SIAM J. COMPUT.
Vol. 28, No. 1, pp. 341–381

SEPARATOR-BASED SPARSIFICATION II: EDGE AND VERTEX CONNECTIVITY*

DAVID EPPSTEIN†, ZVI GALIL‡, GIUSEPPE F. ITALIANO§, AND THOMAS H. SPENCER¶

Abstract. We consider the problem of maintaining a dynamic planar graph subject to edge insertions and edge deletions that preserve planarity but that can change the embedding. We describe algorithms and data structures for maintaining information about 2- and 3-vertex-connectivity, and 3- and 4-edge-connectivity in a planar graph in $O(n^{1/2})$ amortized time per insertion, deletion, or connectivity query. All of the data structures handle insertions that keep the graph planar without regard to any particular embedding of the graph. Our algorithms are based on a new type of sparsification combined with several properties of separators in planar graphs.

Key words. analysis of algorithms, dynamic data structures, edge connectivity, vertex connectivity, planar graphs

AMS subject classifications. 68P05, 68Q20, 68R10

PII. S0097539794269072

1. Introduction.

Sparse certificates, small graphs that retain some property of a larger graph, appear often in graph theory, especially in problems of edge and vertex connectivity [2, 13, 31, 35]. The main motivation for studying sparse certificates lies in the fact that they are effective tools for speeding up many graph algorithms. To check whether a graph G has a given property $\mathcal{P}$, one can first compute a sparse certificate $\mathcal{C}$ for property $\mathcal{P}$ and then run an algorithm for $\mathcal{P}$ on the certificate rather than on G itself. This is favorable whenever computing certificates is faster than checking property $\mathcal{P}$. This method has led to improved algorithms for testing k-edge- and k-vertex-connectivity sequentially [16, 31, 35] and in parallel [2], for finding three independent spanning trees [1], and for reliability in distributed networks [26]. With the *sparsification* technique [8], sparse certificates additionally became an important tool for speeding up dynamic graph algorithms, in which edges may be inserted into and deleted from a graph while some graph property must be maintained throughout the sequence of modifications.

*Received by the editors June 6, 1994; accepted for publication (in revised form) October 30, 1996; published electronically June 15, 1998. Portions of this paper were presented at the 25th Annual ACM Symp. on Theory of Computing, San Diego, CA, 1993 [10].

http://www.siam.org/journals/sicomp/28-1/26907.html

†Department of Information and Computer Science, University of California, Irvine, CA 92697-3425 (eppstein@ics.uci.edu, http://www.ics.uci.edu/∼eppstein/). The research of this author was supported in part by NSF grant CCR-9258355 and by matching funds from Xerox Corp.

‡Computer Science Department, Columbia University, New York, NY 10027 (galil@cs.columbia.edu) and Computer Science Department, Tel-Aviv University, Tel-Aviv, Israel. The research of this author was supported in part by NSF grant CCR-90-14605 and CISE Institutional Infrastructure grant CCR-90-24735.

§Dipartimento di Matematica Applicata e Informatica, Università "Ca' Foscari," Venice, Italy (italiano@dsi.unive.it, http://www.dsi.unive.it/∼italiano/). The research of this author was supported in part by EU ESPRIT Long Term Research Project ALCOM-IT under contract no. 20244, by a Research Grant from University "Ca' Foscari" of Venice, and by the Italian MURST Project "Efficienza di Algoritmi e Progetto di Strutture Informative."

¶Department of Computer Science, University of Nebraska at Omaha, Omaha, NB 68182-0243 (spencer@unocss.unomaha.edu). Current address: 5740 S. 100th Plaza #2A, Omaha, NB 68127. This research was partially supported by the University Committee on Research, University of Nebraska at Omaha and by NSF grant CCR-9319772.

While sparsification has many applications in algorithms for general graphs, it seemed unlikely that it could be used to speed up algorithms for special families of graphs that are already sparse, such as planar graphs. However, algorithms for planar graphs are especially important, as these graphs arise frequently in applications. In the companion paper [11] we developed a new, general technique for dynamic planar graph problems, based upon the notion of *compressed certificates*, which have both fewer edges and fewer vertices than the original graph. We expanded the notion of certificate to a definition for graphs in which a subset of the vertices is denoted as *interesting*; these *compressed certificates* may reduce the size of the graph by removing uninteresting vertices. Note that this is a generalization of the previous certificates, as compressed certificates reduce to sparse certificates in the special case where all the vertices are interesting. Using the notion of compressed certificates, we defined a type of *separator-based sparsification* based on *separators*, small sets of vertices the removal of which splits the graph into roughly equal-size components. We then applied separator-based sparsification to maintain information about the minimum spanning forest, connectivity, and 2-edge-connectivity of a planar graph. We further showed how to maintain a graph subject to *arbitrary* edge insertions and deletions, with queries that test whether the graph is currently planar or whether a potential new edge would violate planarity.

In this paper we extend these ideas in several ways. Our first constribution is to adapt separator-based sparsification from the companion paper [11] to work on more general certificates and properties. Namely, we extend the notion of compressed certificates to properties that can be defined with respect to a particular pair of vertices rather than on the whole graph. We refer to these as *local* certificates as opposed to *global* certificates. The most general notion of certificate is a *full* certificate, which is at the same time a local and global certificate.

Our second contribution is to prove a number of structural properties of certificates for edge connectivity in general graphs. Among these properties, we give necessary and sufficient conditions for a graph to be a local or global certificate of k-edge-connectivity for any k. This characterization is not only a powerful algorithmic tool, as we show in this paper, but also contributes a new insight into the structural properties of edge connectivity and improves our understanding of certificates. We believe that these structural properties may be of independent interest and find applications to other graph-theoretical areas.

