2-connecting Outerplanar Graphs without Blowing Up the Pathwidth

Jasine Babu[1], Manu Basavaraju[2], Sunil Chandran Leela[1], and Deepak Rajendraprasad[1]

[1] Indian Institute of Science, Bangalore 560012, India
[2] The Institute of Mathematical Sciences, Chennai 600113, India
{jasine,sunil,deepakr}@csa.iisc.ernet.in, manub@imsc.res.in

Abstract. Given a connected outerplanar graph G with pathwidth p, we give an algorithm to add edges to G to get a supergraph of G, which is 2-vertex-connected, outerplanar and of pathwidth $O(p)$. As a consequence, we get a constant factor approximation algorithm to compute a straight line planar drawing of any outerplanar graph, with its vertices placed on a two dimensional grid of minimum height. This settles an open problem raised by Biedl [3].

Keywords: Pathwidth, Outerplanar Graph, Bi-connected.

1 Introduction

A graph $G(V, E)$ is outerplanar, if it has a planar embedding with all its vertices lying on the outer face. Computing planar straight line drawings of planar graphs with vertices placed on a two dimensional grid, is a well known problem in graph drawing. The height of a grid is defined as the smaller of the two dimensions of the grid. If G has a planar straight line drawing, with its vertices placed on a two dimensional grid of height h, then we call it a planar drawing of G of height h. It is known that any planar graph on n vertices can be drawn on an $(n-1) \times (n-1)$ sized grid [11].

We use $\mathrm{pw}(G)$ to denote the pathwidth of a graph G. Pathwidth is a structural parameter of graphs, which is widely used in graph drawing and layout problems [3,5,13]. The study of pathwidth, in the context of graph drawings, was initiated by Dujmovic et al. [5]. It is known that any planar graph that has a planar drawing of height h has pathwidth at most h [13]. However, there exist planar graphs of constant pathwidth but requiring $\Omega(n)$ height in any planar drawing [2]. In the special case of trees, Suderman [13] showed that any tree T has a planar drawing of height at most $3\,\mathrm{pw}(T) - 1$. Biedl [3] considered the same problem for the bigger class of outerplanar graphs. For any bi-connected outerplanar graph G, Biedl [3] obtained an algorithm to compute a planar drawing of G of height at most $4\,\mathrm{pw}(G) - 3$. Since it is known that pathwidth is a lower bound for the height of the drawing [13], the algorithm given by Biedl [3] is a 4-factor approximation algorithm for the problem, for any bi-connected outerplanar graph. The method

D.-Z. Du and G. Zhang (Eds.): COCOON 2013, LNCS 7936, pp. 626–637, 2013.

in Biedl [3] is to add edges to the bi-connected outerplanar graph G to make it a maximal outerplanar graph H and then draw H on a grid of height $4\,\mathrm{pw}(G)-3$. The same method would give a constant factor approximation algorithm for arbitrary outerplanar graphs, if it is possible to add edges to any arbitrary outerplanar graph G to obtain a bi-connected outerplanar graph G' such that $\mathrm{pw}(G') = O(\mathrm{pw}(G))$. This was an open problem in Biedl [3].

In this paper, we give an algorithm to augment a connected outerplanar graph G of pathwidth p by adding edges so that the resultant graph is a bi-connected outerplanar graph of pathwidth $O(p)$. Notice that, the non-triviality lies in the fact that G' has to be maintained outerplanar. If we relax this condition, the problem becomes very easy. It is easy to verify that the supergraph G' of G, obtained by making any two vertices of G adjacent to each other and to every other vertex in the graph, is bi-connected and has pathwidth at most $\mathrm{pw}(G)+2$. The problem of augmenting outerplanar graphs to make them bi-connected, while maintaining the outerplanarity and optimizing some other properties, like number of edges added [6,9], have been investigated previously.

2 Background

A *tree decomposition* of a graph $G(V,E)$ [10] is a pair $(T,\mathscr{X})$, where T is a tree and $\mathscr{X} = (X_t : t \in V(T))$ is a family of subsets of $V(G)$, such that:

1. $\bigcup(X_t : t \in V(T)) = V(G)$.
2. For every edge e of G there exists $t \in V(T)$ such that e has both its end points in X_t.
3. For $t, t', t'' \in V(T)$, if t' is on the path of T between t and t'' then, $X_t \cap X_{t''} \subseteq X_{t'}$.

The width of the tree decomposition is $\max_{t\in V(T)}(|X_t|-1)$. Each $X_t \in \mathscr{X}$ is referred to as a bag in the tree decomposition. The graph G has *treewidth* w if w is the minimum such that G has a tree decomposition of width w. A *path decomposition* $(P,\mathscr{X})$ of a graph G is a tree decomposition of G with the additional property that the tree P is a path. The width of the path decomposition is $\max_{t\in V(P)}(|X_t|-1)$. The graph G has *pathwidth* w if w is the minimum such that G has a path decomposition of width w.

