Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 381–410 (2012)
DOI: 10.7155/jgaa.00268

# Subgraph Homeomorphism via the Edge Addition Planarity Algorithm

*John M. Boyer*[1]

[1]IBM Canada Software Lab
200 - 4396 West Saanich Rd., Victoria, BC, Canada V8Z 3E9

**Abstract**

This paper extends the edge addition planarity algorithm from Boyer and Myrvold to provide a new way of solving the subgraph homeomorphism problem for $K_{2,3}$, $K_4$, and $K_{3,3}$. These extensions derive much of their behavior and correctness from the edge addition planarity algorithm, providing an alternative perspective on these subgraph homeomorphism problems based on affinity with planarity rather than triconnectivity. Reference implementations of these algorithms have been made available in an open source project (http://code.google.com/p/planarity).

| Submitted: | Reviewed: | Revised: | Reviewed: | Revised: |
|---|---|---|---|---|
| November 2010 | June 2011 | July 2011 | March 2012 | April 2012 |
| | Accepted: | Final: | Published: | |
| | July 2012 | July 2012 | August 2012 | |

| Article type: | Communicated by: |
|---|---|
| Regular paper | G. Liotta |

*E-mail address:* boyerj@ca.ibm.com; jboyer@acm.org (John M. Boyer)

## 1 Introduction

Recent advancements in combinatorial planar embedding techniques have yielded new vertex addition planarity methods in which an *st*-numbering and a PQ-tree are replaced by depth first search (DFS) and simpler data structures [2, 19]. These vertex addition methods were rationalized with PQ-tree operations by Haeupler and Tarjan [10]. However, using some of the same principles, Boyer and Myrvold [3] developed a new planarity algorithm whose processing model and proof of correctness use edge addition as the atomic operation of planar embedding. For each edge addition, the following two invariant conditions are maintained: first, the planar embedding is a collection of biconnected components formed by the edges added so far; second, a vertex is kept on the external face boundary of its containing biconnected component if the vertex is an endpoint of an unembedded back edge or is a cut vertex on the DFS path between the endpoints of an unembedded back edge.

A planarity characterization by de Fraysseix and Rosenstiehl [6] has been shown to lead to another linear time planarity algorithm [4, 5] that is important because it also embeds edges based on resolving conflicts between DFS back edges as seen from a bottom-up view of the depth first search tree. In comparison, the Boyer-Myrvold algorithm [3] is guided by the DFS vertex numbering, and the planar embedding is constructed by adding each DFS tree edge as a singleton biconnected component and adding each back edge along the external face boundaries of the biconnected components in the planar embedding while maintaining the above invariant conditions.

The processing model of the edge addition planarity algorithm [3] enabled non-planarity to be detected with only two tests, the sufficiency of which was proven with a set of only five minors, four of $K_{3,3}$ and one of $K_5$. Then, a Kuratowski subgraph isolator, i.e. an algorithm that finds a subgraph homeomorphic to $K_{3,3}$ or $K_5$ in any non-planar graph, was developed through further analysis of the $K_5$ minor, which produced four additional $K_{3,3}$ patterns. This paper extends that Kuratowski subgraph isolator to a linear time $K_{3,3}$ search, i.e. an algorithm that finds a subgraph homeomorphic to $K_{3,3}$ in a graph, if one exists.

As with any planarity test, the edge addition algorithm can also be simply modified to provide an outerplanarity test instead. The adjusted algorithm produces an outerplanarity obstruction, i.e. a subgraph homeomorphic to $K_{2,3}$ or $K_4$, in ways that closely match how the edge addition planarity algorithm produces a subgraph homeomorphic to $K_{3,3}$ or $K_5$. This paper describes how the outerplanarity obstructions arise in the edge addition algorithm as the first step of extending to a linear time $K_{2,3}$ search and a linear time $K_4$ search, i.e. $O(n)$ algorithms for finding a subgraph homeomorphic to $K_{2,3}$, if one exists, and finding a subgraph homeomorphic to $K_4$, if one exists.

For the algorithms in this paper, we are given a search graph $H$, and an augmented form of the edge addition planarity algorithm (for $H = K_{3,3}$) or outerplanarity algorithm (for $H = K_{2,3}$ or $H = K_4$) is performed on an input graph $G$ to either find a subgraph homeomorphic to $H$ or determine that $G$ is

$H$-less (has no subgraph homeomorphic to $H$). If $G$ is found to be planar or outerplanar, then $G$ contains no subgraph homeomorphic to $H$. If a planar or outerplanar obstruction is found in $G$, and if it is not homeomorphic to $H$, then some additional analyses are performed to see whether a subgraph homeomorphic to $H$ is entangled with the obstruction. If not, then the obstruction is reduced, an action that removes part of the obstruction thereby allowing the edge addition planarity or outerplanarity algorithm to proceed with the search for a subgraph homeomorphic to $H$. A reduction also typically replaces some parts of the graph with single edges to help achieve linear time performance. Therefore, if a subgraph homeomorphic to $H$ is subsequently found, then some additional post-processing is performed to replace the reduction edges with paths from $G$.

As Williamson [22] once said of Kuratowski subgraph isolation, it is desirable to have not one but several basically different optimal methods for solving a problem because the requirement of optimality forces the emergence of greater insight into underlying theoretic phenomena. An elegant case in point is found in the comparison of the first linear time $K_4$ search in [17] with the linear time $K_4$ search by Asano [1]. The former has priority, whereas Asano's work used triconnectivity to explore a wider class of subgraph homeomorphism problems, which also included polynomial time solutions for $K_{2,3}$ and $K_{3,3}$ search. To achieve linear time performance, Asano's searches rely on a linear time triconnectivity algorithm (e.g., see [9, 11, 21]), and the $K_{3,3}$ search also relies on linear time planarity testing and Kuratowski subgraph isolation [8].

Given the necessity of a planarity algorithm in the prior $K_{3,3}$ search algorithms and the straightforward analogy by which outerplanar obstruction isolation processing may be associated with subgraph homeomorphism searches for $K_{2,3}$ and $K_4$ searches, this paper presents different optimal methods that explore the affinity of these subgraph homeomorphism problems to planarity rather than triconnectivity. A number of the path analyses described in [1, 8] are simply implicit in the operation of the edge addition planarity algorithm. Moreover, rather than fully decomposing the graph into triconnected components, these new algorithms only selectively eliminate or reduce to a single edge certain subgraphs that are separable by a pair of vertices, hereafter called a 2-cut, once they are found to meet specific conditions related to planarity or outerplanarity obstructions. Thus, these new algorithms consist primarily of extended use of the same techniques that are used by the core edge addition planarity algorithm [3] to perform planar embedding and Kuratowski subgraph isolation.

Section 2 provides an updated overview of the core edge addition planarity algorithm [3], including a comprehensive example of several edge additions. Section 3 describes the edge addition outerplanarity algorithm and particularly how outerplanarity obstructions arise. Section 4 extends this method to provide a search for subgraphs homeomorphic to $K_{2,3}$, and Section 5 provides an extension for finding subgraphs homeomorphic to $K_4$. Section 6 extends the core planarity algorithm to provide a search for subgraphs homeomorphic to $K_{3,3}$. Finally, conclusions and future work are discussed in Section 7.

## 2 Edge Addition Planarity Overview (Updated)

The edges of a simple undirected input graph $G$ are added one at a time to the combinatorial planar embedding $\tilde{G}$ in such a way that planarity is preserved by the edge addition. Throughout the process, $\tilde{G}$ is managed as a collection of planar embeddings of the biconnected components that develop as the edges are embedded. Initially, a depth first search (DFS) is performed [20] to number each vertex according to its visitation order and to distinguish a spanning tree called a *DFS tree* in each connected component. Each undirected edge in a DFS tree is called a *tree edge*, and each undirected edge not in a DFS tree is called a *back edge.* The DFS tree of each component establishes parent, child, ancestor and descendant relationships among the vertices in the component. Each vertex has a lower DFS number than its children and descendants and a higher number than its parent and ancestors, except the DFS *tree root* is a vertex with no parent or ancestors. The vertex endpoints of a DFS back edge share the ancestor-descendant relationship. A cut vertex $r$ in $\tilde{G}$ separates at least one DFS child $c$ of $r$ from the DFS ancestors (and any other DFS children) of $r$. A *virtual vertex* is an extra vertex in $\tilde{G}$ but not in $G$ that is used to represent $r$ in the separate biconnected component containing $c$. The virtual vertex is denoted $r^c$, or simply $r'$ when the child identity is unimportant. The virtual vertex $r^c$ is the *root* of the biconnected component containing $c$, which is denoted $B_{r^c}$.

For each DFS tree edge $\{v, c\}$ of $G$, a singleton biconnected component $\{v^c, c\}$ is added to $\tilde{G}$. Then, each back edge of $G$ is added to $\tilde{G}$ in an order that is partially organized into steps based on the depth first index (DFI) order of the vertices. For each vertex $v$, each back edge between $v$ and a DFS descendant of $v$ that can be added while preserving planarity is embedded in $\tilde{G}$. The DFS tree of vertices is processed in a bottom-up fashion by simply using reverse DFI order. Thus, the back edges between a vertex $v$ and its DFS ancestors are embedded in the future steps in which those ancestors are processed.

A single new back edge $\{v, w\}$ has the potential to eliminate cut vertices in $\tilde{G}$. To add a back edge $\{v, w\}$ to $\tilde{G}$, any previously separable biconnected components are first merged, then $\{v^c, w\}$ is embedded to complete the biconnection. A back edge $\{v, w\}$ is always added incident to a virtual vertex $v^c$, where $c$ is a DFS ancestor of $w$, because the back edge does not biconnect $w$ and the parent of $v$. In some future step when a back edge is added that biconnects $w$ and the parent of $v$, then $v^c$ will be merged with $v$, and the edge $\{v^c, w\}$ will become $\{v, w\}$ due to that merge operation.

A biconnected component *flip* is an operation that logically reverses the order of the adjacency lists of all vertices in a biconnected component. For efficiency, the edge addition planarity algorithm includes an optimization that defers most of the reversals until post-processing. It may be necessary to flip a biconnected component embedding during the merge operation in order to keep certain vertices on the external face boundary when the new back edge is embedded. Generally, a vertex $w$ is *active* if there is an unembedded back edge $e$ for which $w$ is either the descendant endpoint or a cut vertex in $\tilde{G}$ on the DFS tree path between the endpoints of $e$. A vertex $w$ is *inactive* if it is not active, and it can become inactive during the embedding of any single back edge. Since each back edge is embedded incident to a virtual vertex, extending into the external face region, and then incident to the descendant endpoint, the algorithm keeps all active vertices on the external face boundaries of biconnected components in $\tilde{G}$ so that a newly embedded edge does not cross an edge of an external face boundary.

A vertex $w$ is *pertinent* in step $v$ of edge addition processing if it is active due to an unembedded back edge $e$ incident to $v$. In this definition, the descendant endpoint of edge $e$ may be $w$ or a descendant $d$ of $w$, in which case $w$ has a child biconnected component rooted by a virtual vertex $w^c$, where $c$ is the root of a DFS subtree containing $d$. Similarly, a vertex $w$ is *future pertinent* in step $v$ if it is active due to an unembedded back edge $e$ incident to a DFS ancestor $u$ of $v$. Again, the descendant endpoint of edge $e$ may be $w$ or a descendant $d$ of $w$, in which case $w$ has a child biconnected component rooted by a virtual vertex $w^c$, where $c$ is the root of a DFS subtree containing $d$. A vertex is *only pertinent* if it is pertinent but not future pertinent, and it is *only future pertinent* if it is future pertinent but not pertinent.