Thirdly, as a first application of our compressed certificates, we use them to develop dynamic planar graph algorithms. We maintain information about 3- and 4-edge-, and 2- and 3-vertex-connectivity in a planar graph during an intermixed sequence of edge deletions, edge insertions that keep the graph planar, and connectivity queries in $O(n^{1/2})$ amortized time per operation. All our algorithms improve previous bounds: for 2- and 3-vertex- and 3-edge-connectivity, the best previous time bound was $O(n^{2/3})$ amortized [8, 19, 21], while for 4-edge-connectivity nothing better than testing the graph from scratch after each update was known. These bounds apply to problems in which insertions need not respect a fixed embedding of the graph; a number of other papers have worked on dynamic graph problems such as minimum spanning forests, connectivity, and planarity testing for graphs with a fixed planar embedding [12, 14, 15, 18, 19, 22, 21, 24, 32, 33].

Finally, our methods apply to static as well as dynamic graph problems. A general certificate construction method from our companion paper, together with the certificates defined here, gives a unified method of testing 3- and 4-edge-, and 2- and

3-vertex-connectivity in planar graphs, in linear time. In recent work, Eppstein [7] has shown how to compute k-edge- or k-vertex-connectivity in planar graphs in linear time for any constant k.

The remainder of this paper consists of the following sections. Section 2 contains basic definitions. In section 3 we recall some properties of separator-based sparsification and compressed certificates from reference [11]. In section 4 we prove the properties and describe the tools we will be using for our certificates for edge connectivity. In section 5 compressed certificates for edge connectivity are developed, and our bounds for 3- and 4-edge-connectivity are proved. In section 6 sparsification is applied to fully dynamic 2- and 3-vertex-connectivity by using compressed certificates already available in the literature. Finally, in section 7 we list some open problems and concluding remarks.

2. Preliminaries. In this section we introduce the notions of vertex and edge connectivity of a graph. Next, we review a generalized tree, due to Dinitz, Karzanov, and Lomonosov [4], that describes an elegant decomposition of a graph using its connectivity edge-cuts.

2.1. Edge and vertex connectivity. Let $G = (V, E)$ be an undirected graph, possibly with parallel edges. Throughout the paper, we denote by m the number of edges and by n the number of vertices in G. Given an integer $k \geq 2$, a pair of vertices $\langle u, v\rangle$ is said to be *k-edge-connected* in G if the removal of any $(k-1)$ edges in G leaves u and v connected. It is well known that u and v are k-edge-connected if and only if there are k edge-disjoint paths between u and v. k-edge-connectivity is an equivalence relationship, and we denote it by $\equiv_k$; i.e., if a pair of vertices $\langle x, y\rangle$ is k-edge-connected, we write $x \equiv_k y$. The vertices of a graph G are partitioned by this relationship into equivalence classes that we call *k-edge-connected classes*. Note that according to this definition a k-edge-connected class of G is a subset of vertices of G. G is said to be *k-edge-connected* if the removal of any $(k-1)$ edges leaves G connected. As a result of these definitions, G is k-edge-connected if and only if any two vertices of G are k-edge-connected. An edge set $E' \subseteq E$ is an *edge-cut for vertices x and y* if the removal of all the edges in E' disconnects G into two graphs, one containing x and the other containing y. An edge-set $E' \subseteq E$ is an *edge-cut for G* if the removal of all the edges in E' disconnects G into two graphs. An edge-cut E' for G (for x and y, respectively) is *minimal* if removing any edge from E' and re-inserting it back into G reconnects G (x and y, respectively). The cardinality of an edge-cut E', denoted by $|E'|$, is equal to the number of edges in E'. An edge-cut E' for G (for x and y, respectively) is said to be a *minimum edge-cut* or a *connectivity edge-cut*, if there is no other edge-cut E'' for G (for x and y, respectively) such that $|E''| < |E'|$. A connectivity edge-cut of cardinality 1 is called a *bridge*. The graphs left after deleting all the bridges of G are called the *2-edge-connected components* of G. Note the difference between 2-edge-connected classes and 2-edge-connected components: a 2-edge-connected class is a *subset of vertices* of G, while a 2-edge-connected component is a *subgraph* of G. However, 2-edge-connected classes and 2-edge-connected components are strictly related: indeed, a 2-edge-connected component is a subgraph of G induced by a 2-edge-connected class.

We now list some well-known properties that are an immediate consequence of the previous definitions and that will be used throughout this paper. First, any connectivity edge-cut must be minimal and must disconnect G exactly into two graphs. Furthermore, all the connectivity edge-cuts for G must have the same cardinality. Thus, the notion of *cardinality of the connectivity edge-cuts for G* is well defined: it

gives exactly the minimum number of edges whose deletion disconnects G. Similarly, given any two vertices x and y in G, all the connectivity edge-cuts for x and y have the same cardinality, and we speak about *the cardinality of the connectivity edge-cuts for x and y*. Given a graph G, the *edge connectivity of G* is defined as the cardinality of the connectivity edge-cuts for G, i.e., the minimum number of edges whose deletion disconnects G. We denote the edge connectivity of G by $\lambda(G)$: note that G is k-edge-connected if and only if $k \leq \lambda(G)$. Similarly, given any two vertices x and y in G, the *edge connectivity between x and y in G* is defined as the cardinality of the connectivity edge-cuts for x and y, i.e., the minimum number of edges whose deletion disconnects x and y in G. We denote the edge connectivity between x and y in G by $\lambda_{x,y}(G)$. As a consequence of this definition, $x \equiv_k y$ if and only if $k \leq \lambda_{x,y}(G)$.