Without loss of generality we can assume that, in any path decomposition $(\mathcal{P},\mathscr{X})$ of G, the vertices of the path $\mathcal{P}$ are labeled as $1,2,\ldots$, in the order in which they appear in $\mathcal{P}$. Accordingly, the bags in $\mathscr{X}$ also get indexed as $1,2,\ldots$. For each vertex $v \in V(G)$, define $FirstIndex_{\mathscr{X}}(v) = \min\{i \mid X_i \in \mathscr{X} \text{ contains } v\}$, $LastIndex_{\mathscr{X}}(v) = \max\{i \mid X_i \in \mathscr{X} \text{ contains } v\}$ and $Range_{\mathscr{X}}(v) = [FirstIndex_{\mathscr{X}}(v), LastIndex_{\mathscr{X}}(v)]$. By the definition of a path decomposition, if $t \in Range_{\mathscr{X}}(v)$, then $v \in X_t$. If v_1 and v_2 are two distinct vertices, define $Gap_{\mathscr{X}}(v_1,v_2)$ as follows:

- If $Range_{\mathscr{X}}(v_1) \cap Range_{\mathscr{X}}(v_2) \neq \emptyset$, then $Gap_{\mathscr{X}}(v_1,v_2) = \emptyset$.
- If $LastIndex_{\mathscr{X}}(v_1) < FirstIndex_{\mathscr{X}}(v_2)$, then $Gap_{\mathscr{X}}(v_1,v_2) = [LastIndex_{\mathscr{X}}(v_1)+1, FirstIndex_{\mathscr{X}}(v_2)]$.
- If $LastIndex_{\mathscr{X}}(v_2) < FirstIndex_{\mathscr{X}}(v_1)$, then $Gap_{\mathscr{X}}(v_1,v_2) = [LastIndex_{\mathscr{X}}(v_2)+1, FirstIndex_{\mathscr{X}}(v_1)]$.

The motivation for this definition is the following. Suppose $(P, \mathscr{X})$ is a path decomposition of a graph G and v_1 and v_2 are two non-adjacent vertices of G. If we add a new edge between v_1 and v_2, a natural way to modify the path decomposition to reflect this edge addition is the following. If $Gap_{\mathscr{X}}(v_1, v_2) = \emptyset$, there is an $X_t \in \mathscr{X}$, which contains v_1 and v_2 together and hence, we need not modify the path decomposition. If $LastIndex_{\mathscr{X}}(v_1) < FirstIndex_{\mathscr{X}}(v_2)$, we insert v_1 into all $X_t \in \mathscr{X}$, such that $t \in Gap_{\mathscr{X}}(v_1, v_2)$. On the other hand, if $LastIndex_{\mathscr{X}}(v_2) < FirstIndex_{\mathscr{X}}(v_1)$, we insert v_2 to all $X_t \in \mathscr{X}$, such that $t \in Gap_{\mathscr{X}}(v_1, v_2)$. It is clear from the definition of $Gap_{\mathscr{X}}(v_1, v_2)$, that this procedure gives a path decomposition of the new graph. Whenever we add an edge (v_1, v_2), we stick to this procedure to update the path decomposition.

A *block* of a graph G is a maximal connected subgraph of G without a cut vertex. Every block of a connected graph G is thus either a single edge which is a bridge in G, or a maximal bi-connected subgraph of G. If a block of G is not a single edge, we call it as a non-trivial block of G. Given a connected outerplanar graph G, we define a rooted tree T (hereafter referred to as the *rooted block tree* of G) as follows. The vertices of T are the blocks of G and the root of T is an arbitrary block of G which contains at least one non-cut vertex (it is easy to see that such a block always exists). Two vertices B_i and B_j of T are adjacent if the blocks B_i and B_j share a cut vertex in G. It is easy to see that T, as defined above, is a tree. In our discussions, we restrict ourselves to a fixed rooted block tree of G. If block B_i is a child block of block B_j in the rooted block tree of G, and they share a cut vertex x, we say that B_i is a child block of B_j at x.

It is known that every bi-connected outerplanar graph has a unique Hamiltonian cycle [14]. Though the Hamiltonian cycle of a bi-connected block of G can be traversed either clockwise or anticlockwise, let us fix one of these orderings, so that the successor and predecessor of each vertex in the Hamiltonian cycle of the block is fixed. We call this order as the clockwise order. Consider a non-root block B_i of G such that B_i is a child block of its parent, at the cut vertex x. If B_i is a non-trivial block and y_i and y_i' respectively be the predecessor and successor of x in the Hamiltonian cycle of B_i, we call y_i as the last vertex of B_i and y_i' as the first vertex of B_i. If B_i is a trivial block, the neighbor of x in B_i is regarded as both the first vertex and the last vertex of B_i. By the term path we always mean a simple path, i.e., a path in which no vertex repeats.

3 An Overview of Our Method

Given a connected outerplanar graph $G(V, E)$ of pathwidth p, our algorithm will produce a bi-connected outerplanar graph $G''(V, E'')$ of pathwidth $O(p)$, where $E \subseteq E''$. Our algorithm proceeds in three stages.

(1) We use a modified version of the algorithm proposed by Govindan et al. [7] to obtain a *nice tree decomposition* (defined in Section 4) of G. Using this nice tree decomposition of G, we construct a special path decomposition of G of width at most $4p + 3$.