In planarity-related algorithms, such as in [3] as well as in the $K_{3,3}$ subgraph homeomorphism algorithm of this paper, an *externally active* vertex is future pertinent. However, this paper exploits the technique of definition relaxation to enable the planarity solution to be adapted to conceptually similar outerplanarity-related problems, whereas definition relaxation is more typically used to improve algorithm efficiency. Specifically, in the edge addition outerplanarity algorithm and the associated $K_{2,3}$ and $K_4$ subgraph homeomorphism algorithms of this paper, each vertex is always *externally active*, regardless of whether or not the vertex is future pertinent, since all vertices must always remain on the external face boundaries of outerplanar embeddings. This polymorphic definition of *externally active* directly reflects the underlying difference in the graph theoretic definitions of planarity and outerplanarity. In both the planarity-related and outerplanarity-related edge addition algorithms, a *stopping vertex* is externally active but not pertinent.

The definitions related to activity and pertinence are applicable to vertices, but not to virtual vertices, which are automatically kept on the external face boundaries until they are merged with the cut vertices they represent. There are, however, analogous definitions for activity and pertinence for each whole biconnected component. A biconnected component is *active*, *pertinent* or *future pertinent* if it contains an active, pertinent or future pertinent vertex, respectively. These definitions provide underlying operational primitives for the top-level edge addition processing model presented in Figure 1.

Initialization includes such measures as performing the depth first search, sorting the vertices into DFS number order (in linear time), and calculating the least ancestor and lowpoint [20] of each vertex. The DFS tree edges can be embedded either all at once during initialization as shown in [3] or, as shown in line 5 of Figure 1, each DFS tree edge $\{v, c\}$ can be embedded as a singleton biconnected component $\{v^c, c\}$ immediately before embedding the back edges between $v$ and descendants of $c$.

```
Algorithm: Edge Addition Planarity (updated)

(1) Initialize embedding $\tilde{G}$ based on input graph $G$

(2) For each vertex $v$ from $n-1$ down to 0
(3)     Determine pertinent vertices and biconnected components, relative to $v$
(4)     For each successive DFS child $c$ of $v$
(5)         Embed tree edge $\{v, c\}$ as a singleton biconnected component $\{v^c, c\}$
(6)         Call Walkdown to embed back edges between $v^c$ and descendants of $c$
(7)         If stopping vertices blocked the embedding of a back edge, then
(8)             Isolate an obstruction and return NONEMBEDDABLE

(9) Postprocess $\tilde{G}$ and return EMBEDDABLE
```

Figure 1: The Top Level of the Edge Addition Planarity Algorithm

The edge embedding process is performed for each vertex $v$ in reverse DFI order. The back edges between $v$ and its descendants are added systematically for each child $c$ by traversing the external faces of $\{v^c, c\}$ and its descendant biconnected components. This traversal is performed by a method called Walkdown. The planarity of $\tilde{G}$ is preserved for each edge addition that the Walkdown performs, but if the Walkdown is unable to traverse to the descendant endpoint of any back edge, then the input graph is not planar [3].

When the Walkdown fails to embed a back edge, it is because it has been blocked from traversing to a pertinent vertex $w$ by stopping vertices $x$ and $y$ appearing along each of the two external face paths emanating from the root of a biconnected component. The root may either be $v^c$ or it may be a root $r'$ where $r$ is a descendant of $v$. As the Walkdown descends to various biconnected components in search of the the descendant endpoint of a back edge to embed, the roots of the biconnected components are pushed onto a mergeStack. The mergeStack is empty if the Walkdown is blocked on the biconnected component rooted by $v^c$ and otherwise the root $r'$ of the blocked biconnected component is on the top of the mergeStack.

In the core planarity algorithm, the response to a blocked Walkdown is to isolate the obstruction. Solving subgraph homeomorphism involves extending the technique to handle the cases in which the obstruction does not match the desired subgraph. Unless the desired subgraph can be found entangled with the obstruction, a reduction is performed to unblock the obstruction, thereby enabling the Walkdown to proceed.

The edge addition planarity algorithm in Figure 1 has been updated relative to [3]. The main update is the ability of the Walkdown to directly detect non-planarity (i.e., blockages). This capability is enabled in part by the successive order processing of DFS children in step 4. During the initial depth first search, each vertex is equipped with a sortedDFSChildList, which is easily computed since the children of a vertex are visited in order. Each vertex is also equipped with a sortedForwardArcList containing the list of back edges from each vertex to its descendants, sorted by the descendant endpoints. These lists can also be computed in linear time during the initial depth first search. As soon as the DFS encounters a back edge $\{v, w\}$ from a vertex $w$ to a previously numbered ancestor $v$, the back edge is added to the sortedForwardArcList of $v$. Later, during edge addition processing, when a back edge between a root copy of $v$ and a descendant $w$ is embedded, it is removed from the sortedForwardArcList of $v$. Also, unembedded back edges may be removed, rather than embedded, by the subgraph homeomorphism algorithms when a reduction is performed, in which case the back edges are also removed from the front of the sortedForwardArcList. Given these revisions, let $d$ denote the descendant endpoint of the first element of the sortedForwardArcList of $v$ or 0 if the list is empty, and let $s$ denote the next DFS child of $v$ after $c$, or $n + 1$ if $c$ is the last child of $v$. It is a property of DFS that if $c$ is a child of $v$ and $s$ is a successor child of $v$, then the descendants of $c$ have lower DFS numbers than $s$. Thus, at the end of the Walkdown for $v^c$, we can detect if it was blocked from embedding a back edge if $c < d < s$.

## 2.1 In-depth Review of the Walkdown

To embed the back edges between a vertex $v$ and its descendants in a DFS subtree rooted by child $c$, the Walkdown is invoked on a biconnected component $B_{v^c}$ rooted by the virtual vertex $v^c$. The Walkdown performs two traversals of the external face of $B_{v^c}$, corresponding to the two opposing external face paths emanating from $v^c$. The traversals perform the same operations and are terminated by the same types of conditions, so the method of traversal will only be described once.

A traversal begins at $v^c$ and proceeds in a given direction from vertex to vertex along the external face boundary in search of the descendant endpoints of back edges. Whenever a vertex is found to have a pertinent child biconnected component, the Walkdown descends to its root and proceeds with the search. Once the descendant endpoint $w$ of a back edge is found, the biconnected component roots visited along the way must be merged (and the biconnected components flipped as necessary) before the back edge $\{v^c, w\}$ is embedded. An initially empty mergeStack is used to help keep track of the biconnected component roots to which the Walkdown has descended as well as information that helps determine whether each biconnected component must be flipped when it is merged.

A Walkdown traversal terminates either when it returns to $v^c$ or when it encounters a vertex that is only future pertinent. If the Walkdown were to proceed to embed an edge after traversing beyond such a vertex, then the vertex would not remain on the external face, but it must because it is future pertinent. This is why the vertex is called a stopping vertex. By comparison, a future pertinent vertex that is also pertinent is processed because the Walkdown can descend to a pertinent child biconnected component rather than traversing past the vertex. Once the traversal visits the descendant endpoint of a back edge, the pertinent child biconnected component is merged with the future pertinent vertex without removing it from the external face.

Observe that if a child biconnected component $B_{w'}$ is only pertinent, then after its root is merged with $w$, the Walkdown traversal eventually visits the entire external face boundary of $B_{w'}$ and returns to $w$. By comparison, if $B_{w'}$ is pertinent and also future pertinent, then the Walkdown traversal encounters a stopping vertex before returning to $w$. To avoid prematurely encountering a stopping vertex, the Walkdown enforces Rule 2.1.

**Rule 2.1** *When vertex $w$ is encountered, first embed a back edge to $w$ (if needed) and then descend to all of its child biconnected components that are only pertinent (if any) before descending to a pertinent child biconnected component that is also future pertinent.*

A similar argument governs how the Walkdown chooses a direction from which to exit a biconnected component root $r^s$ to which it has descended. Both external face paths emanating from $r^s$ are searched to find the first active vertices $x$ and $y$ in each direction. The path along which traversal continues is then determined by Rule 2.2.

**Rule 2.2** *When selecting an external face path from the root $r^s$ of a biconnected component to the next vertex, preferentially select the path to a vertex that is only pertinent, if one exists, and select an external face path to a pertinent vertex otherwise.*

Finally, if both external face paths from $r^s$ lead to vertices that are only future pertinent, then both are stopping vertices and the entire Walkdown (not just the current traversal) can be immediately terminated due to a non-planarity condition. Similarly, if both traversals from $v^c$ encounter stopping vertices, and there is an unembedded back edge between $v$ and a descendant of $c$, then a non-planarity condition has occurred.

## 2.2 Example of Walkdown Processing

This section provides an illustration of the key processing rules of the Walkdown method. Figure 2(a) presents a partial embedding of a graph at the beginning of step $v$ and with the following edges still to embed: $\{u,d\}$, $\{u,s\}$, $\{u,x\}$, $\{u,y\}$, $\{v,p\}$, $\{v,q\}$, $\{v,t\}$, $\{v,x\}$, and $\{v,y\}$. Note that the vertex $i$ is inactive and the biconnected component rooted by $w'$ is not pertinent. The square vertices are future pertinent; some are only future pertinent, such as $d$ and $s$, and others are also pertinent, such as $p$, $x$ and $y$. During edge embedding, vertices such as $p$, $x$ and $y$ become only future pertinent as the edges corresponding to their pertinence are embedded.

The first Walkdown traversal begins at $v'$, proceeds counterclockwise (on the left of the edge) to $c$, then descends to $c'$. The first active vertices along the two external face paths are $x$ and $p$. Both are future pertinent and pertinent, so the decision to proceed in the direction of $x$ is made arbitrarily. At $x$, there

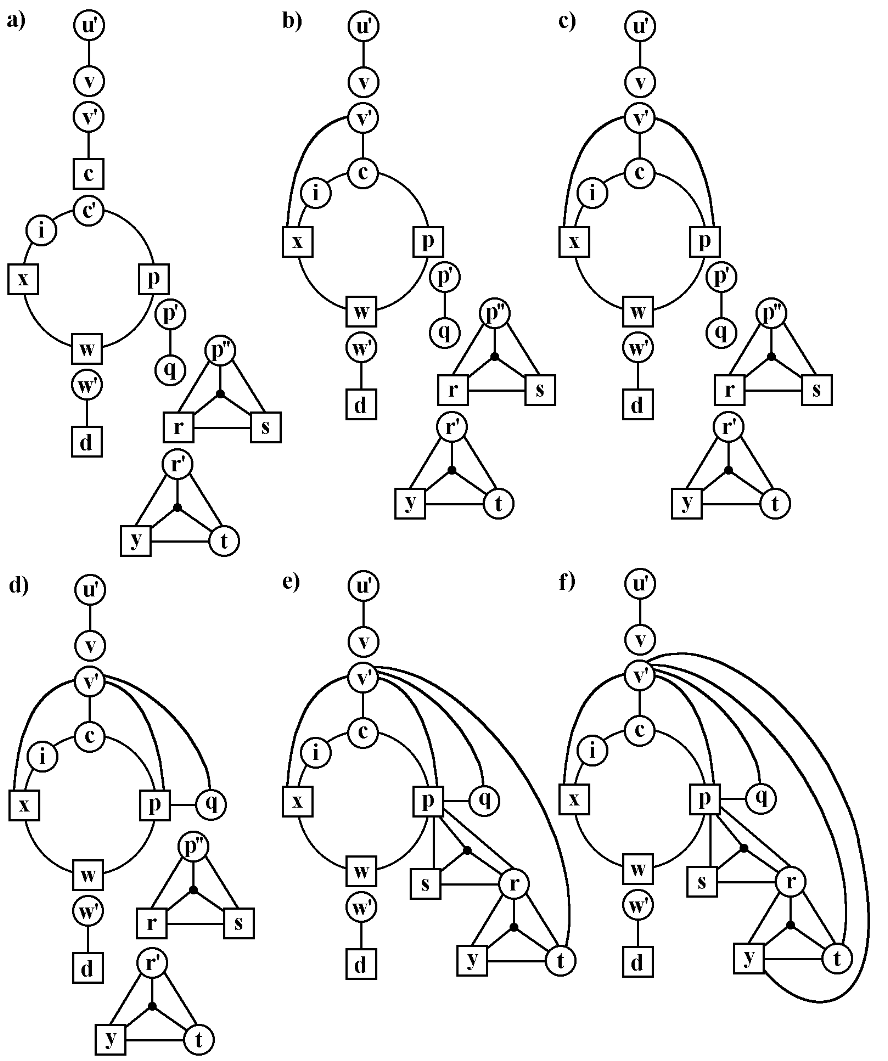

Figure 2: An Example of a Walkdown in Step $v$. Square vertices are future pertinent due to unembedded back edges $\{u, d\}$, $\{u, s\}$, $\{u, x\}$, and $\{u, y\}$. Back edges $\{v, p\}$, $\{v, q\}$, $\{v, t\}$, $\{v, x\}$, and $\{v, y\}$ are to be added in step $v$. a) Embedding at the start of step $v$. b) Merge at $c$ to add $\{v, x\}$, then stop counterclockwise traversal. c) Clockwise traversal visits $p$ and embeds $\{v, p\}$. d) Merge $p$ and $p'$ and embed $\{v, q\}$. e) Flip biconnected component rooted by $p''$, merge $p$ and $p''$, merge $r$ and $r'$, then embed $\{v, t\}$. f) Embed $\{v, y\}$ and stop clockwise traversal.