Analogous definitions can be given for the case of vertex connectivity. A vertex set $V' \subset V$ (respectively, $V' \subseteq V - \{x, y\}$) is a *vertex-cut* for G (respectively, for vertices x and y) if the removal of all the vertices in V' disconnects G (respectively, x and y). The cardinality of a vertex-cut V', denoted by $|V'|$, is given by the number of vertices in V'. A vertex-cut V' for G (for x and y, respectively) is said to be a *minimum vertex-cut* or a *connectivity vertex-cut* if there is no other vertex-cut V'' for G (x and y, respectively) such that $|V''| < |V'|$. Two vertices x and y are k-vertex-connected if and only if a minimum vertex-cut for x and y contains at least k vertices. G is said to be *k-vertex-connected* if all its pairs of vertices are k-vertex-connected; equivalently, the minimum vertex-cut of G has cardinality k or more. A minimum vertex-cut of cardinality 1 is called an *articulation point*. Again, two vertices x and y in G are k-vertex-connected if and only if there are at least k vertex-disjoint paths between x and y.

2.2. The cactus tree. We now describe a tree-like decomposition of a k-edge-connected graph G into its $(k+1)$-edge-connected classes, which can be found in the beautiful work of Dinitz, Karzanov, and Lomonosov [4], and which we will be using throughout this paper. The generalized tree that describes this decomposition is called the *cactus tree*: note that it need not be a standard tree. We do not give many details here, referring the interested reader to references [4, 28]. The heart of this decomposition is the *Crossing Lemma* of [4], which can be informally stated as follows. Let G be a graph of edge connectivity λ: if any two λ-edge-cuts of G divide $V(G)$ into four (nonempty) parts, then shrinking these four parts produces a super-cycle having four super-nodes and exactly $\lambda/2$ parallel edges between any two neighbor super-nodes.

We mention some consequences of the Crossing Lemma, referring to [4, 28] for the full details and explanations. First of all, for λ odd, $\lambda/2$ is not an integer. Thus, according to the Crossing Lemma, for λ odd two different λ-edge-cuts of G cannot divide $V(G)$ into four nonempty parts. The Crossing Lemma also implies that there can be only $O(n)$ connectivity edge-cuts if λ is odd, and $O(n^2)$ of them if λ is even [4]. The set of all the connectivity edge-cuts of G can be compactly represented by a "tree-like" graph with weights on the edges, called the *cactus tree of G* and denoted by $\mathcal{T}(G)$, which can be constructed in $O(m + \lambda^2 n \log n)$ time [17]. Each vertex in G maps to exactly one node in $\mathcal{T}(G)$, so that any node of $\mathcal{T}(G)$ corresponds to a (possibly empty) subset of vertices from G. An edge-cut $(\mathcal{A}, \bar{\mathcal{A}})$ in $\mathcal{T}(G)$ corresponds to an edge-cut $(A, \bar{A})$ of G, where A consists of all the vertices of G that are mapped into nodes of $\mathcal{A}$. A λ-cut of $\mathcal{T}(G)$ is an edge-cut of $\mathcal{T}(G)$ of total weight λ. Each minimal edge-cut of $\mathcal{T}(G)$ (i.e., an edge-cut from which no edge can be removed) is also a λ-cut in $\mathcal{T}(G)$, and it corresponds to a connectivity edge-cut in G. Each connectivity edge-cut in G

corresponds to one or more λ-cuts in $\mathcal{T}(G)$. Thus, the minimal edge-cuts of $\mathcal{T}(G)$ compactly represent all the connectivity edge-cuts of G.

For λ odd, the cactus tree is particularly simple: it is actually a tree, all its edges are of weight λ, and any minimal edge-cut of $\mathcal{T}(G)$ is obtained by removing one of its edges. For λ even, we can have crossing connectivity edge-cuts, and $\mathcal{T}(G)$ is a tree of cycles. Namely, $\mathcal{T}(G)$ consists of cycles, such that any two cycles have at most one single node in common; thus, no edge of $\mathcal{T}(G)$ can be in more than one cycle. Every edge in a cycle is called a *cycle-edge* and has weight $\lambda/2$. Note that there can be cycles consisting only of two edges: we refer to these cycles as 2-*cycles*. Minimal edge-cuts of $\mathcal{T}(G)$ are obtained by removing any pair of cycle-edges that lies on the same cycle. Hence, a 2-cycle defines only one minimal edge-cut of $\mathcal{T}(G)$, while a cycle with $p \geq 3$ edges defines exactly $p(p-1)/2$ distinct minimal edge-cuts of $\mathcal{T}(G)$. The cactus tree of a graph is basically unique, up to the following convention: if λ is even, either three nodes can be in a triangle of cycle-edges, or they can all be joined to another new node with 2-cycles.

We now describe in more detail the cactus trees we will be using, namely, the cases $\lambda = 1, 2, 3$. The interested reader can find further details on cactus trees in references [4, 28]. If G has edge connectivity $\lambda = 1$, the cactus tree is actually a tree, called the *bridge-block tree of* G: its nodes correspond to the 2-edge-connected classes (and thus components) of G, and its edges to the bridges of G. For $\lambda = 2$, the cactus tree is a tree of cycles: indeed, if we shrink the 3-edge-connected classes of a 2-edge-connected graph, each biconnected component of the shrunken graph is a simple cycle. Even though the shrunken graph is not a tree, it can easily be represented as such (see Figure 1).

For $\lambda = 3$ the cactus tree is actually a tree having one node for each 4-edge-connected class and one edge for each 3-edge-cut. There might also be nodes in the cactus that correspond to no 4-edge-connected classes of G, as shown in Figure 2. We refer the interested reader to [4, 28] for a more detailed explanation about these nodes and only mention here that they are needed to keep the correspondence between minimal cuts of $\mathcal{T}(G)$ and connectivity edge-cuts of G.