(2) For each cut vertex x of G, we define an ordering among the child blocks attached through x to their parent block. To define this ordering, we use the

special path decomposition constructed in the first stage. This ordering helps us in constructing an outerplanar supergraph $G'(V, E')$ whose pathwidth is at most $8p+7$, and for every cut vertex x in G', $G' \setminus x$ has exactly two components. The properties of the special path decomposition of G obtained in the first stage is crucially used in our argument to bound the width of the path decomposition of G', produced in the second stage.

(3) We bi-connect G' to construct $G''(V, E'')$, using a straightforward algorithm. As a by-product, this algorithm also gives us a surjective mapping from the cut vertices of G' to the edges in $E'' \setminus E'$. We give a counting argument based on this mapping and some basic properties of path decompositions to show that the width of the path decomposition of G'' produced in the third stage is at most $16p + 15$.

4 Stage 1: Construct a Nice Path Decomposition of G

In this section, we construct a *nice tree decomposition* of the outerplanar graph G and then use it to construct a *nice path decomposition* of G. We begin by giving the definition of a nice tree decomposition.

Given an outerplanar graph G, Govindan et al. [7, Section 2] gave a linear time algorithm to construct a width 2 tree decomposition $(T, \mathscr{Y})$ of G where $\mathscr{Y} = (Y_t : t \in V(T))$, with the following special properties:

1. There is a bijective mapping b from $V(G)$ to $V(T)$ such that $v \in Y_{b(v)}$. (Hereafter, for any $v \in V(G)$, while referring to the vertex $b(v)$ of T, we just call it as vertex v of T.)
2. If B_i is a child block of B_j at a cut vertex x, the induced subgraph T' of T on the vertex set $V(B_i \setminus x)$ is a spanning tree of $B_i \setminus x$ and $(T', \mathscr{Y}')$ where $\mathscr{Y}' = (Y_t : t \in V(T'))$ gives a tree decomposition of B_i.

Definition 1 (Nice tree decomposition of an outerplanar graph G). *A tree decomposition $(T, \mathscr{Y})$ of G, where $\mathscr{Y} = (Y_t : t \in V(T))$ having properties 1 and 2 above, together with the following additional property, is called a nice tree decomposition of G.*

3. *If y_i and y'_i are respectively the last and first vertices of a non-root, non-trivial block B_i, then the bag $Y_{y_i} \in \mathscr{Y}$ contains both y_i and y'_i.*

In the discussion that follows, we will show that any outerplanar graph G has a nice tree decomposition $(T, \mathscr{Y})$ of width at most 3. Initialize $(T, \mathscr{Y})$ to be the tree decomposition of G, constructed using the method proposed by Govindan et al. [7], satisfying properties 1 and 2 of nice tree decompositions. We need to modify $(T, \mathscr{Y})$ in such a way that, it satisfies property 3 as well.

For every non-root, non-trivial block B_i of G, do the following. If y_i and y'_i are respectively the last and first vertices of B_i, then, for each $t \in V(B_i \setminus x)$, we insert y'_i to Y_t, if it is not already present in Y_t and we call y'_i as a *propagated* vertex.

Claim 1. After the modification, $(T, \mathscr{Y})$ remains as a tree decomposition of G.

Claim 2. After the modification, $(T, \mathscr{Y})$ becomes a nice tree decomposition of G of width at most 3.

The proofs of the claims above are included in the full version [1]. From these claims, we can conclude the following.

Lemma 1. *Every outerplanar graph G has a nice tree decomposition $(T, \mathscr{Y})$ of width 3, constructable in polynomial time.*

Definition 2 (Nice path decomposition of an outerplanar graph). *Let $(\mathcal{P}, \mathscr{X})$ be a path decomposition of an outerplanar graph G. If, for every non-root non-trivial block B_i, there is a bag $X_t \in \mathscr{X}$ containing both the first and last vertices of B_i together, then $(\mathcal{P}, \mathscr{X})$ is called a nice path decomposition of G.*

Lemma 2. *Let G be outerplanar with $\mathrm{pw}(G) = p$. A nice path decomposition $(\mathcal{P}, \mathscr{X})$ of G, of width at most $4p + 3$, is constructable in polynomial time.*

Proof. Let $(\mathcal{T}, \mathscr{Y})$ be a nice tree decomposition of G of width 3, obtained using Lemma 1. Obtain an optimal path decomposition $(\mathcal{P}_T, \mathscr{X}_T)$ of the tree $\mathcal{T}$ in polynomial time, using a standard algorithm (See [12]). Since $\mathcal{T}$ is a spanning tree of G, the pathwidth of $\mathcal{T}$ is at most that of G. Therefore, the width of the path decomposition $(\mathcal{P}_T, \mathscr{X}_T)$ is at most p; i.e. there are at most $p + 1$ vertices of $\mathcal{T}$ in each bag $X_{T_i} \in \mathscr{X}_T$.

Let $\mathcal{P} = \mathcal{P}_T$ and for each $X_{T_i} \in \mathscr{X}_T$, we define $X_i = \bigcup_{v_T \in X_{T_i}} Y_{v_T}$. Clearly, $(\mathcal{P}, \mathscr{X})$, with $\mathscr{X} = (X_1, \ldots, X_{|V(\mathcal{P}_T)|})$, is a path decomposition of G (See [7]). The width of this path decomposition is at most $4(p + 1) - 1 = 4p + 3$, since $|Y_{v_T}| \leq 4$, for each bag $Y_{v_T} \in \mathscr{Y}$ and $|X_{T_i}| \leq p + 1$, for each bag $X_{T_i} \in \mathscr{X}_T$.