is a back edge to embed, so the Walkdown first merges $c$ and $c'$ with no flip operation since the traversal direction was consistently counterclockwise when entering $c$ and exiting $c'$. After this merge, $c$ becomes non-pertinent because it has no more pertinent child biconnected components, and it is no longer future pertinent because it has no separated DFS children with back edge connections to ancestors of $v$. Figure 2(b) shows the result of the merge and the embedding of $\{v, x\}$.

Once the back edge to $x$ has been embedded, the Walkdown determines that $x$ is a stopping vertex, so the second Walkdown traversal commences in a clockwise direction from $v'$ to $c$. In this example, $c$ became inactive in the first traversal, so the second traversal proceeds beyond $c$ to $p$.

At $p$, the back edge $\{v, p\}$ is embedded first, as shown in Figure 2(c). Then, the Walkdown descends to $p'$, rather than $p''$, because $B_{p'}$ is only pertinent. Both paths lead to $q$, which is only pertinent, so $p$ and $p'$ are merged and the back edge $\{v, q\}$ is embedded as shown in Figure 2(d). Since $q$ becomes inactive, the Walkdown proceeds to its successor on the external face, which is $p$.

In this second visitation of $p$, the Walkdown again tests whether a back edge to $p$ must be embedded, but since the back edge has already been embedded, the result is negative. The Walkdown again tests for pertinent child biconnected components, but this time there are none which are only pertinent, so the Walkdown descends to $p''$. The two external face paths from $p''$ lead to future pertinent vertices $r$ and $s$, but $r$ is pertinent and $s$ is not, so the Walkdown selects the counterclockwise direction from $p''$ to $r$. This is contrary to the clockwise direction by which the Walkdown entered $p$, so the indication of a flip operation is pushed onto the mergeStack, along with $p''$.

The Walkdown proceeds to $r$, where it finds $r$ has no back edge to embed, but $r$ does have a pertinent child biconnected component, so the Walkdown descends to $r'$. The two external face paths from $r'$ lead to $y$ and $t$. While $y$ is pertinent, it is also future pertinent, whereas $t$ is only pertinent. The clockwise path to $t$ is selected, in opposition to the counterclockwise direction used to enter $r$. Thus, $r'$ and a flip indicator are pushed onto the mergeStack.

At $t$, the Walkdown determines that a back edge must be embedded. First, the mergeStack is processed. $B_{r'}$ is flipped, and $r'$ is merged with $r$. Then, $p''$ is popped and the component comprised of $B_{p''}$ merged with $B_{r'}$ is flipped. Finally, the back edge $\{v, t\}$ is embedded. Notice that $B_{r'}$ is logically flipped a second time, restoring its original orientation. All such double flips are effectively eliminated using an efficient implementation technique described in [3]. The logical result of these operations is shown in Figure 2(e).

The clockwise traversal then continues from vertex $t$, which is now inactive, to vertex $y$. The back edge $\{v, y\}$ is embedded as shown in Figure 2(f). Once the back edge to $y$ is embedded, $y$ is no longer pertinent since it has no pertinent child biconnected components. Thus, $y$ is a stopping vertex that terminates the second, clockwise traversal of the Walkdown.

## 3 Outerplanarity

As a first step toward subgraph homeomorphism for $K_{2,3}$ and $K_4$, the edge addition planarity algorithm is adjusted to form the Edge Addition Outerplanarity Algorithm. It is well-known that a planarity algorithm can be extended to an outerplanarity algorithm by first adding a special vertex incident to all vertices of the input graph, then performing the planarity algorithm, and then removing the special vertex and its incident edges. Rather than going through these explicit steps, the edge addition planarity algorithm is adjusted to achieve the same effect by classifying all vertices to be always externally active. As mentioned in Section 2, this change affects the meaning of stopping vertex but not future pertinence. This alteration ensures that edges are only added as long as all vertices can be kept on the external face of the embedding. Specifically, the Walkdown can process the pertinence of a vertex, including descending to a pertinent child biconnected component and selecting a direction, but it is blocked from traversing past a vertex once its pertinence is resolved.

If the Walkdown is blocked with a non-empty mergeStack, then edge contraction and deletion can be used on $\tilde{G}$, along with the addition of certain unembedded edges represented in $\tilde{G}$ as vertex activity, to produce the non-outerplanarity minor A in Figure 3(a), which indicates a $K_{2,3}$ homeomorph.

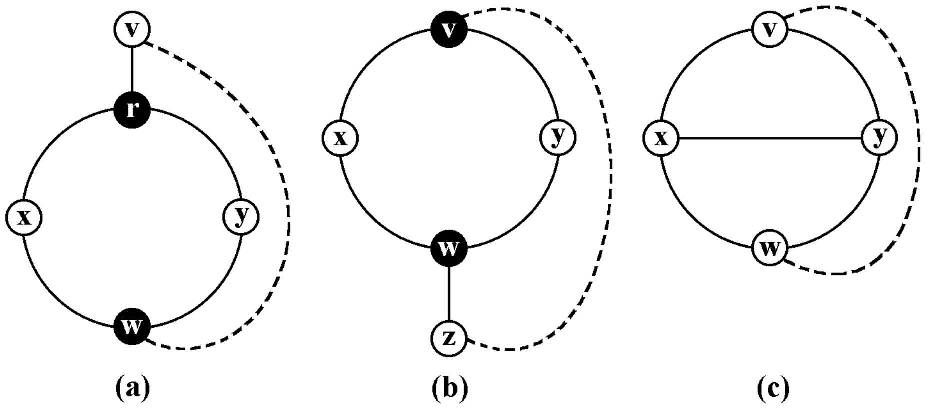

Figure 3: (a) Non-Outerplanarity Minor A signals a $K_{2,3}$ homeomorph, (b) Non-Outerplanarity Minor B signals a $K_{2,3}$ homeomorph, (c) Non-Outerplanarity Minor E signals a $K_4$ homeomorph.

If the Walkdown is blocked with an empty mergeStack, there are two more non-outerplanarity cases. For both, consider the two external face neighbors of $v^c$, denoted $x$ and $y$. There is an external face path from $x$ to $y$ that contains $v^c$ and a second external face path $P$ from $x$ to $y$ that excludes $v^c$. $P$ contains a pertinent vertex $w$. If $w$ has a pertinent child biconnected component, then $G$ is not outerplanar due to non-outerplanarity minor B in Figure 3(b), which also indicates a $K_{2,3}$ homeomorph.

Now suppose a pertinent vertex $w$ along path $P$ has no pertinent child biconnected component, in which case there is an unembedded back edge $\{v^c, w\}$. Let the term $x$-$y$ path refer to an edge[1] embedded inside the biconnected component rooted by $v^c$ with one endpoint incident to a vertex $p_x$ internal to the path $(v^c, \ldots, x, \ldots, w)$ and the other endpoint incident to a vertex $p_y$ internal to the path $(v^c, \ldots, y, \ldots, w)$. An $x$-$y$ path separates the region inside the biconnected component's external face cycle such that the edge $\{v^c, w\}$ cannot be embedded in the internal region. Suppose there is no $x$-$y$ path for $w$. Due to processing Rules 2.1 and 2.2, this contradicts the supposition that $\{v^c, w\}$ is unembedded since the Walkdown would then have no way to avoid visiting $w$ prior to merging $w$, $x$ and $y$ into the same biconnected component. Hence, $w$ must have an $x$-$y$ path if it has no pertinent child biconnected component.

If there is an $x$-$y$ path associated with $w$, then the input graph $G$ is not outerplanar due to non-outerplanarity minor E[2] in Figure 3(c), which indicates a $K_4$ homeomorph with image vertices $v$, $p_x$, $w$, and $p_y$.

The three non-outerplanarity minors in Figure 3 form a characterization of outerplanarity, i.e. they form an unavoidable set for non-outerplanar graphs and an avoidable set for outerplanar graphs. The reasoning is analogous to that appearing in [3] for the non-planarity minors associated with the core edge addition planarity algorithm. To summarize, when the Walkdown adds all back edges for a biconnected component $B_{v^c}$ rooted by $v^c$, then the result is an outerplanar embedding of $B_{v^c}$. When the Walkdown fails to embed a back edge for $B_{v^c}$, then the mergeStack is either non-empty, which results in a $K_{2,3}$ homeomorph (Figure 3(a)), or it is empty. If the mergeStack is empty, then $B_{v^c}$ contains a pertinent vertex $w$ that either has a pertinent child biconnected component, which results in a $K_{2,3}$ homeomorph (Figure 3(b)), or it does not. In this case, we reach the crucial contradiction described above unless $w$ has a separating $x$-$y$ path, which results in a $K_4$ homeomorph (Figure 3(c)). Thus, when the Walkdown fails to embed a back edge in $B_{v^c}$, the input graph $G$ is not outerplanar and an outerplanar obstruction is isolated. By the arguments above, we have the following theorem:

**Theorem 1** *Given a graph $G$, the Edge Addition Outerplanarity Algorithm determines an outerplanar embedding if $G$ is outerplanar or a subgraph homeomorphic to $K_{2,3}$ or $K_4$ if $G$ is not outerplanar in $O(n)$ time.*

## 4 Search for $K_{2,3}$ Homeomorphs

The Edge Addition $K_{2,3}$ Search algorithm extends the outerplanarity algorithm of Section 3 with further examinations of the graph if non-outerplanarity minor

[1] The notion of $x$-$y$ path comes from the planarity-related algorithms, but it is a single edge in outerplanarity-related algorithms because no vertices are embedded within the regions surrounded by the external face bounding cycles of biconnected components.

[2] This non-outerplanarity minor is labeled E because it is analogous to non-planarity minor E. The non-planarity minors C and D (Figure 9(c,d)) have no analogous non-outerplanarity minors because the conditions they characterize cannot occur in the outerplanarity algorithm.

E is found since it indicates an obstructing $K_4$ homeomorph. This section proves that either there is additional graph structure necessary to find a $K_{2,3}$ homeomorph entangled with the $K_4$ homeomorph, or the non-outerplanarity minor E is a $K_4$ separable by a cut vertex. Due to the separability of the $K_4$, the $K_{2,3}$ search algorithm unblocks the Walkdown, enabling the search to proceed elsewhere in the graph as if the $K_4$ had not obstructed outerplanarity.

