. Forests, frames, and games: Algorithms for matroid sums and applications. *Algorithmica*, 7:465–497, 1992.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.
9. G. Kant. Drawing planar graphs using the canonical ordering. *Algorithmica*, 16:4–32, 1996.
10. A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs, International Symposium Rome 1966*, pages 215–232. Gordon and Breach, 1967.
11. C. S. J. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36:445–450, 1961.
12. C. S. J. Nash-Williams. Decomposition of finite graphs into forests. *J. London Math. Soc.*, 39:12, 1964.
13. C. Papadimitriou. *Computational Complexity.* Addison Wesley, Reading MA, 1994.
14. G. Ringel. Two trees in maximal planar bipartite graphs. *J. Graph Theory*, 17: 755–758, 1993.
15. W. Schnyder. Embedding planar graphs on the grid. In *1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990.