Let B_i be a non-root, non-trivial block in G and y_i and y_i' respectively be the first and last vertices of B_i. Since y_i is a vertex of the tree $\mathcal{T}$, there is some bag $X_{T_j} \in \mathscr{X}_T$, containing y_i. The bag $Y_{y_i} \in \mathscr{Y}$ contains both y_i and y_i', since $(\mathcal{T}, \mathscr{Y})$ is a nice tree decomposition of G. It follows from the definition of X_j, that $X_j \in \mathscr{X}$ contains both y_i and y_i'. Therefore, $(\mathcal{P}, \mathscr{X})$ is a nice path decomposition of G. □

5 Edge Addition without Spoiling the Outerplanarity

In this section, we give two technical lemmas, which will be later used to prove that the intermediate graph G' obtained in Stage 2 and the bi-connected graph G'' obtained in Stage 3 are outerplanar. These lemmas are easy to visualize (See Fig 1). The proofs are included in the full version [1].

Lemma 3. *Let $G(V, E)$ be a connected outerplanar graph. Let u and v be two distinct non-adjacent vertices in G and let $P = (u = x_0, x_1, x_2, \ldots, x_k, x_{k+1} = v)$ where $k \geq 1$ be a path in G such that:*

P shares at most one edge with any block of G.

For $0 \leq i \leq k$, if the block containing the edge (x_i, x_{i+1}) is non-trivial, then x_{i+1} is the successor of x_i in the Hamiltonian cycle of that block.

Then the graph $G'(V, E')$, where $E' = E \cup \{(u, v)\}$, is outerplanar.

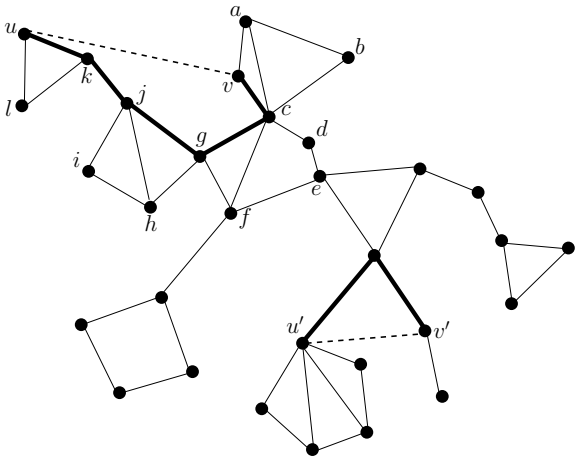

Fig. 1. The path between u and v and the path between u' and v' (shown in thick edges) satisfy the conditions stated in Lemma 3. By adding any one of the dotted edges (u, v) or (u', v'), the graph remains outerplanar. When the edge (u, v) is added, $u, v, a, b, c, d, e, f, g, h, i, j, k, l, u$ is the Hamiltonian cycle of the new block formed.

The following lemma explains the effect of the addition of an edge (u, v) as mentioned in Lemma 3, to the block structure and the Hamiltonian cycle of each block. Assume that for $0 \leq i \leq k$, the edge (x_i, x_{i+1}) belongs to the block B_i.

Lemma 4. *1. Other than the blocks B_0 to B_k of G merging together to form a new block B' of G', blocks in G and G' are the same.*

2. Vertices in blocks B_0 to B_k, except x_i, $0 \leq i \leq k+1$, retains their successor and predecessor in the Hamiltonian cycle of B' same as it was in its respective block's Hamiltonian cycle in G.

3. Each x_i, $0 \leq i \leq k$, retains its Hamiltonian cycle predecessor in B' same as it was in the block B_i of G and each x_i, $1 \leq i \leq k+1$, retains its Hamiltonian cycle successor in B' same as in the block B_{i-1} of G.

6 Stage 2: Construction of G' and Its Path Decomposition

For each cut vertex x of G, we define an ordering among the child blocks attached through x to their parent block, using the nice path decomposition $(\mathcal{P}, \mathcal{X})$ of G obtained using Lemma 2. This ordering is then used in defining a supergraph $G'(V, E')$ of G such that for every cut vertex x in G', $G' \setminus x$ has exactly two components. Using repeated applications of Lemma 3, we then show that G' is outerplanar. We extend the path decomposition $(\mathcal{P}, \mathcal{X})$ of G to a path decomposition $(\mathcal{P}', \mathcal{X}')$ of G', as described in Section 2. By a counting argument using the properties of the nice path decomposition $(\mathcal{P}, \mathcal{X})$, we show that the width of $(\mathcal{P}', \mathcal{X}')$ is at most $2p' + 1$, where p' is the width of $(\mathcal{P}, \mathcal{X})$.

6.1 Defining an Ordering of Child Blocks

If $(\mathcal{P}, \mathscr{X})$ is a nice path decomposition of G, then, for each non-root block B of G, at least one bag in $\mathscr{X}$ contains both the first and last vertices of B together.