3. Separator-based sparsification. *Sparsification* was introduced in [8] as a technique for designing fully dynamic graph algorithms, in which edges may be inserted into and deleted from a graph while some graph property must be maintained. This technique is based upon a suitable combination of graph decomposition and edge elimination and can be described as follows. Let G be a graph with m edges and n vertices: we partition the edges of G into a collection of sparse subgraphs (i.e., subgraphs with $O(n)$ edges) and summarize the relevant information for each subgraph in an even sparser *certificate*. Intuitively, a certificate for a given property is a smaller graph that retains the same property (we will give a more precise definition later). We merge certificates in pairs, producing larger subgraphs which we make sparse by again applying the certificate reduction. The result is a balanced binary tree in which each node is represented by a sparse certificate. Each edge insertion or deletion causes changes in $\log(m/n)$ tree nodes, but each such change occurs in a subgraph with $O(n)$ edges, reduced from the m edges in the original graph. This reduces a time bound of $T(m, n)$ to $O(T(O(n), n) \log(m/n))$. Using a more sophisticated approach (described in [9]), we can eliminate the logarithmic factor from this bound. This reduces the time bounds for many dynamic graph problems, including vertex and edge connectivity and minimum spanning forests, to exactly match the bounds known for sparse graphs.

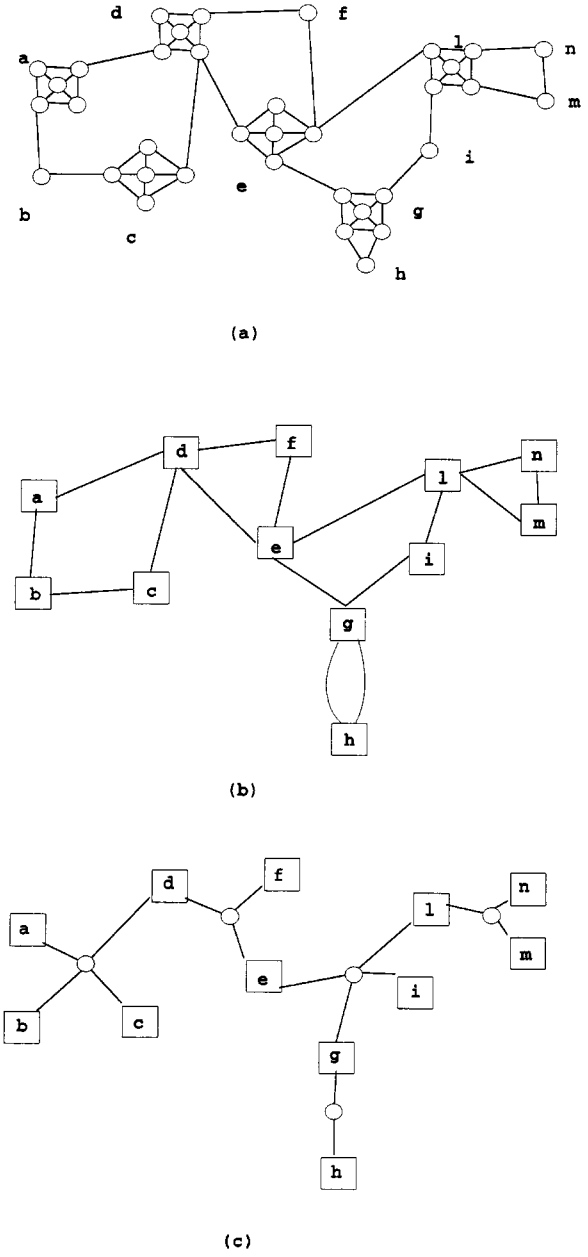

FIG. 1. (a) *A 2-edge-connected graph G;* (b) *the cactus tree of G;* (c) *the tree obtained by replacing each cycle of the cactus tree with a new vertex.*

In the companion paper [11], we developed a new, general technique for dynamic planar graph problems. In all these problems, we deal with either arbitrary or planarity-preserving insertions and therefore allow changes of the embedding. The new ideas behind this technique are the following. We expand the notion of a certificate to a definition for graphs in which a subset of the vertices is denoted as *interesting*; these *compressed certificates* may reduce the size of the graph by remov-

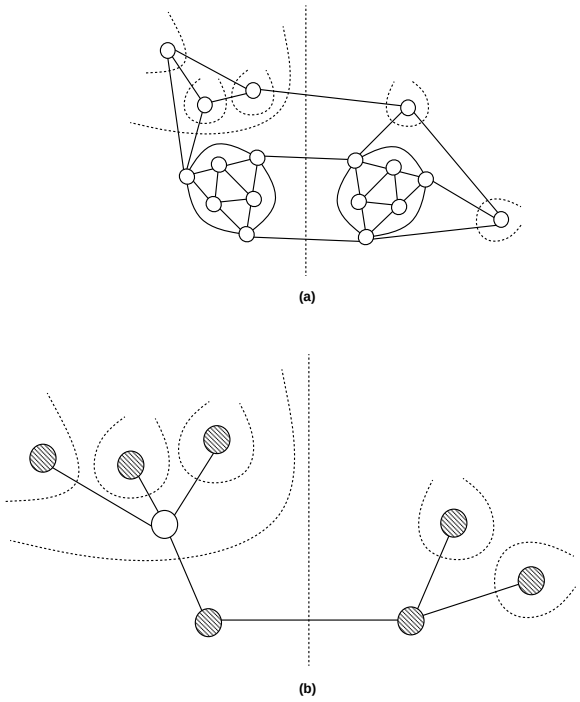

FIG. 2. *A 3-edge-connected graph G and its cactus tree. The 3-edge-cuts of G are shown as dashed lines. Nodes of the cactus not corresponding to 4-edge-connected classes of G are shown in white.*

ing uninteresting vertices. Using this notion of compressed certificates, we define a type of sparsification based on *separators*, small sets of vertices the removal of which splits the graph into roughly equal-size components. Recursively finding separators in these components gives a *separator tree* which we also use as our *sparsification tree*; the interesting vertices in each certificate will be those vertices used in separators at higher levels of the tree. We introduce the notion of a *balanced separator tree*, which also partitions the interesting vertices evenly in the tree. In [11], we show how to compute such a tree in linear time and how to maintain it dynamically. We call this technique *separator-based sparsification*.