When the Walkdown is blocked on biconnected component $B_{v^c}$, if it is blocked by non-outerplanarity minor A or B, then a subgraph homeomorphic to $K_{2,3}$ is indicated. If the Walkdown is blocked by non-outerplanarity minor E, then the $x$-$y$ path is obtained, and additional analyses are performed corresponding to the diagrams in Figure 4.

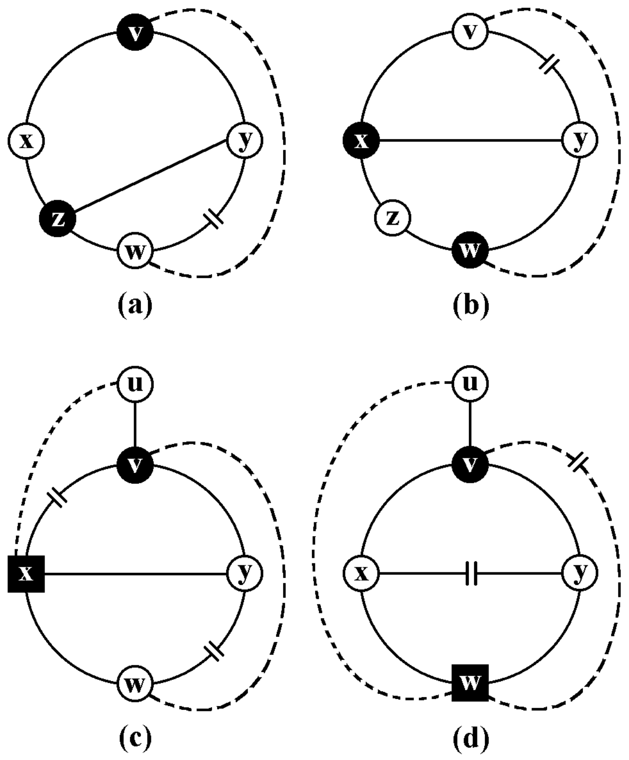

Figure 4: $K_{2,3}$ Homeomorphs from Non-Outerplanarity Minor E. (a) the $x$-$y$ path has a point of attachment not equal to $x$ or $y$ , (b) the external face path ($x$, ..., $w$, ..., $y$) contains an extra vertex $z$, (c) $x$ or $y$ is future pertinent, (d) $w$ is future pertinent.

If either point of attachment of the $x$-$y$ path is attached below $x$ or $y$ (i.e. if $p_x \neq x$ or $p_y \neq y$), then a $K_{2,3}$ homeomorph can by isolated according to Figure 4(a) (note the simple variation cases of having $p_x = z \neq x$, $p_y = z \neq y$, and having both $p_x \neq x$ and $p_y \neq y$). Hence, consider the case in which the $x$-$y$

path (a single edge) is attached directly to $x$ and $y$. If any other vertex $z$ is on the lower external face path ($x$, …, $w$, …, $y$), then a $K_{2,3}$ homeomorph can be isolated according to Figure 4(b) (note the symmetric case of having $z$ along the lower external face path between $w$ and $y$).

Thus, $w$ must be the only vertex along the lower external face path, and the connection from $w$ to $v$ must be an edge (otherwise, non-outerplanarity minor B). From above, we also know that $x$ and $y$ are direct neighbors of each other, and by construction of the outerplanarity algorithm we know that $x$ and $y$ are direct neighbors of $v$. Thus, the vertices $v$, $x$, $y$ and $w$ form a $K_4$.

If $x$ or $y$ can be connected to an ancestor $u$ of $v$ by zero or more separated biconnected components plus an unembedded back edge, then a $K_{2,3}$ homeomorph can be isolated according to Figure 4(c) (note symmetric case of a connection from $y$ to $u$). If $w$ can be connected to an ancestor $u$ of $v$ by zero or more separated biconnected components plus an unembedded back edge, then a $K_{2,3}$ homeomorph can be isolated according to Figure 4(d). The absence of the conditions corresponding to Figures 4(c) and 4(d) imply that neither $w$, $x$ nor $y$ connect to ancestors of $v$, except through $v$. Thus, the $K_4$ formed by $v$, $x$, $y$ and $w$ is separable in $G$ by the cut vertex $v$.

As a result, there are not enough paths available for any of $w$, $x$, and $y$ to be in any $K_{2,3}$ homeomorph that may be in $G$, so the Edge Addition $K_{2,3}$ Homeomorph Search can therefore proceed with outerplanarity testing as if $x$, $y$ and $w$ were not in $G$. Such further outerplanarity testing will either find another non-outerplanarity minor and repeat the logic above, or it will not.

If the conditions for a $K_{2,3}$ homeomorph are found by the above steps, then the top-level algorithm of Figure 1 isolates it and returns NONEMBEDDABLE. Otherwise, no post-processing of $\tilde{G}$ is required, and EMBEDDABLE is returned to indicate that $G$ has no subgraph homeomorphic to $K_{2,3}$. By the arguments above, we have the following theorem:

**Theorem 2** *Given an input graph $G$, the Edge Addition $K_{2,3}$ Homeomorph Search algorithm finds a subgraph homeomorphic to $K_{2,3}$ in $G$ or determines that $G$ is $K_{2,3}$-less in $O(n)$ time.*

## 5 Search for $K_4$ Homeomorphs

The Edge Addition $K_4$ Search algorithm extends the outerplanarity algorithm of Section 3 with further examinations of the graph if non-outerplanarity minors A or B are found since they indicate an obstructing $K_{2,3}$ homeomorph. This section proves that either there is additional graph structure necessary to find a $K_4$ homeomorph entangled with the $K_{2,3}$ homeomorph, or the non-outerplanarity minor A or B can be reduced to unblock the Walkdown, enabling the search to proceed as if the $K_{2,3}$ had not obstructed outerplanarity.

## 5.1 Handling Non-Outerplanarity Minor A

When the Walkdown is blocked by non-outerplanarity minor A, the root $r'$ of the blocked biconnected component $B_{r'}$ is on top of the mergeStack. The non-pertinent vertices $x$ and $y$ along both external face paths emanating from $r'$ prevent traversal to the pertinent vertex $w$.[3] Although non-outerplanarity minor A is associated with a $K_{2,3}$ homeomorph, there are two cases below in which it can also be associated with a $K_4$ homeomorph. Otherwise, $B_{r'}$ is unblocked by reducing it to a single edge.

Case A1: A $K_4$ homeomorph can be obtained from non-outerplanarity minor A if any vertex $z$ other than $w$ in $B_{r'}$ is pertinent or future pertinent. In Figure 5(a), vertex $z$ is pertinent and it is depicted as distinct from $x$. In Figure 5(b), vertex $z$ is future pertinent and depicted being equal to $x$. The difference of pertinence and future pertinence only affects how $z$ connects to $v$. Similarly, being distinct from or equal to $x$ has no important effect on how $z$ connects to $r'$ and $w$. Furthermore, the case of $z$ being along the external face path ($r'$, ..., $y$, ..., $w$) is symmetric with the cases depicted in Figure 5. A $K_4$ homeomorph can be isolated with image vertices $v$, $r$, $w$ and $z$.

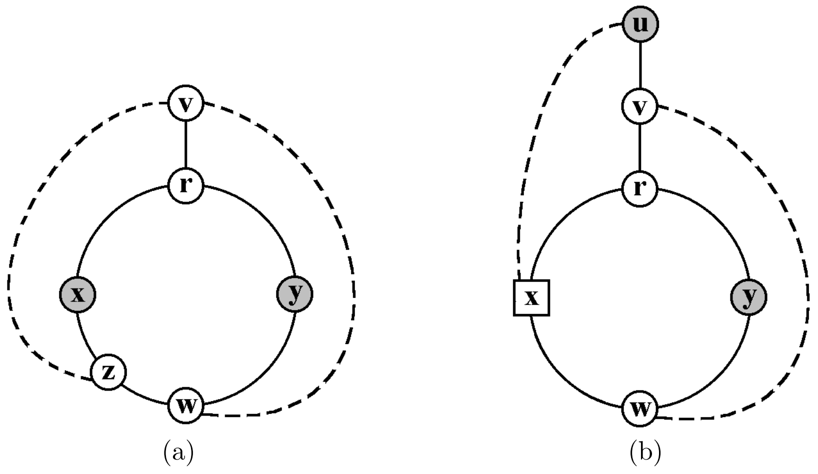

Figure 5: Illustrations of Case A1: A $K_4$ homeomorph entangled with the $K_{2,3}$ of non-outerplanarity minor A based on a second pertinent or future pertinent vertex $z$. (a) Shows $z$ being pertinent and distinct from $x$ and $y$. (b) Shows $z$ being future pertinent and equal to $x$. All other cases are symmetric.

Case A2: A $K_4$ homeomorph can be obtained from non-outerplanarity minor A if the biconnected component $B_{r'}$ contains an edge $e$ connecting a vertex along $r \ldots x \ldots w$ and a vertex along $r \ldots y \ldots w$, excluding $r$ and $w$. The edge $e$ is known as an $x$-$y$ path, but the path has no internal vertices because $B_{r'}$ is outerplanar. Figure 6 depicts the condition described by this case. The

[3]If $x$ or $y$ were pertinent, then the Walkdown would not have been blocked; instead, $B_{r'}$ would have been merged into $B_{v'}$ as part of resolving the pertinence of $x$ or $y$.

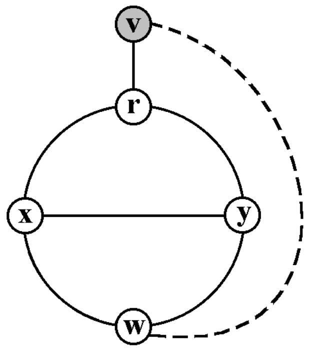

Figure 6: Illustration of Case A2: A $K_4$ homeomorph entangled with non-outerplanarity minor A based on an $x$-$y$ path appearing within $B_{r'}$. The path is a single edge due to outerplanarity, and its points of attachment may not be equal to $x$ and $y$ due to the use of edge contraction in the diagram.

endpoints $p_x$ and $p_y$ of the $x$-$y$ path need not be equal to $x$ and $y$ since edge contraction was used to depict $e$ incident to $x$ and $y$ while still bisecting the internal region. Thus, $r'$ and $w$ are on the bounding cycles of separate internal regions of $B_{r'}$, and a $K_4$ homeomorph can be isolated with image vertices $r$, $w$, $p_x$ and $p_y$.

If cases A1 and A2 are not applicable to non-outerplanarity minor A, then $B_{r'}$ can be reduced to the edge $\{r', w\}$. If case A1 does not occur, then $w$ is the only pertinent vertex in $B_{r'}$. Hence, the subgraph induced by the vertices of $B_{r'}$ is separable from the input graph by the 2-cut $\{r, w\}$. If case A2 also does not occur, then $B_{r'}$ also lacks the internal structure to contribute more than a single path to a $K_4$ homeomorph. Thus, $B_{r'}$ can be reduced to the edge $\{r', w\}$ to remove the obstruction to outerplanarity while preserving the essential structure of any $K_4$ homeomorph that may be elsewhere in the input graph.

## 5.2 Handling Non-Outerplanarity Minor B

When the Walkdown is blocked by non-outerplanarity minor B, the external face paths emanating from the root $v'$ of a biconnected component $B_{v'}$ are blocked by non-pertinent vertices $x$ and $y$, preventing the Walkdown from reaching a vertex $w$ that has a pertinent child biconnected component.

The first active vertices, $a_x$ and $a_y$, along each external face path emanating from $v'$ are obtained. Since $x$ and $y$ stopped the outerplanarity Walkdown, they are not pertinent, but they may not be future pertinent (i.e. $x$ and $y$ blocked the WalkDown simply because all vertices are externally active). Thus, $a_x$ and $a_y$ may not equal $x$ and $y$.