Definition 3 (Sequence number of a non-root block). *Let $(\mathcal{P}, \mathscr{X})$ be the nice path decomposition of G obtained using Lemma 2. For each non-root block B of G, we define the sequence number of B as* $\min\{i \mid X_i \in \mathscr{X}$ *simultaneously contains both the first and last vertices of* $B\}$.

For each cut vertex x, there is a unique block B^x such that B^x and its child blocks are intersecting at x. For each cut vertex x, we define an ordering among the child blocks attached at x, as follows. If $B_1, \ldots, B_k$ are the child blocks attached at x, we order them in the increasing order of their sequence numbers in $(\mathcal{P}, \mathscr{X})$. If B_i and B_j are two child blocks with the same sequence number, their relative ordering is arbitrary.

From the ordering defined, we can make some observations about the appearance of the first and last vertices of a block B_i in the path decomposition. These observations are crucially used for bounding the width of the path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'. Let $B_1, \ldots, B_k$ are the child blocks attached at a cut vertex x, occurring in that order according to the ordering we defined above. For $1 \leq i \leq k$, let y_i and y'_i respectively be the last and first vertices of B_i.

Property 1. For any $1 \leq i \leq k-1$, if $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$, then $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i)+1, FirstIndex_{\mathscr{X}}(y_{i+1})]$ and $x \in X_t$ for all $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$.

Property 2. For any $1 \leq i < j \leq k-1$, $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \cap Gap_{\mathscr{X}}(y'_j, y_{j+1}) = \emptyset$.
The proofs of these properties directly follow from the definitions and are given in the full version [1].

6.2 Algorithm for Constructing G' and Its Path Decomposition

We use Algorithm 1 to construct $G'(V, E')$ and a path decomposition $(\mathcal{P}', \mathscr{X}')$ of G'. The processing of each cut vertex is done in lines 2 to 7 of Algorithm 1. While processing a cut vertex x, the algorithm adds the edges $(y'_1, y_2), (y'_2, y_3), \ldots, (y'_{k_x-1}, y_{k_x})$ (as defined in the algorithm) and modifies the path decomposition, to reflect each edge addition.

Lemma 5. *G' is outerplanar and for each cut vertex x of G', $G' \setminus x$ has exactly two components.*

We can prove this by applying Lemma 3, following the addition of each edge in $E' \setminus E$ by Algorithm 1. Refer to the full version [1] for the proof.

Lemma 6. *$(\mathcal{P}', \mathscr{X}')$ is a path decomposition of G' of width at most $8p + 7$.*

Proof. Algorithm 1 initialized $(\mathcal{P}', \mathscr{X}')$ to $(\mathcal{P}, \mathscr{X})$ and modified it during each edge addition. By Property 1, we have $Gap_{\mathscr{X}}(y'_i, y_{i+1}) = [LastIndex_{\mathscr{X}}(y'_i) + 1, FirstIndex_{\mathscr{X}}(y_{i+1})]$. Hence, by the modification done in lines 7 to 7 while adding a new edge (y'_i, y_{i+1}), $(\mathcal{P}', \mathscr{X}')$ becomes a path decomposition of the

Algorithm 1. Computing G' and its path decomposition

Input: An outerplanar graph $G(V, E)$ and a nice path decomposition $(\mathcal{P}, \mathcal{X})$ of G, the rooted block tree of G, the Hamiltonian cycle of each non-trivial block of G and the first and last vertices of each non-root block of G

Output: An outerplanar supergraph $G'(V, E')$ of G such that, for every cut vertex x of G', $G' \setminus x$ has exactly two connected components, a path decomposition $(\mathcal{P}', \mathcal{X}')$ of G'

1 $E' = E$, $(\mathcal{P}', \mathcal{X}') = (\mathcal{P}, \mathcal{X})$
2 **for** *each cut vertex* $x \in V(G)$ **do**
3 Let $B_1, \ldots, B_{k_x}$, in that order, be the child blocks attached at x, according to the ordering defined in Section 6.1
4 **for** $i = 1$ *to* $k_x - 1$ **do**
5 Let y'_i be the first vertex of B_i and y_{i+1} be the last vertex of B_{i+1}
6 $E' = E' \cup \{(y'_i, y_{i+1})\}$
7 **if** $Gap_{\mathscr{X}}(y'_i, y_{i+1}) \neq \emptyset$ **then for** $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$ **do** $X'_t = X'_t \cup \{y'_i\}$

graph containing the edge (y'_i, y_{i+1}), by the method explained in Section 2. It follows that $(\mathcal{P}', \mathcal{X}')$ is a path decomposition of G'.