Separator-based sparsification, as described in [11], applies to properties of the entire graph (such as planarity). However, there are some other properties that can be described either as a global graph property or in terms of pairs of vertices. More generally, let $\mathcal{P}$ a property of graphs: we say that $\mathcal{P}$ is *local* if it can be defined with respect to a particular pair (x, y) of vertices in the graph. A query related to property $\mathcal{P}$ is referred to as *global* if $\mathcal{P}$ is meant for the entire graph and is referred to as *local* if $\mathcal{P}$ is meant for a particular pair (x, y). Examples of local properties are edge and vertex connectivity, which are defined both for the entire graph (*global property*) and for any pair of vertices in the graph (*local property*). While maintaining a graph under edge insertions and deletions, at any time we might ask queries on whether the entire graph is k-vertex- (edge-) connected (*global k-connectivity query*), or rather, we might want to ask queries on whether any two given vertices are k-vertex- (edge-) connected (*local k-connectivity query*).

In this paper, we extend separator-based sparsification to work also on local properties. We first need the notion of *certificate* from [11], for which we adopt the new terminology of *global certificate*,

DEFINITION 3.1. *Let graph property $\mathcal{P}$ be fixed and let G be a graph with a set $X \subseteq V(G)$. A* global certificate for X in G *is a graph C, with $X \subseteq V(C)$, such that for any H with $V(G) \cap V(H) \subseteq X$, $V(C) \cap V(H) \subseteq X$, $G \cup H$ has property $\mathcal{P}$ if and only if $C \cup H$ has the property.*

The set X in Definition 3.1 represents the interesting vertices of G. According to this definition, a global certificate C captures the behavior of the entire graph G with respect to additions that only touch the interesting vertices. For instance, let $\mathcal{P}$ be k-edge-connectivity and let C be a global certificate for X in G: then, for any H such that $V(H) \cap V(G) \subseteq X$, $C \cup H$ is k-edge-connected if and only if $G \cup H$ is k-edge-connected. Note that when $X = V(G)$, this definition reduces to the one in [8].

For local properties, we need a slightly different notion of certificate, which we call a *local certificate*.

DEFINITION 3.2. *Let $\mathcal{P}$ be a local property of graphs and let G be a graph with a set $X \subseteq V(G)$. A graph C is a* local certificate *of $\mathcal{P}$ for X in G if and only if for any H with $V(H) \cap V(G) \subseteq X$, $V(C) \cap V(H) \subseteq X$, and any x and y in $V(H)$, $\mathcal{P}$ is true for (x, y) in $G \cup H$ if and only if it is true for (x, y) in $C \cup H$.*

Note that a local certificate C has to preserve the behavior of the property not only with respect to the interesting vertices in G, but also with respect to all vertices in H. For instance, let $\mathcal{P}$ be k-edge-connectivity, and let C be a local certificate for X in G: then, for any H such that $V(H) \cap V(G) \subseteq X$, and any $x, y \in V(H)$, $x \equiv_k y$ in $C \cup H$ if and only if $x \equiv_k y$ in $G \cup H$.

The two notions of global and local certificate can be combined together to yield a stronger notion of certificate.

DEFINITION 3.3. *Let graph property $\mathcal{P}$ be fixed and let G be a given graph with $X \subseteq V(G)$. A* full certificate for X in G *is graph C, which is both a global and a local certificate for X in G.*

For instance, let $\mathcal{P}$ be k-edge-connectivity and let C be a full certificate for X in G: then, for any H such that $V(H) \cap V(G) \subseteq X$, we have that

(i) $C \cup H$ is k-edge-connected if and only if $G \cup H$ is k-edge-connected; and
(ii) for any $x, y \in V(H)$, $x \equiv_k y$ in $C \cup H$ if and only if $x \equiv_k y$ in $G \cup H$.

Note that both conditions are needed, since neither (i) implies (ii), nor (ii) implies (i). As an example of full certificate, let $\mathcal{P}$ be connectivity. We partition the vertices of X into their connected components in G and replace each connected component by any spanning tree. We claim that this yields a full certificate for connectivity. Indeed, if two vertices in $G \cup H$ are connected by a path, then at each point the path switches between edges of G and edges of H, it will pass through a vertex $x \in X$, and the portion of the path in G can be replaced by a path through the spanning forest of the partition set containing x. Thus vertices are connected in $G \cup H$ if and only if they are connected in $C \cup H$ (i.e., C is a local certificate) and $G \cup H$ is connected if and only if $C \cup H$ is connected (i.e., C is a global certificate). This shows that C is indeed a full certificate.

We will use the term certificates to refer in general to certificates of all types (full, global, or local). We will use explicitly the terms full certificates, global certificates, and local certificates when we refer to these kinds of certificates only. Our method hinges upon the notion of *compressed* certificate.

DEFINITION 3.4. *Let graph property $\mathcal{P}$ be fixed and let G be a given graph with $X \subseteq V(G)$. A* compressed certificate for X in G *is a certificate C for X in G, such that $|C| = O(|X|)$.*

Some basic lemmas about global certificates were proved in the companion paper [11]. They can be easily extended to local and full certificates as well.

LEMMA 3.1 (see [11]). *Let C be a certificate for some set X in a given graph G and let C' be a certificate for X in C. Then C' is also a certificate for X in G.*

LEMMA 3.2 (see [11]). *Let C' be a certificate for X' in G' and let C'' be a certificate for X'' in G'', with $V(G') \cap V(G'') \subseteq X' \cap X''$. Then $C' \cup C''$ is a certificate for $X' \cup X''$ in $G' \cup G''$.*

The following lemma is an immediate consequence of the definition of certificate.