Case B1: A $K_4$ homeomorph can be obtained from non-outerplanarity minor B if the vertices $a_x$ and $a_y$ are distinct and future pertinent. Figure 7 depicts this condition. A $K_4$ homeomorph can be isolated based on the external face bounding cycle of $B_{v'}$, the future pertinence paths of $a_x$ and $a_y$, and the DFS tree path from $v$ to the minimum numbered ancestor $u$ associated with the future pertinence of $a_x$ and $a_y$. The degree 3 image vertices are $a_x$, $a_y$, $v$ and the maximum numbered DFS ancestor associated with the future pertinence of $a_x$ and $a_y$.[4]

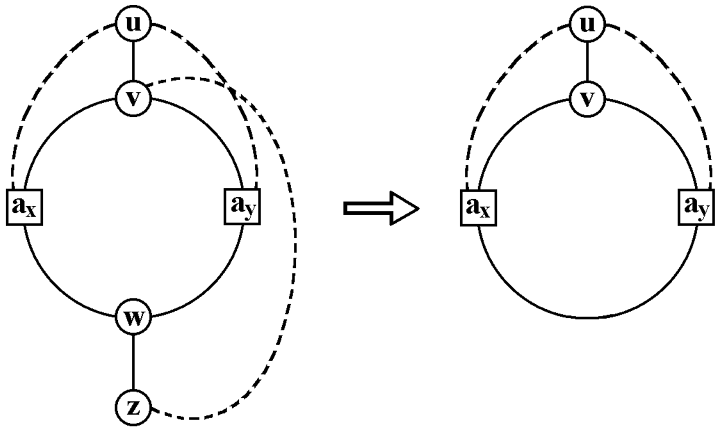

Figure 7: Illustration of Case B1: Finding a $K_4$ homeomorph entangled in non-outerplanarity minor B based on the future pertinence and inequality of the first active vertices $a_x$ and $a_y$ along the external face paths emanating from the root of $B_{v'}$. Vertex $w$ may be distinct, or it may equal $a_x$ or $a_y$.

Case B2: A $K_4$ homeomorph can be obtained from non-outerplanarity minor B if either $a_x$ or $a_y$ has an $x$-$y$ path. The vertex, $a_x$ or $a_y$, can be pertinent or future pertinent. Figure 8 depicts these conditions. If the vertex is pertinent, as depicted for $a_x$ in Figure 8(a), then a $K_4$ homeomorph can be isolated, which can be seen by replacing the label $a_x$ with $w$ to recognize non-outerplanarity minor E. Hence, consider the case in which the vertex $a_x$ or $a_y$ is future pertinent and has a separating $x$-$y$ path. In Figure 8(b), vertex $a_y$ is shown to be future pertinent, which is shown to have no significant difference from pertinence with regard to obtaining a $K_4$ homeomorph.[5]

If the conditions of cases B1 and B2 are not applicable to non-outerplanarity minor B, then the paths $v' \dots a_x$ and $v' \dots a_y$ can be reduced to the edges $\{v', a_x\}$ and $\{v', a_y\}$, respectively. Specifically, since the vertices $a_x$ and $a_y$ are the first active vertices along the external face paths $v' \dots a_x$ and $v' \dots a_y$, no preceding vertices closer to $v'$ along those paths have unembedded back edges or

[4]This $K_4$ homeomorph includes $w$ but not the pertinent path from $w$ to $v$. A $K_4$ homeomorph that includes the pertinent path could also be isolated, but special cases arise such as when $w$ is equal to $a_x$ or $a_y$. The isolation method depicted in Figure 7 avoids special cases.

[5]The $K_4$ homeomorph corresponding to Figure 8(b) is not based on pertinence in step $v$. This case could be ignored since it would be found in step $u$ via case A2. However, the path $(v' \dots a_y)$ has been explored but cannot be reduced to an edge due to the $x$-$y$ path attached to it, so a $K_4$ homeomorph must be isolated in order to maintain linear time performance.

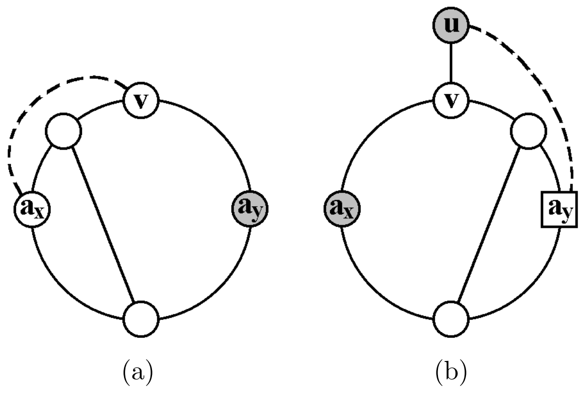

Figure 8: Illustration of Case B2: Finding a $K_4$ homeomorph entangled in non-outerplanarity minor B based on an $x$-$y$ path for $a_x$ or $a_y$. (a) The active vertex is pertinent, (b) The active vertex is future pertinent.

separate child biconnected connected components that connect to $v$ or ancestors of $v$. Furthermore, neither $a_x$ nor $a_y$ has an $x$-$y$ path. Since the biconnected component $B_{v'}$ is an outerplanar embedding, any additional structure attached to these paths consists of individual edges that are parallel to the paths. Thus, the subgraphs induced by the vertices in the paths $v' \dots a_x$ and $v' \dots a_y$ are 2-cut separable components that lack the structure needed to contribute more than a single path to any $K_4$ homeomorph in the graph. The paths can therefore be reduced to the single edges $\{v', a_x\}$ and $\{v', a_y\}$.

The purpose of the reductions of case B2 is to remove vertices from the external face so that the Walkdown can continue to explore the pertinent subgraph attached to $B_{v'}$. This is guaranteed if the condition of case B1 are not met, since then at least one of $a_x$ or $a_y$ is pertinent. The vertices $a_x$ and $a_y$ were selected due to being active, so each is either pertinent or future pertinent. The condition of case B1 is that the vertices are distinct and both are future pertinent, so failing that condition means either they are not distinct or are not both future pertinent. If $a_x$ and $a_y$ are distinct, then at least one is not future pertinent, but it is active so it must be pertinent. If $a_x$ and $a_y$ are equal, then $a_x$ is the only active vertex on the external face of the biconnected component $B_{v'}$. Since $B_{v'}$ was selected for non-outerplanarity processing, it is pertinent, so $a_x$ (equivalently $a_y$) must be pertinent, whether or not it is also future pertinent.

## 5.3 Coda

When the Walkdown on $v^c$ becomes blocked due to non-outerplanarity minor E, then a subgraph homeomorphic to $K_4$ is indicated. If the Walkdown is blocked by non-outerplanarity minor A or B, then additional analyses are performed as described in Sections 5.1 and 5.2. If the conditions for a $K_4$ homeomorph are found by the above steps, then the planarity algorithm of Figure 1 isolates it and returns NONEMBEDDABLE. Otherwise, a reduction is performed to unblock the biconnected component, and the Walkdown is able to continue to the next pertinent vertex. The processing leading up to a reduction only need perform work over the portion(s) of the graph being reduced. Proceeding with the Walkdown may result in further occurrences of non-outerplanarity minors A or B may be encountered, which are handled in the manner described above, or an occurrence of non-outerplanarity minor E, which results in a desired $K_4$ homeomorph. Thus, either a $K_4$ homeomorph is found attached to $B_{v'}$ or the pertinence of all vertices in $B_{v'}$ is resolved. If this occurs throughout the graph, then the planarity algorithm of Figure 1 performs no post-processing of $\tilde{G}$, and EMBEDDABLE is returned to indicate that $G$ has no subgraph homeomorphic to $K_4$. By the arguments above, we have the following theorem:

**Theorem 3** *Given an input graph $G$, the Edge Addition $K_4$ Homeomorph Search algorithm finds a subgraph homeomorphic to $K_4$ in $G$ or determines that $G$ is $K_4$-less in $O(n)$ time.*

## 6 Search for $K_{3,3}$ Homeomorphs

The edge addition planarity algorithm characterizes planarity based on the five graph minors in Figure 9. A subgraph homeomorphic to $K_{3,3}$ can be obtained based on non-planarity minors A, B, C and D. Non-planarity minor E is a $K_5$ minor, from which a subgraph homeomorphic to $K_{3,3}$ can be obtained based on four additional non-planarity minors in Figure 10. Absent the conditions corresponding to the patterns in Figure 10, the edge addition planarity algorithm isolates a subgraph homeomorphic to $K_5$. [3]

The Edge Addition $K_{3,3}$ Search algorithm extends the edge addition planarity algorithm with additional test cases to perform in lieu of isolating a subgraph homeomorphic to $K_5$. Either the additional conditions exist that enable a subgraph homeomorphic to $K_{3,3}$ to be isolated or the pertinent subgraph attached to the blocking biconnected component $B$ is discarded, which eliminates the local obstruction to planarity and allows the planarity algorithm to continue searching for a subgraph homeomorphic to $K_{3,3}$.

The following terms and notation aid the presentation of the main result. Let $u_w$, $u_x$, and $u_y$ denote ancestor endpoints of future pertinent connections from $w$, $x$, and $y$, respectively, to ancestors of $v$. Initially, $u_w$, $u_x$, and $u_y$ have the least depth first index (DFI) from among all the future pertinent connections of $w$, $x$, and $y$, respectively, except in cases 2 and 3 below, which test for future pertinent connections with endpoints closer to $v$ (i.e. with higher numbered DFIs). Let $u_{min}$ and $u_{max}$ denote the minimum DFI and maximum DFI, respectively, of $u_w$, $u_x$, and $u_y$. A *piece* $\Pi$ of a graph $G$ with respect to a subgraph $H$ is either an edge in $G - H$ whose endpoints are in $H$ or a connected component of $G - H$ plus the edges of $G$ having one endpoint in $G - H$ and the other in $H$. An *attachment point* of $\Pi$ to $H$ is a vertex of $H$ incident with an edge of $\Pi$. A *bridge* of a graph $G$ with respect to a subgraph

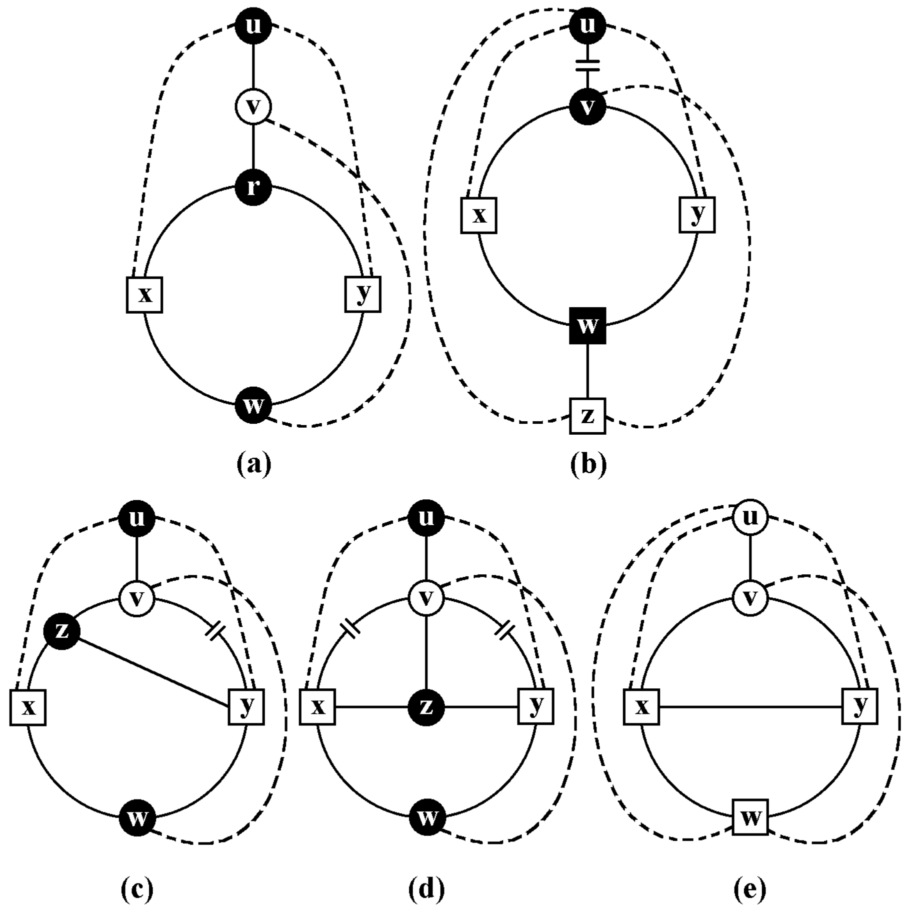

Figure 9: Edge Addition Non-planarity Minors A, B, C, D and E

$H$ is a piece of $G$ with respect to $H$ that has more than one attachment point to $H$. Let $B$ denote the blocked biconnected component containing $v^c$. Let $T_c$ denote the DFS subtree with root $c$, and let $P$ denote the DFS path $(u_{min}, \ldots, v)$. Consider the bridges of the input graph $G$ with respect to path $P$. The $T_c$-bridge is the bridge of $G$ with respect to $P$ that contains the vertices in $T_c$. Let $\beta_P$ denote the set of all bridges of $G$ with respect to $P$ except the $T_c$-bridge. A bridge in $\beta_P$ *straddles* the vertex $u_{max}$ if the bridge attaches to a descendant of $u_{max}$ in $P$ and to an ancestor of $u_{max}$ (see Figure 12).