Consider any $X'_t \in \mathcal{X}'$. While processing the cut vertex x, if Algorithm 1 inserts a new vertex y'_i to X'_t, to reflect the addition of a new edge (y'_i, y_{i+1}) then, $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$. Suppose (y'_i, y_{i+1}) and (y'_j, y_{j+1}) are two new edges added while processing the cut vertex x, where, $1 \leq i < j \leq k_x - 1$. By property 2, we know that if $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$, then, $t \notin Gap_{\mathscr{X}}(y'_j, y_{j+1})$. Therefore, when the algorithm is processing a cut vertex x in lines 2 to 7, at most one vertex is newly getting inserted to X'_t. Moreover, if $t \in Gap_{\mathscr{X}}(y'_i, y_{i+1})$ then, the cut vertex $x \in X_t$, by Property 1. It follows that $|X'_t| \leq |X_t|$+number of cut vertices in $X_t \leq 2|X_t| \leq 2(4p + 4)$. Therefore, the width of the path decomposition $(\mathcal{P}', \mathcal{X}')$ is at most $8p + 7$. □

7 Construction of G'' and Its Path Decomposition

In this section, we give an algorithm to add some more edges to $G'(V, E')$ so that the resultant graph $G''(V, E'')$ is bi-connected. The algorithm also extend the path decomposition $(\mathcal{P}', \mathcal{X}')$ of G' to a path decomposition $(\mathcal{P}'', \mathcal{X}'')$ of G''. An analysis of the algorithm shows the existence of a surjective mapping from the cut vertices of G' to the edges in $E'' \setminus E'$. A counting argument based on the surjective mapping shows that the width of the path decomposition $(\mathcal{P}'', \mathcal{X}'')$ is at most $16p + 15$. For making our presentation simpler, if a block B_i is just an edge (u, v), we abuse the definition of a Hamiltonian cycle and say that u and v are clockwise neighbors of each other in the Hamiltonian cycle of B_i.

Recall that the graph G' has the property that for every cut vertex x of G', $G' \setminus x$ has exactly two components. Since any cut vertex belongs to exactly two blocks of G, based on the rooted block tree structure of G, we call them as the parent block containing x and the child block containing x. We use $child_x(B)$ to denote the child block of the block B at the cut vertex x and $parent(B)$ to

denote the parent block of the block B. For a block B, $next_B(v)$ denotes the successor of the vertex v in the Hamiltonian cycle of B.

To get an intuition about our algorithm, the reader may consider it as a traversal of vertices of G', starting from a non-cut vertex in the root block of G' and proceeding to the successor of v on reaching a non-cut vertex v. On reaching a cut vertex x, the algorithm recursively traverses the child block containing x and its descendant blocks and comes back to x to continue the traversal of the remaining graph. However, before starting the recursive traversal of the child block containing x and its descendant blocks, the algorithm sets $bypass(x) = TRUE$. (Note that, since there is only one child block attached to any cut vertex, each cut vertex is bypassed only once.) In this way, when a sequence of one or more cut vertices is bypassed, an edge is added from the vertex preceding the first bypassed vertex in the sequence to the vertex succeeding the last bypassed vertex in the sequence. The path decomposition is also modified, to reflect this edge addition. The detailed algorithm to bi-connect G' is given in Algorithm 2. The following Lemma summarizes some observations about how Algorithm 2 works. A proof this lemma can be found in the full version [1].

Algorithm 2. Computing a bi-connected outerplanar supergraph

Input: An outerplanar graph $G'(V, E')$ such that $G' \setminus x$ has exactly two connected components for every cut vertex x of G'. A path decomposition $(\mathcal{P}', \mathcal{X}')$ of G'. The rooted block tree of G', the Hamiltonian cycle of each non-trivial block of G' and the first and last vertices of each non-root block of G'

Output: A bi-connected outerplanar supergraph $G''(V, E'')$ of G', a path decomposition $(\mathcal{P}'', \mathcal{X}'')$ of G''

1 $E'' = E'$, $(\mathcal{P}'', \mathcal{X}'') = (\mathcal{P}', \mathcal{X}')$
2 **for** *each vertex* $v \in V(G')$ **do**
3 completed(v) = FALSE, **if** v *is a cut vertex* **then** bypass(v) = FALSE
4 Choose v to be some non-cut vertex of the root block
5 B = root block, *completed*(v) =TRUE, *completedCount* = 1
6 **while** *completedCount* $< |V(G')|$ **do**
7 $v' = next_B(v)$
8 **while** v' *is a cut vertex and* $bypass(v')$ *is FALSE* **do**
9 $bypass(v')$ =TRUE, $B = child_{v'}(B)$, $v' = next_B(v')$
10 **if** v' *is a cut vertex and* $bypass(v')$ *is TRUE* **then** $B = parent(B)$
11 *completed*(v')= TRUE, *completedCount* = *completedCount* + 1
12 **if** (v, v') *is not an edge in* G' **then**
13 $E'' = E'' \cup \{(v, v')\}$
14 **if** $Gap_{\mathcal{X}'}(v, v') \neq \emptyset$ **then**
15 **if** $LastIndex_{\mathcal{X}'}(v) < FirstIndex_{\mathcal{X}'}(v')$ **then for** $t \in Gap_{\mathcal{X}'}(v, v')$ **do** $X''_t = X''_t \cup \{v\}$
16 **else if** $LastIndex_{\mathcal{X}'}(v') < FirstIndex_{\mathcal{X}'}(v)$ **then for** $t \in Gap_{\mathcal{X}'}(v, v')$ **do** $X''_t = X''_t \cup \{v'\}$
17 $v = v'$