LEMMA 3.3. *Let C be a certificate for some set X in a given graph G. Then C is also a certificate for any set $X' \subset X$ in G.*

Proof. Fix any $X' \subset X$. Let H be given, with $V(H) \cap V(G) \subseteq X'$. Since $X' \subset X$, we have that $V(H) \cap V(G) \subset X$. We claim that this is enough to prove the lemma. Indeed, let $\mathcal{P}$ be the property for which C is a certificate. If C is a global certificate for X in G, it follows that $C \cup H$ has property $\mathcal{P}$ if and only if $G \cup H$ has property $\mathcal{P}$. Thus, C is a global certificate for X' in G. If C is a local certificate for X in G, it follows that, given any two vertices $x, y \in V(H)$, $\mathcal{P}$ is true for x and y in $C \cup H$ if and only if $\mathcal{P}$ is true for x and y in $G \cup H$. Thus, C is a local certificate for X' in G too. If C is both a local and a global certificate for X in G, then by the previous argument, C will be both a local and a global certificate for X' in G. ∎

We showed in [11] that, under certain weak assumptions, the existence of compressed certificates for all G and X is sufficient to prove the existence of a linear-time algorithm for computing such certificates. We require our certificates to satisfy the following additional property.

DEFINITION 3.5. *Given a graph G and a set of interesting vertices X, we say that a certificate C for X in G* preserves planarity *if, for any H such that $V(H) \cap V(G) \subseteq X$, if $G \cup H$ is planar, $C \cup H$ will also be planar.*

According to Definition 3.5, $C \cup H$ may be planar even when $G \cup H$ is not. As examples of planarity-preserving certificates, C may itself be a certificate for planarity; alternately, C may be a subgraph or minor of G.

The following lemma is proved in the companion paper [11] for global certificates and the proof can be easily extended to local and full certificates.

LEMMA 3.4 (see [11]). *Let $\mathcal{P}$ be a property for which there exist compressed certificates that preserve planarity. Then in linear time we can compute a compressed certificate for $\mathcal{P}$.*

We now describe an abstract version of our sparsification technique. We first consider global certificates (used to maintain global graph properties) and then show how the same technique can be made to work with local certificates, used to maintain local properties.

Let $\mathcal{P}$ be a property of planar graphs for which we can find compressed global certificates in time $T(n) = \Omega(n)$ and such that we can construct a data structure for testing property $\mathcal{P}$ in time $P(n)$ which can answer queries in time $Q(n)$. Then we wish to use these global certificates to maintain $\mathcal{P}$ quickly.

We construct a separator tree for the graph, by finding a set of $cn^{1/2}$ vertices (for some constant c) which splits the remaining graph into two components of less than $2n/3$ vertices each, and repeatedly split each component until there are $O(n^{1/2})$ components of size $O(n^{1/2})$ each; we call these the *leaf components*. This can all

be done in $O(n)$ time [23]. The resulting tree has height $O(\log n)$. When an edge connects two separator vertices, we arbitrarily choose which component to include it in, so each edge is included in a unique leaf component. Each time we insert a new edge, we will include its two endpoints in the separator for the node in the tree (if one exists) for which the two vertices are in the two separate components. After $O(n^{1/2})$ insertions, we reconstruct the separator tree, in amortized time $O(n^{1/2})$ per insertion.

At each node in the tree, the *interesting vertices* are those that are used either in the separator for that node or for separators at higher levels in the tree. Note that there will initially be at most

$$cn^{\frac{1}{2}} \sum_{i=0}^{\lceil \log n \rceil} \left(\frac{2}{3}\right)^{\frac{i}{2}} = O\left(n^{\frac{1}{2}}\right)$$

interesting vertices per tree node, and at most $O(n^{1/2})$ interesting vertices can be added by insertions before we reconstruct the tree. By the construction above, leaf components can share only interesting vertices. Furthermore, a vertex that is not interesting (in any leaf component) belongs to exactly one leaf component.

Each tree node corresponds to a subgraph which will be represented by a compressed global certificate for its interesting vertices. We form this global certificate by taking the union of the two compressed global certificates for the two daughter nodes (which by Lemma 3.2 is a global certificate for the graph at the node itself), and then computing a compressed global certificate of this union (which by Lemma 3.1 is also a global certificate for the node). We construct the data structure for testing property $\mathcal{P}$ using the global certificate at the tree root. This allows us to test property $\mathcal{P}$ in $Q(O(n^{1/2}))$ time.

When we reconstruct the separator tree, we must also reconstruct the global certificates, in $T(O(n^{1/2}))$ time per tree node. There are $O(n^{1/2})$ tree nodes, and we reconstruct after every $O(n^{1/2})$ insertions, so the amortized time per insertion is $T(O(n^{1/2}))$.

When we perform an insertion of an edge (x, y) that does not reconstruct the separator tree, we may move the two vertices x and y into the separator of a tree node N; then in all nodes descending from N and containing either of the two vertices, x and y may become newly interesting, and we must recompute the global certificates. However, this can happen only if either x or y was not interesting already. In other words, only the global certificates in the path between N and at most two leaves need to be updated. Furthermore, we must also recompute certificates for all tree nodes containing the newly inserted edge: these are exactly the nodes between N and the root of the separator tree. In either case, $O(\log n)$ tree nodes need recomputation, and the time to recompute certificates in each node is $T(O(n^{1/2}))$. Finally, we reconstruct the data structure for testing property $\mathcal{P}$ in the global certificate at the tree root in $P(O(n^{1/2}))$ time. The implementation of deletion is similar; here, too, we recompute global certificates in $O(\log n)$ nodes, in the same time bound.