Based on these definitions, Theorem 4 proves the sufficiency of testing only the seven cases below to determine whether a $K_{3,3}$ homeomorph is entangled in the $K_5$ homeomorph found by the edge addition planarity algorithm:

Case 1: If there is any pertinent or future pertinent vertex other than $w$, $x$ and $y$ along the external face path $(x, \ldots, w, \ldots, y)$ of $B$, then a $K_{3,3}$ homeomorph can be isolated by non-planarity minor $E_1$ from [3] (see Figure 10(a)).

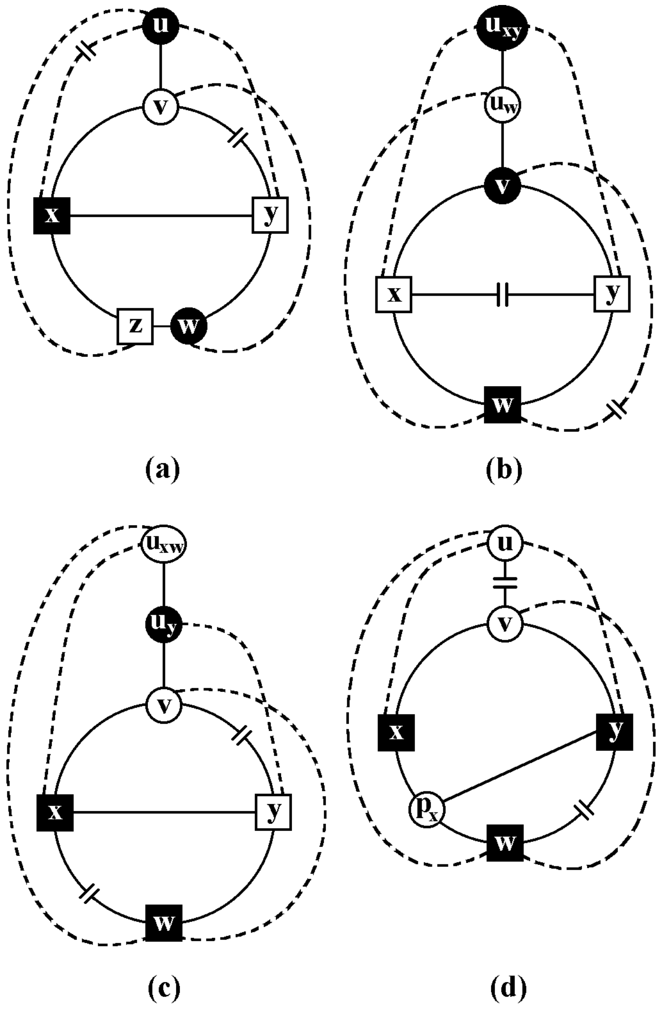

Figure 10: Additional $K_{3,3}$ Minors from the $K_5$ Minor E in Figure 9. (a) Minor $E_1$, (b) Minor $E_2$, (c) Minor $E_3$, (d) Minor $E_4$

Note that $w$ is pertinent and future pertinent in the $K_5$ minor pattern. If the vertex $z$ is future pertinent, then only the pertinence of $w$ is needed, as shown in see Figure 10(a). If $z$ is pertinent but not future pertinent, then swap $w$ and $z$ in order to obtain a $K_{3,3}$ homemorph according to Figure 10(a).

Case 2: If $w$ or any of its descendants in separate biconnected components can connect by a back edge to an ancestor of $v$ that is descendant to $u_x$ and $u_y$, then a $K_{3,3}$ homeomorph can be isolated by non-planarity minor E$_2$ from [3] (see Figure 10(b)).

Case 3: A $K_{3,3}$ homeomorph can be isolated by non-planarity minor E$_3$

from [3] (see Figure 10(c)) if $x$ or its descendants in separate biconnected components can connect by a back edge to an ancestor of $v$ that is descendant to $u_y$ and $u_w$, or symmetrically if $y$ or its descendants in separate biconnected components can connect by a back edge to an ancestor of $v$ that is descendant to $u_x$ and $u_w$.

Case 4: If there exists an $x$-$y$ path in $B$ with a point of attachment $p_x \neq x$, then a $K_{3,3}$ homeomorph can be isolated by non-planarity minor $E_4$ from [3] (see Figure 10(d)). The condition $p_y \neq y$ is symmetric.

Case 5: If the $x$-$y$ path in $B$ contains a single endpoint $z$ of a second path $p$ of the form $(w, \ldots, z)$, where all vertices in the path (except $w$) are embedded inside $B$, then a $K_{3,3}$ homeomorph can be isolated by the non-planarity minor $E_5$ (see Figure 11(a)), which is symmetric to non-planarity minor D (see Figure 9(d)). The paths represented by edges $\{u, v\}$, $\{x, w\}$ and $\{w, y\}$ are not needed in the $K_{3,3}$ homeomorph.

Case 6: If $u_w < u_{max}$ and there exists a bridge in $\beta_P$ that straddles $u_{max}$, then a $K_{3,3}$ homeomorph can be isolated by non-planarity minor $\mathrm{E}_6$ (see Figure 11(b)), which uses a path from the straddling bridge represented by edge $\{u_w, v\}$ and omits the paths corresponding to the edges $\{u_{xy}, v\}$, $\{x,\ y\}$ and $\{v,\ w\}$.

Case 7: If $u_y < u_{max}$ and there exists a bridge in $\beta_P$ that straddles $u_{max}$, then a $K_{3,3}$ homeomorph can be isolated by non-planarity minor $\mathrm{E}_7$ (see Figure 11(c)). Note the symmetric case for $u_x < u_{max}$. The paths in the graph corresponding to the edges $\{v, y\}$, $\{x, w\}$ and $\{u_{wx}, v\}$, excluding the endpoints, are not needed to form the $K_{3,3}$ homeomorph. In the symmetric case (in which $u_x < u_{max}$), the edges to omit are $\{v, x\}$, $\{y, w\}$ and $\{u_{wy}, v\}$).

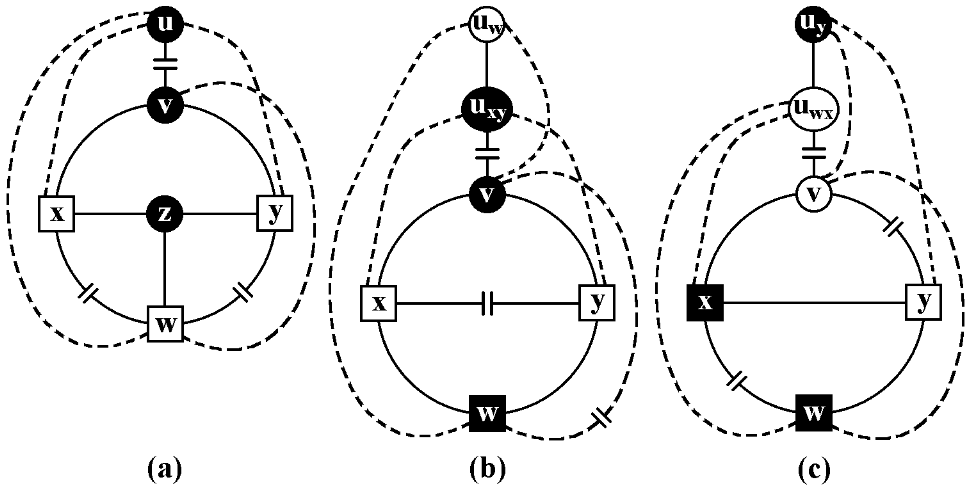

Figure 11: Minor characterizations of additional $K_{3,3}$ homeomorphs that can be extracted despite meeting minimal conditions for isolating a $K_5$ homeomorph. (a) Minor $\mathrm{E}_5$, (b) Minor $\mathrm{E}_6$, (c) Minor $\mathrm{E}_7$

**Theorem 4** *Given an input graph $G$, the Edge Addition $K_{3,3}$ Homeomorph Search algorithm finds a subgraph homeomorphic to $K_{3,3}$ in $G$ or determines that $G$ is $K_{3,3}$-less.*

**Proof:** If $G$ is planar, then it is $K_{3,3}$-less, and a planar embedding is created without encountering a non-planarity condition and therefore without invoking any augmentations of the $K_{3,3}$ homeomorph search algorithm. Hence, assume $G$ is non-planar. If the edge addition Kuratowski subgraph isolator identifies a $K_{3,3}$ homeomorph, or if a $K_{3,3}$ homeomorph is identified by one of the 7 cases above, then it is isolated and returned. Some of the edges of the $K_{3,3}$ homeomorph may be virtual edges arising from prior reductions of $K_5$ homeomorphs, but any such reduction edge is replaced with a path from the component it reduced. Hence, the key to correctness is ensuring that $K_5$ homeomorph reductions preserve any existing $K_{3,3}$ homeomorph in $G$. If the edge addition Kuratowski subgraph isolator identifies a $K_5$ homeomorph, and none of the above cases 1 to 7 find a $K_{3,3}$ homeomorph, then we prove that the $K_5$ homeomorph can be planarized, enabling the search for a $K_{3,3}$ homeomorph to proceed elsewhere in $G$.

Due to the failure to find minor C, the planar subgraphs represented by edges $\{v, x\}$ and $\{v, y\}$ are separable from biconnected component $B$ by 2-cuts $\{v, x\}$ and $\{v, y\}$. The failure to find minors $\text{E}_1$ and $\text{E}_4$ implies that edges $\{x, w\}$ and $\{w, y\}$ represent planar subgraphs separable from $B$ by 2-cuts $\{x, w\}$ and $\{w, y\}$. The failure to find minors C, D, $\text{E}_4$ and $\text{E}_5$ implies that edge $\{x, y\}$ represents a planar subgraph separable from $B$ by the 2-cut $\{x, y\}$. The failure to find minor B implies that edge $\{v, w\}$ represents a planar subgraph $H$ that is separable from $B$ by the 2-cut $\{v, w\}$. Although $H$ includes one or more edges not yet embedded incident to $v$ plus planar embeddings of any pertinent descendant biconnected components of $B$, the roots of the descendant biconnected components and the endpoints of the unembedded edges are on external face boundaries, so $H + \{v, w\}$ is planar, hence $K_{3,3}$-less. Similarly, each other component of $B$ represented by an edge above is a planar embedding that remains planar, hence $K_{3,3}$-less, when an edge between the 2-cut endpoints is added. Thus, $B$ plus the unembedded edges can be reduced to a $K_4$ on $v$, $x$, $y$ and $w$. The rest of the proof consists of showing that $B$ can be reduced to $K_4$ minus the edge $\{v, w\}$ while preserving every $K_{3,3}$ homeomorph in $G$. Since $H$ is separable from $G$ by the 2-cut $\{v, w\}$, we need only focus on $K_{3,3}$ homeomorphs in $G$ that would be required to contain a single path in $H$ to join $v$ and $w$. We show that the path can be obtained from $B - (H - v - w)$ instead.