Lemma 7. *1. Inside a block, the algorithm traverses vertices in the clockwise order of the unique Hamiltonian cycle of the block.*
2. *When the algorithm encounters a non-cut vertex x during the traversal, it declares that x is completed.*
3. *The algorithm encounters a cut vertex x for the first time, while traversing the parent block containing x. Then, the algorithm bypasses x (i.e. set $bypass(x) = TRUE$) and descends to the child block containing x and start traversing the child block from the successor of x in the child block's Hamiltonian cycle.*
4. *When the algorithm encounters a cut vertex x for a second time, the current block being traversed is the child block containing x. Then the algorithm traverses x and declare that x is completed and ascends to the parent block containing x. Then it continues the traversal of the parent block containing x, by considering the successor of x in the parent block's Hamiltonian cycle.*
5. *When the algorithm declares that a cut vertex x is completed, all vertices of the child block containing x and all its descendant blocks have been completed.*
6. *Every vertex is encountered at least once. Every vertex is completed and a vertex which is declared completed is never encountered again. When $completedCount = |V(G')|$, all the vertices of the graph have been completed.*
7. *When the algorithm is bypassing a sequence of one or more cut vertices, an edge is added from the vertex preceding the first bypassed vertex in the sequence to the vertex succeeding the last bypassed vertex in the sequence and the path decomposition is modified, to reflect this edge addition.*
8. *Every new edge added has a sequence of bypassed cut vertices associated with it. If $x_1, x_2, \ldots, x_k$ is the sequence of bypassed cut vertices associated with an edge $(u, v) \in E'' \setminus E'$, then $u, x_1, x_2, \ldots, x_k, v$ is a path in G'. Each cut vertex of G' is bypassed exactly once in our traversal and hence associated with a unique edge in $E'' \setminus E'$.*

Lemma 8. *G'' is bi-connected.*

Proof. We show that G'' does not have any cut vertices. Since G'' is a supergraph of G', if a vertex x is not a cut vertex in G', it will not be a cut vertex in G''. We need to show that the cut vertices in G' become non-cut vertices in G''. Consider a newly added edge (u, v) of G''. Without loss of generality, assume that u was completed before v in the traversal, and $(x_1, x_2, \ldots, x_k)$ is the sequence of bypassed cut vertices associated with the edge (u, v). When our algorithm adds the edge (u, v), it creates the cycle $u, x_1, x_2, \ldots, x_k, v, u$ in the resultant graph. Recall that, for each $1 \leq i \leq k$, $G' \setminus x_i$ had exactly two components; one containing x_{i-1} and the other containing x_{i+1}. After the addition of the edge, vertices x_{i-1}, x_i and x_{i+1} lie on a common cycle. Hence, when the edge (u, v) is added, for $1 \leq i \leq k$, x_i is no longer a cut vertex. Since every cut vertex in G' was part of the bypass sequence associated with some edge in $E'' \setminus E'$, all of them become non-cut vertices in G''. □

Lemma 9. *G'' is outerplanar.*

For a proof of this lemma, refer to the full version [1].

Lemma 10. *$(\mathcal{P}'', \mathcal{X}'')$ is a path decomposition of G'' of width at most $16p+15$.*

Proof. It is clear that $(\mathcal{P}'', \mathcal{X}'')$ is a path decomposition of G'', since we constructed it using the method explained in Section 2.

For each $1 \leq i \leq m$, let $S_i = \{x_1, \ldots, x_k\}$ denote the set of cut vertices that belong to the bypassed cut vertex sequence associated with the edge $e_i = (u_i, v_i) \in E'' \setminus E'$. While adding the edge e_i, a vertex was inserted into $X''_t \in \mathcal{X}''$ only if $t \in Gap_{\mathcal{X}'}(u_i, v_i)$. We will now show that, if $t \in Gap_{\mathcal{X}'}(u_i, v_i)$, then, $X'_t \cap S_i \neq \emptyset$. Without loss of generality, assume that $LastIndex_{\mathcal{X}'}(u_i) < FirstIndex_{\mathcal{X}'}(v_i)$. Let $x_1, \ldots, x_k$ be the sequence of cut vertices bypassed while adding the edge (u_i, v_i). Since u_i is adjacent to x_1, both of them are together present in some bag in $X'_t \in \mathcal{X}'$, with $t \leq LastIndex_{\mathcal{X}'}(u_i)$. Similarly, since v_i is adjacent to x_k, they both are together present in some bag $X'_t \in \mathcal{X}'$, with $t \geq FirstIndex_{\mathcal{X}'}(v_i)$. The sequence $x_1, \ldots, x_k$ is a path in G' between x_1 and x_k. Therefore, every bag in $X'_t \in \mathcal{X}'$ with $t \in Gap_{\mathcal{X}'}(u_i, v_i)$ should contain at least one of the cut vertices from the set S_i.