Thus there is a fully dynamic algorithm for maintaining $\mathcal{P}$, which takes $P(O(n^{1/2}))$ $+ T(O(n^{1/2}))O(\log n)$ amortized time per edge insertion or deletion, and $Q(O(n^{1/2}))$ time per query. The amortized bound can be made worst-case by standard techniques of keeping two copies of the data structure, one of which can be gradually rebuilt while the other is being used.

In the companion paper [11], we develop a more complicated variant of this technique that allows us to save an $O(\log n)$ factor in the time bound above. The basic

idea is to use a separator tree which also partitions the interesting vertices evenly in the tree. In this way the nodes at lower levels of the separator tree will be able to have certificates smaller than $O(n^{1/2})$. In order to maintain this property of the separator tree we must then recompute lower-level separators after smaller numbers of updates.

DEFINITION 3.6. *Let G be a planar graph. A* balanced separator tree for G *is a separator tree such that a node at level i, $i \geq 0$, has at most $ab^i n^{1/2}$ interesting vertices, for some constants $a > 0$ and $0 < b < 1$.*

The proofs of the following two theorems are in [11].

THEOREM 3.1 (see [11]). *A balanced separator tree can be constructed in linear time.*

THEOREM 3.2 (see [11]). *Let $\mathcal{P}$ be a graph property for which we can find compressed global certificates in time $T(n) = \Omega(n)$ and such that we can construct, in $P(n)$ time, a data structure that tests property $\mathcal{P}$ in $Q(n)$ time. Then there is a fully dynamic algorithm for maintaining $\mathcal{P}$ in a planar graph subject to insertions and deletions preserving planarity, which takes $P(O(n^{1/2})) + T(O(n^{1/2}))$ amortized time per edge insertion or deletion and $Q(O(n^{1/2}))$ time per global query.*

We next describe how sparsification may apply to local properties such as vertex and edge connectivity.

THEOREM 3.3. *Let $\mathcal{P}$ be a local graph property for which we can find compressed local certificates in time $T(n) = \Omega(n)$ and such that we can construct a data structure for testing property $\mathcal{P}$ in time $Q(n)$. Then there is a fully dynamic algorithm for maintaining $\mathcal{P}$ in a planar graph, which takes amortized time $T(O(n^{1/2}))$ per edge insertion or deletion, and worst-case time $Q(O(n^{1/2})) + T(O(n^{1/2}))$ per local query.*

Proof. We use the balanced separator tree of Theorem 3.1, but this time we store at its nodes local certificates rather than global certificates. The amortized bound for updates follow now from Theorem 3.2. To test the local property $\mathcal{P}$ for two given vertices x and y, we first make x and y interesting vertices in the local certificate at the tree root. Once x and y are interesting, it is then easily verified that a local certificate for local property $\mathcal{P}$ is a global certificate for the simple property $\mathcal{P}(x, y)$. To make x and y interesting, we reconstruct the local certificates of all nodes containing either one of them. We do not reconstruct the separator tree even if the operation should normally do so. As in the proof of Theorem 3.2, this involves recomputing local certificates in $O(\log n)$ nodes in the separator tree of sizes increasing in a geometric series, and therefore can be done in $T(O(n^{1/2}))$ time. To answer a query regarding property $\mathcal{P}$ for vertices x and y, we construct the data structure for testing property $\mathcal{P}$ in the local certificate at the tree root in $P(O(n^{1/2}))$ time. Finally, we undo all the changes we made. □

Theorems 3.2 and 3.3 can be combined as follows.

THEOREM 3.4. *Let $\mathcal{P}$ be a local graph property for which we can find compressed full certificates in time $T(n) = \Omega(n)$ and such that we can construct a data structure for testing property $\mathcal{P}$ in time $Q(n)$. Then there is a fully dynamic algorithm for maintaining $\mathcal{P}$ in a planar graph, which takes amortized time $T(O(n^{1/2}))$ per edge insertion or deletion, and worst-case time $Q(O(n^{1/2})) + T(O(n^{1/2}))$ per either global or local query.*

4. Properties of certificates for edge connectivity. Let G be an undirected graph, with interesting vertices $X \subseteq V(G)$. In this section we give necessary and sufficient conditions for a graph C to be a full certificate of k-edge-connectivity for X in G. Namely, we will show that it is crucial to keep information about the connectivity edge-cuts that involve only the interesting vertices in X. This will be exploited in the

next sections in order to build our full certificates for edge connectivity.

Let V_1 and V_2 be any two nonempty disjoint subsets of vertices in G. We say that a set of edges $E' \subseteq E(G)$ *disconnects* V_1 *and* V_2 if removing all the edges in E' from G leaves no path between vertices in V_1 and vertices in V_2. We denote by $\lambda_{V_1,V_2}(G)$ the *minimum* number of edges of G whose removal disconnects V_1 and V_2. Note that $\lambda_{V_1,V_2}(G) = \lambda_{V_2,V_1}(G)$. When $V_1 = \{u\}$ and $V_2 = \{v\}$, we obtain the definition of edge connectivity between vertices u and v. The edge connectivity $\lambda(G)$ of G satisfies the equality

$$\lambda(G) = \min_{u,v \in V(G)} \{\lambda_{u,v}(G)\} = \min_{\substack{\emptyset \subset V_1, V_2 \subset V(G) \\ V_1 \cap V_2 = \emptyset}} \{\lambda_{V_1,V_2}(G)\}.$$

Let $\emptyset \subset R \subset X$; then we denote by $\lambda_R(G)$ the quantity $\lambda_{R,X-R}(G)$. If $R = \emptyset$ or $R = X$, we instead let $\lambda_R(G)$ denote the minimum number of edges that must be removed from G in order to disconnect X from at least one other vertex in $[V(G) - X]$.