Let $\beta_{P+B}$ denote the set of all bridges of $G - (H - v - w)$ with respect to the subgraph induced by the vertices of $B$ plus the path $P = (u_{min}, \dots, v)$. Let $\beta_x$, $\beta_y$ and $\beta_w$ denote the subsets of bridges in $\beta_{P+B}$ whose attachment points include $x$, $y$ or $w$, respectively. As Figure 12 shows, $\beta_{P+B} = \beta_P \cup \beta_x \cup \beta_y \cup \beta_w$. Note that $\beta_P$ does not include $\beta_x$, $\beta_y$ and $\beta_w$ because they are part of the $T_c$-bridge, which is excluded from $\beta_P$ by its definition above.

The bridges of $\beta_{P+B}$ are subgraphs of $G$ that can attach to $P + B$. We examine various subsets of $\beta_{P+B}$ that, when combined with $P$, are separable from $B$ by a 2-cut $\{s_1, s_2\}$. Consider a subgraph $S$ of $G$ that is separable from

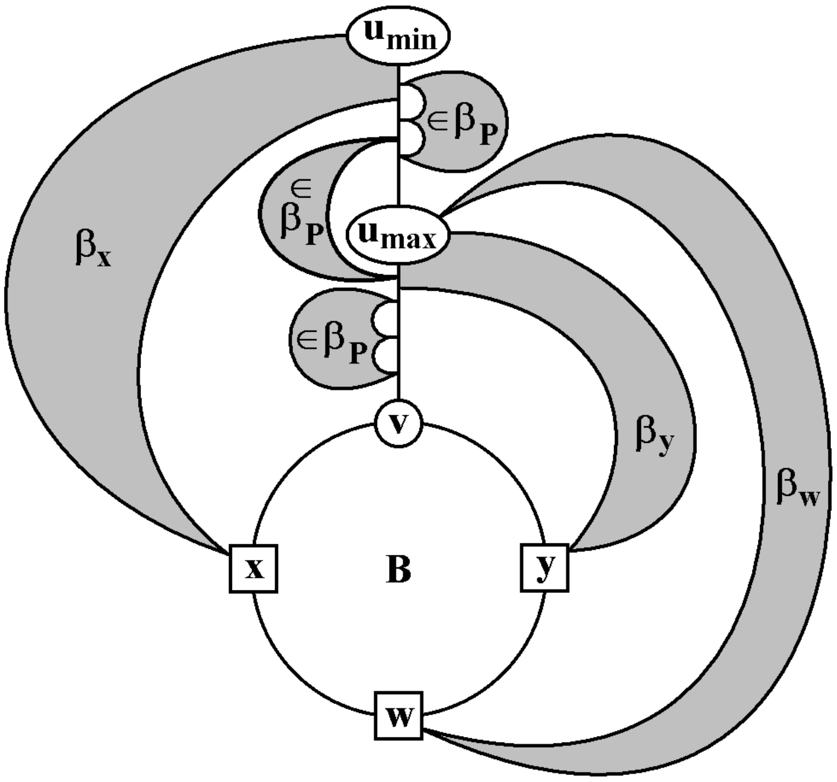

Figure 12: Example of bridges in $\beta_{P+B}$. In this depiction, $\beta_x$ has the farthest attachment to an ancestor of $v$, i.e. $u_{min} = u_x$. The farthest ancestor attachment for $\beta_y$ and $\beta_w$ are shown to be equal, so $u_{max} = u_w = u_y$. Since $\beta_y$ has a point of attachment closer to $v$ than $u_{max}$, a $K_{3,3}$ homeomorph can be obtained via minor E$_3$ in Figure 10(c). In this example, $\beta_w$ is shown attached only to $u_{max}$, so $u_w = u_{max} = \hat{u}_w$, whereas for $\beta_x$ and $\beta_y$, $u_x = u_{max} < \hat{u}_x$ and $u_y = u_{max} < \hat{u}_y$. In this example, there are three bridges in $\beta_P$ attached along the path $P = (u_{min}, \ldots, v)$. The topmost bridge in $\beta_P$, attached closest to $u_{min}$, does not straddle $u_{max}$, so it is also in $B_L$. The bottommost bridge in $\beta_P$, attached closest to $v$, is also a non-straddling bridge, so it is in $B_H$. The middle bridge in $\beta_P$, shown with two only points of attachment, is a bridge that straddles $u_{max}$, so it is not in $B_L$ nor $B_H$. Due to this straddling bridge, a $K_{3,3}$ homeomorph can be obtained via minor E$_7$ in Figure 11(c).

$B$ by the 2-cut $\{s_1, s_2\}$. Since three disjoint paths are needed to access an image vertex of a $K_{3,3}$ homeomorph, if $S - \{s_1, s_2\}$ contains an image vertex of a $K_{3,3}$ homeomorph, then all its image vertices are in $S$. Thus, to preserve any $K_{3,3}$ homeomorph with one or more image vertices in $S - \{s_1, s_2\}$, we need at most one path from $B$ to help connect $s_1$ and $s_2$.

The desired 2-cuts exist due to the absence of the conditions in cases 2, 3, 6 and 7. Some additional notation will aid the proof. Let $\hat{u}_w$ denote the maximum numbered ancestor connection of the bridge set $\beta_w$, i.e. the closest ancestor of $v$ to which a bridge of $\beta_w$ attaches by an unembedded back edge. Similarly, let $\hat{u}_x$ denote the maximum numbered ancestor connection of the bridge set $\beta_x$, and let $\hat{u}_y$ denote the maximum numbered ancestor connection of the bridge set $\beta_y$. The *ancestor attachment range* of a bridge set $\beta_w$, $\beta_x$ or $\beta_y$ is $(u_w, \ldots, \hat{u}_w)$, $(u_x, \ldots, \hat{u}_x)$ or $(u_y, \ldots, \hat{u}_y)$, respectively. Let $\beta_H$ denote a subset of $\beta_P$ consisting of non-straddling bridges that attach to the path $(u_{max}, \ldots, v)$. Let $\beta_L$ denote the subset of non-straddling bridges in $\beta_P - \beta_H$. The bridges in $\beta_L$ attach to ancestors of $v$ with DFS numbers less than or equal to $u_{max}$.

The absence of the conditions of cases 2 and 3 places severe restrictions on the ancestor attachment ranges of bridge sets $\beta_w$, $\beta_x$ and $\beta_y$. When the condition of case 2 is absent, $\hat{u}_w \leq u_{max}$, and all vertices in the ancestor attachment range of $\beta_w$ must be less than or equal to one of $u_x$ or $u_y$ (no closer to $v$ than one of $u_x$ or $u_y$). Otherwise, $\hat{u}_w$ would be a descendant of both $u_x$ and $u_y$, which contradicts the absence of the condition of case 2. Without loss of generality, let $\hat{u}_w \leq u_x$ since $\hat{u}_w \leq u_y$ is symmetric.

There are four initial possibilities to consider: $u_w = \hat{u}_w$ and $u_w < \hat{u}_w$ combined with each of $u_x = \hat{u}_x$ and $u_x < \hat{u}_x$. However, when $u_x < \hat{u}_x$, there is no position for $u_y$ that does not result in meeting the conditions of case 3 since we already have $\hat{u}_w \leq u_x$ from above. Hence, $u_x = \hat{u}_x$.

Under both remaining possibilities ($u_w = \hat{u}_w$ and $u_w < \hat{u}_w$), if $\hat{u}_w < u_x$, then the absence of the conditions of case 3 ensures that $u_y = \hat{u}_y = u_x$. This common ancestor attachment point for $\beta_X$ and $\beta_Y$, denoted $u_{xy}$, is equal to $u_{max}$. Hence, $\beta_x$ is separable by the 2-cut $\{u_{max}, x\}$, $\beta_y$ is separable by the 2-cut $\{u_{max}, y\}$, and $\beta_H$ is separable by the 2-cut $\{u_{max}, v\}$. Since $\hat{u}_w < u_{max}$, we have the ancestor attachment range structure corresponding to case 6 (see Figure 11(b)) except the absence of the condition of case 6 ensures that $\beta_P$ contains no straddling bridges. Therefore, $\beta_w \cup \beta_L$ is also separable from $B$ by the 2-cut $\{u_{max}, w\}$.

Now consider both possibilities ($u_w = \hat{u}_w$ and $u_w < \hat{u}_w$) when $\hat{u}_w = u_x$. If $u_w < \hat{u}_w$, then the absence of the conditions of case 3 ensures that $u_y = \hat{u}_y = u_x$, so the separability arguments in the above paragraph apply. If $u_w = \hat{u}_w$, then the fact that $\hat{u}_w = u_x$ means that $\beta_W$ and $\beta_X$ share a common ancestor attachment point, denoted $u_{wx}$, and it equals $u_{max}$. Note that $\hat{u}_y \leq u_{max}$ due to the absence of the conditions of case 3. If $u_y = u_{max}$, then $u_{max}$ is a common ancestor attachment point for $\beta_W$, $\beta_X$ and $\beta_Y$; thus, $\beta_L$ is empty, $\beta_H$ is separable by the 2-cut $\{u_{max}, v\}$, and each of $\beta_W$, $\beta_X$ and $\beta_Y$ are separable by a 2-cut containing $u_{max}$ and $w$, $x$ and $y$, respectively. On the other hand, if $u_y < u_{max}$, then we have the ancestor attachment range structure corresponding to case 7 (see Figure 11(c)) except the absence of the condition of case 7 ensures that $\beta_P$ contains no straddling bridges. Thus, $\beta_w$ is separable by the 2-cut $\{u_{max}, w\}$, $\beta_x$ is separable by the 2-cut $\{u_{max}, x\}$, $\beta_H$ is separable by the 2-cut $\{u_{max}, v\}$, and $\beta_y \cup \beta_L$ is separable by the 2-cut $\{u_{max}, y\}$.

Finally, suppose there is a $K_{3,3}$ homeomorph whose image vertices are on the 2-cut vertices for the various bridge subsets of $\beta_{P+B}$ discussed above, i.e. on three or more of $u_{max}$, $v$, $w$, $x$, and $y$. In the absence of the conditions of cases 2, 3, 6 and 7, only five vertices appear in all the 2-cuts that separate the bridge subsets from one another. Therefore, at least one of the six image vertices would be internal to one of the bridge subsets, i.e. in the bridge subset excluding the 2-cut vertices, which contradicts the assumption that any of the image vertices are outside of that bridge subset plus its 2-cut.