Thus, by the modification done to the path decomposition to reflect the addition of the edge e_i, the size of each bag in $X''_t \in \mathcal{X}''$ with $t \in Gap_{\mathcal{X}'}(u_i, v_i)$ increases by exactly one and in that case, $X'_t \cap S_i \neq \emptyset$. The other bags are unaffected by this modification. Therefore, for any t in the index set, $|X''_t| = |X'_t| + |\{i \mid 1 \leq i \leq m, S_i \cap X'_t \neq \emptyset\}|$. But, $|\{i \mid 1 \leq i \leq m, S_i \cap X'_t \neq \emptyset\}| \leq |X'_t|$, because $S_i \cap S_j = \emptyset$, for $1 \leq i < j \leq m$, by part 8 of Lemma 7. Therefore, for any t, $|X''_t| \leq 2|X'_t| \leq 2(8p + 8)$. Therefore, width of the path decomposition $(\mathcal{P}'', \mathcal{X}'')$ is at most $16p + 15$. □

8 Efficiency

The preprocessing step of computing a rooted block tree of the given outerplanar graph G and finding the Hamiltonian cycles of each non-trivial block can be done in linear time [4,8,14]. The special tree decomposition in Govindan et al.[7] is also computable in linear time. Using the Hamiltonian cycle of each non-trivial block, we did only a linear time modification in Section 4, to produce the nice tree decomposition $(T, \mathcal{Y})$ of G. An optimal path decomposition of the tree T, of total size $O(n \text{ pw}(T))$ can be computed in time $O(n \text{ pw}(T))$[12]. The time taken is $O(n \log n)$, since outerplanar graphs have pathwidth at most $\log n$, and T was a spanning tree of the outerplanar graph G. For computing the nice path decomposition $(\mathcal{P}, \mathcal{X})$ of G in Section 4, the time spent is linear in the size of the path decomposition obtained for T, i.e, $O(n \log n)$ and the total size of $(\mathcal{P}, \mathcal{X})$ is $O(n \log n)$. Computing the FirstIndex, LastIndex and Range of vertices and the sequence number of blocks can be done in time linear in the size of the path decomposition. Since the resultant graph is outerplanar, Algorithm 1 and Algorithm 2 adds only a linear number of new edges. Since the size of each bag in the path decompositions $(\mathcal{P}', \mathcal{X}')$ of G' and $(\mathcal{P}'', \mathcal{X}'')$ of G'' are only a constant times the size of the corresponding bag in $(\mathcal{P}, \mathcal{X})$, the time taken for modifying $(\mathcal{P}, \mathcal{X})$ to obtain $(\mathcal{P}', \mathcal{X}')$ and later modifying it to $(\mathcal{P}'', \mathcal{X}'')$ takes

only time linear in size of $(\mathcal{P}, \mathcal{X})$; i.e., $O(n \log n)$ time. Hence, the time spent in constructing G'' and its path decomposition of width $O(\mathrm{pw}(G))$ is $O(n \log n)$.

9 Conclusion

In this paper, we have described a $O(n \log n)$ time algorithm to add edges to a given outerplanar graph G of pathwidth p to get a bi-connected outerplanar graph G'' of pathwidth at most $16p + 15$. We also get the corresponding path decomposition of G'' in $O(n \log n)$ time. Our technique is to produce a nice path decomposition of G and make use of the properties of this decomposition, while adding the new edges. Our algorithm can be used as a preprocessing step, in the algorithm proposed by Biedl [3], to produce a planar drawing of G on a grid of height $O(p)$. As explained by Biedl [3], this is a constant factor approximation algorithm, to get a planar drawing of G of minimum height.

References

1. Babu, J., Basavaraju, M., Chandran, L.S., Rajendraprasad, D.: 2-connecting outerplanar graphs without blowing up the pathwidth. CoRR abs/1212.6382 (2012), http://arxiv.org/abs/1212.6382
2. Biedl, T.: Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs. Discrete Comput. Geom. 45(1), 141–160 (2011)
3. Biedl, T.: A 4-approximation for the height of 2-connected outer-planar graph drawings. In: WAOA 2012 (2012)
4. Chartrand, G., Harary, F.: Planar permutation graphs. Annales de l'institut Henri Poincaré (B) Probabilités et Statistiques 3, 433–438 (1967)
5. Dujmovic, V., Morin, P., Wood, D.R.: Path-width and three-dimensional straight-line grid drawings of graphs. In: Goodrich, M.T., Kobourov, S.G. (eds.) GD 2002. LNCS, vol. 2528, pp. 42–53. Springer, Heidelberg (2002)
6. García, A., Hurtado, F., Noy, M., Tejel, J.: Augmenting the connectivity of outerplanar graphs. Algorithmica 56(2), 160–179 (2010)
7. Govindan, R., Langston, M.A., Yan, X.: Approximating the pathwidth of outerplanar graphs. Inf. Process. Lett. 68(1), 17–23 (1998)
8. Hopcroft, J., Tarjan, R.: Algorithm 447: Efficient algorithms for graph manipulation. Commun. ACM 16(6), 372–378 (1973)
9. Kant, G.: Augmenting outerplanar graphs. Journal of Algorithms 21(1), 1–25 (1996)
10. Robertson, N., Seymour, P.D.: Graph minors. iii. planar tree-width. J. Comb. Theory, Ser. B 36(1), 49–64 (1984)
11. Schnyder, W.: Embedding planar graphs on the grid. In: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1990, pp. 138–148 (1990)
12. Skodinis, K.: Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. J. Algorithms 47(1), 40–59 (2003)
13. Suderman, M.: Pathwidth and layered drawings of trees. Int. J. Comput. Geometry Appl. 14(3), 203–225 (2004)
14. Syslo, M.M.: Characterizations of outerplanar graphs. Discrete Mathematics 26(1), 47–53 (1979)