Thus, the failure to find a $K_{3,3}$ homeomorph based on the biconnected component $B$ implies that the $K_5$ homeomorph involving $B$ can be planarized by ignoring the back edges that the Walkdown failed to embed between $v$ and vertices in the DFS subtree rooted by $w$. Provided that this occurs on all biconnected components for which the Walkdown failed to embed a back edge, the planarity algorithm can simply proceed at vertex $v-1$, except for $v=0$, in which case the planarity of the reduced graph proves that $G$ is $K_{3,3}$-less. □

Much of the testing is performed on the biconnected component $B$, so if a $K_{3,3}$ homeomorph is not found using $B$, then $B$ is reduced to the edge $\{x, y\}$ plus the 4-cycle $(v, \ldots, x, \ldots, w, \ldots, y, \ldots, v)$. For any of these edges that do not exist in the original graph, a virtual edge $e$ is added and the path from the original graph that connects the endpoints of $e$ is associated with $e$. Later, if a $K_{3,3}$ is found, all virtual edges are replaced by their associated paths. This reduction of $B$ avoids a non-constant cost for $B$ should it become part of a larger biconnected component that must subsequently be inspected for a $K_{3,3}$ homeomorph. Further, if that inspection fails, note that $B$ is supplanted by the reduced form of the containing biconnected component. Thus, further processing of any virtual edge is limited to a constant as the visitation of a virtual edge will correspond to one of its removal from the external face during an edge addition, its elimination during isolation of a $K_{3,3}$ homeomorph, or its elimination by the aforementioned reduction.

The biconnected component reduction strategy solves most of the problems of maintaining linear total work for the $K_{3,3}$ search augmentations to the edge addition planarity algorithm. However, the tests for cases 2, 3, 6, and 7 require work to be done outside of the biconnected component $B$. In particular, work done along the path from $v$ to $u_{max}$ (except the endpoints) may be repeated numerous times if there are many $K_5$ homeomorphs that use the path.

For cases 6 and 7, a bridge that straddles $u_{max}$ is sought. The algorithm proceeds from $v$ up to and excluding $u_{max}$. For each vertex $p$, we test whether either the least ancestor directly adjacent to $p$ by a back edge is less than $u_{max}$ or $p$ has a DFS child $c$ not an ancestor of $x$, $y$ and $w$ that has a connection to an ancestor of $u_{max}$ (in other words, whether the child $c$ has a lowpoint less than $u_{max}$). The planarity algorithm initialization calculates the least ancestor of each vertex as well as a DFS child list sorted by lowpoint, so we process $p$ in constant time by examining its least ancestor setting and at most the first two children in the DFS child list of $p$. To ensure that failed straddling bridge tests are not performed more than once along a path, the negative result of a search is cached in an initially *nil* data member called noStraddle. In each tree edge along the portion of the path $(v, \ldots, u_{max})$ traversed in the straddling bridge test, we set the member noStraddle equal to $u_{max}$. Any future search for a straddling bridge can be terminated (before reaching $u_{max}$) at the first edge whose noStraddle member is not *nil*. If the terminated straddling bridge test was searching for a bridge straddling $u_{max}$, then the test result is negative. If a future step tests for a bridge that straddles a descendant $u_d$ of $u_{max}$, then the result is positive since the bridge containing biconnected component $B$ in the current step straddles $u_d$. Finally, a future step will never test for a straddling bridge between a descendant and an ancestor of $u_{max}$ because the mere need for such a search (whether or not it would succeed) implies that a straddling bridge would be found in the current step.

For cases 2 and 3, the path $(v, \ldots, u_{max})$ (exclusive of its endpoints) could be directly inspected for vertices that connect by unembedded back edges to $x$, $y$, or $w$ (possibly through their separated descendants). However, an alternative is required in order to prevent multiple searches along the path $(v, \ldots, u_{max})$. Instead, it is possible to achieve linear time by taking advantage of the fact that the planarity algorithm itself will encounter the conditions for cases 2 and 3, if they exist. If a future pertinent connection from $x$, $y$ or $w$ to a descendant $u_d$ of $u_{max}$ exists, then in step $u_d$ the planarity algorithm will be required to embed a back edge connnection that would join $u_d$, $x$, $y$ and $w$ into one biconnected component. For each vertex, we can define an initially *nil* data member called mergeBlocker that can be used to delay the work of detecting the condition for cases 2 and 3. The mergeBlocker members of $x$, $y$ and $w$ are simply set equal to $u_{max}$. Then, the Walkdown can be slightly modified to pre-test each biconnected component merge sequence with a check of the mergeBlocker member of each vertex on the mergeStack. If any have a mergeBlocker setting that indicates an ancestor of the current vertex, then a prior step could have isolated a $K_{3,3}$ homeomorph based on case 2 or 3. The data structures are then easily reset to return to the settings of step $v$, except that the additional information about case 2 or 3 is known without violating the total linear time bound. Finally, note that the modified Walkdown may not discover the merge blocked vertex if it first discovers a non-planarity condition in step $u_d$. However, in this case a $K_{3,3}$ homeomorph will be discovered by non-planarity minor A or B since the merge blocked vertex ($x$, $y$ or $w$) is both pertinent to $u_d$ and future pertinent due to its connection to $u_{max}$. Indeed, this always happens for case 2, so the mergeBlocker member needs only to be set on $x$ and $y$ for case 3.

Besides these optimizations of the $K_{3,3}$ search augmentations, much of the linear time performance of the Edge Addition $K_{3,3}$ Search is inherited from the core edge addition planarity algorithm. In a few cases, this included more careful implementation of preexisting low level routines that did not have to be as fast when only isolating Kuratowski subgraphs. Finally, since Asano [1] proved that a graph with at least $3n - 5$ edges contains a subgraph homeomorphic to $K_{3,3}$, then by the arguments above, we have the following theorem:

**Theorem 5** *Given an input graph $G$ with $n$ vertices, the Edge Addition $K_{3,3}$ Homeomorph Search algorithm finds a subgraph homeomorphic to $K_{3,3}$ in $G$ or determines that $G$ is $K_{3,3}$-less in $O(n)$ time.*

## 7 Conclusion

This paper has presented new linear time solutions to several subgraph homeomorphism problems, unifying them with the theoretical framework of the edge addition planarity algorithm [3]. This eliminated the requirement of an efficient SPQR-tree [9] or other means of triconnected component decomposition, which was particularly relevant to $K_{3,3}$ search since the requirement of planarity processing was already present in the prior solutions [1, 8]. Efficient implementations of these new methods are available in an open source project (http://code.google.com/p/planarity).

An additional outcome that will likely prove useful in future work is that new techniques related to depth first search were found and exploited to enable the removal of obstructions to planarity or outerplanarity during the operation of the core edge addition planarity algorithm. The creation of the new $K_4$ search in particular forced the full measure of this improvement to be developed since the $K_4$ search algorithm must continue Walkdown operations on a biconnected component directly after performing reductions on it.

As future work, there are new problems and solutions suggested by the algorithms reported in this paper. One problem will be determining a graph result that can be interpreted as a certificate of non-existence of a requested homeomorphic subgraph, just as a planar embedding is a certificate of non-existence of a Kuratowski subgraph. Another will be determining whether the subgraph homeomorphism algorithms of this paper can be adjusted to efficiently find all $K_{2,3}$, $K_{3,3}$ or $K_4$ homeomorphs in a graph, where efficiency is measured as a constant ratio of work to output size. Further investigation will also focus on new solutions for more planarity-related problems, such as for level planarity, finding a maximal planar subgraph, searching for a $K_5$ minor or a subgraph homeomorphic to $K_5$, and projective planar graph embedding. The current techniques for level planarity [13, 15] are linear time but are based on the *PQ*-tree data structure, so a solution based on edge addition planarity is likely to be simpler and more efficient. Similarly, there are two recently reported linear time solutions of the maximal planar subgraph problem [7, 12]. Particularly given prior challenges with using complex data structures to solve this problem [14], it will be valuable to have implementations of those new algorithms to compare with a future method based on edge addition planarity, which would avoid batch processing models and complex satellite data structures. There are no linear time solutions for searching for a subgraph homeomorphic to $K_5$ or a $K_5$ minor. The best reported result is an $O(n^2)$ method for finding a $K_5$ minor [16]. For projective planar embedding, there is a reported linear time solution [18], but as yet no implementation. The intent of the future work will be to create and implement new methods that use a graph as the dominant data structure and that are conceptually straightforward extensions of the edge addition planarity algorithm along with the variations and reduction techniques presented herein.

## References

[1] T. Asano. An approach to the subgraph homeomorphism problem. *Theoretical Computer Science*, 38:249–267, 1985.

[2] J. Boyer and W. Myrvold. Stop minding your P's and Q's: A simplified $O(n)$ planar embedding algorithm. *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 140–146, 1999.

[3] J. Boyer and W. Myrvold. On the cutting edge: Simplified $O(n)$ planarity by edge addition. *Journal of Graph Algorithms and Applications*, 8(3):241–273, 2004.

[4] H. de Fraysseix. Trémaux trees and planarity. *Electronic Notes in Discrete Mathematics*, 31:169–180, 2008.

[5] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. Trémaux trees and planarity. *International Journal of Foundations of Computer Science*, 17(5):1017–1029, 2006.

[6] H. de Fraysseix and P. Rosenstiehl. A characterization of planar graphs by trémaux orders. *Combinatorica*, 5(2):127–135, 1985.

[7] H. N. Djidjev. A linear-time algorithm for finding a maximal planar subgraph. *SIAM Journal on Discrete Mathematics*, 20(2):444–462, 2006.

[8] M. R. Fellows and P. A. Kaschube. Searching for $K_{3,3}$ in linear time. *Linear and Multilinear Algebra*, 29:279–290, 1991.

[9] C. Gutwenger and P. Mutzel. A linear time implementation of SPQR-trees. In J. Marks, editor, *Proceedings of the* $8^{th}$ *International Symposium on Graph Drawing (GD 2000)*, volume 1984 of *Lecture Notes in Computer Science*, pages 77–90. Springer-Verlag, 2001.

[10] B. Haeupler and R. E. Tarjan. Planarity algorithms via PQ-trees. In P. O. de Mendez, M. Pocchiola, D. Poulalhon, J. L. R. Alfonsn, and G. Schaeffer, editors, *The International Conference on Topological and Geometric Graph Theory*, volume 31 of *Electronic Notes in Discrete Mathematics*, pages 143–149. ScienceDirect, 2008.

[11] J. Hopcroft and R. Tarjan. Dividing a graph into triconnected components. *SIAM Journal of Computing*, 2:135–158, 1973.

[12] W.-L. Hsu. A linear time algorithm for finding a maximal planar subgraph based on PC-trees. In L. Wang, editor, *COCOON 2005*, volume 3595 of *Lecture Notes in Computer Science*, pages 787–797. Springer-Verlag, 2005.

[13] M. Jünger and S. Leipert. Level planar embedding in linear time. *Journal of Graph Algorithms and Applications*, 6(1):67–113, 2002.

[14] M. Jünger, S. Leipert, and P. Mutzel. Pitfalls of using PQ-trees in automatic graph drawing. In G. D. Battista, editor, *Proceedings of the* $5^{th}$ *International Symposium on Graph Drawing '97*, volume 1353 of *Lecture Notes in Computer Science*, pages 193–204. Springer, Sept. 1997.

[15] M. Jünger, S. Leipert, and P. Mutzel. Level planarity testing in linear time. In S. Whitesides, editor, *Proceedings of the* $6^{th}$ *International Symposium on Graph Drawing (GD-98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 1998.

[16] A. Kézdy and S. McGuinness. Sequential and parallel algorithms to find a $K_5$ minor. *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–356, 1992.

[17] P. C. Liu and R. C. Geldmacher. An $O(\max(m, n))$ algorithm for finding a subgraph homeomorphic to $K_4$. *Proceedings of the* $11^{th}$ *Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 597–609, 1980.

[18] B. Mohar. Projective planarity in linear time. *Journal of Algorithms*, 15:482–502, 1993.

[19] W.-K. Shih and W.-L. Hsu. A new planarity test. *Theoretical Computer Science*, 223:179–191, 1999.

[20] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.

[21] K. P. Vo. Finding triconnected components of graphs. *Linear and Multilinear Algebra*, 13:143–165, 1983.

[22] S. G. Williamson. Depth-first search and Kuratowski subgraphs. *Journal of the Association for Computing Machinery*, 31(4):681–693, 1984.