We can use this notation to characterize full certificates of edge connectivity. Before doing this, we need a technical lemma. Often the easiest way to show that a graph C is a certificate of edge connectivity is to show that appropriate edge-cuts exist. These edge-cuts are edge-cuts of a graph that is the union of two other graphs. The following lemma gives sufficient conditions for constructing an edge-cut of $G \cup H$ from an edge-cut of G and an edge-cut of H.

LEMMA 4.1. *Let G and H be graphs such that $V(G) \cap V(H) \subseteq X$. Furthermore, let γ_G and γ_H be edge-cuts of G and H, respectively. Suppose that γ_G divides G into G_1 and G_2 and that γ_H divides H into H_1 and H_2. Then, if $(V(G_1) \cap V(H_2)) \cup (V(G_2) \cap V(H_1)) = \emptyset$, then $\gamma = \gamma_G \cup \gamma_H$ is an edge-cut of $G \cup H$ that divides it into $G_1 \cup H_1$ and $G_2 \cup H_2$.*

Proof. No edge outside $\gamma_G \cup \gamma_H$ can cross from $G_1 \cup H_1$ to $G_2 \cup H_2$. For if such an edge belonged to G, it would have to cross from $(G_1 \cup H_1) \cap G = G_1$ to $(G_2 \cup H_2) \cap G = G_2$ and would therefore belong to γ_G, and symmetrically, if such an edge belonged to H, it would belong to γ_H. ∎

Suppose that we are interested in k-edge-connectivity for $k \leq \mathcal{K}$. Then, for C to be a certificate of k-edge-connectivity for X in G, the edge-cuts with k or fewer edges in G should correspond to edge-cuts of the same size in C. Making this formal, we have the following.

LEMMA 4.2. *If C is a full certificate of k-edge-connectivity for X in G, for every $k \leq \mathcal{K}$, then, for any subset $\emptyset \subseteq R \subseteq X$ of interesting vertices,*

$$\min\{\lambda_R(G), \mathcal{K}\} = \min\{\lambda_R(C), \mathcal{K}\}. \tag{4.1}$$

Proof. Suppose that for some $\emptyset \subseteq R \subseteq X$, $\min\{\lambda_R(G), \mathcal{K}\} \neq \min\{\lambda_R(C), \mathcal{K}\}$. We want to find a graph H that shows that C is not a full certificate of k-edge-connectivity for X in G. Note that either $\lambda_R(C) < \mathcal{K}$ or $\lambda_R(G) < \mathcal{K}$.

If $R = \emptyset$ or $R = X$, let H have vertices $X \cup \{z\}$ and $\mathcal{K}$ edges between z and each vertex in X. Any minimal edge-cut of $G \cup H$ that contains any edges in H must then contain all $\mathcal{K}$ of the edges between z and some vertex $x \in X$. Any edge-cut that does not contain any edges from H must separate X from some other vertex in G. Thus, if $\lambda(G \cup H) < \mathcal{K}$, then $\lambda(G \cup H) = \lambda_\emptyset(G) = \lambda_R(G)$. Similarly, the edge connectivity $\lambda(C \cup H)$ is $\lambda_\emptyset(C) = \lambda_R(C)$ unless $\lambda_\emptyset(C) > \mathcal{K}$, in which case the edge connectivity is at least $\mathcal{K}$. Therefore, since $\lambda_\emptyset(C) < \mathcal{K}$ or $\lambda_\emptyset(G) < \mathcal{K}$, we have that

$\lambda(G \cup H) \neq \lambda(C \cup H)$, so C is not a global certificate of k-edge-connectivity for some $k \leq \mathcal{K}$.

Alternately, it may be the case that $\emptyset \subset R \subset X$. In this case, construct H with vertices $X \cup \{z_1, z_2\}$, $\mathcal{K}$ edges from z_1 to each vertex in R, and $\mathcal{K}$ edges from z_2 to each vertex in $X - R$. Again, no minimal edge-cut separating z_1 and z_2 in $G \cup H$ contains an edge of H unless it contains all $\mathcal{K}$ of the edges between two vertices in H. Thus, any edge-cut separating z_1 and z_2 and containing no edges of H must separate R and $X - R$. This implies that if $\lambda_{z_1,z_2}(G \cup H) < \mathcal{K}$, then $\lambda_{z_1,z_2}(G \cup H) = \lambda_R(G)$. Similarly, the edge-connectivity between z_1 and z_2 in $C \cup H$ is $\lambda_R(C)$ unless $\lambda_R(C) > \mathcal{K}$, in which case the edge-connectivity is at least $\mathcal{K}$. Therefore, since $\lambda_R(C) < \mathcal{K}$ or $\lambda_R(G) < \mathcal{K}$, we have that $\lambda_{z_1,z_2}(G \cup H) \neq \lambda_{z_1,z_2}(C \cup H)$, so C is not a local certificate of k-edge-connectivity for some $k \leq \mathcal{K}$. ∎

Not only is (4.1) a necessary condition for C to be a full certificate, it is also sufficient. First, let us see that it is a sufficient condition for C to be a local certificate.

LEMMA 4.3. *Let $\mathcal{K}$ be a given integer. Let G be a given graph, with interesting vertices $X \subseteq V(G)$. If for any proper subset R of interesting vertices, $\emptyset \subset R \subset X$:*

$$\min\{\lambda_R(G), \mathcal{K}\} = \min\{\lambda_R(C), \mathcal{K}\},$$

then C is a local certificate of k-edge-connectivity for X in G, for every $k \leq \mathcal{K}$.

Proof. This is equivalent to showing that, for any graph H such that $V(H) \cap V(G) \subseteq X$, $V(H) \cap V(C) \subseteq X$, and for any two vertices $x, y \in H$, the following is true:

1. if $\lambda_{x,y}(G \cup H) < \mathcal{K}$, then $\lambda_{x,y}(C \cup H) \leq \lambda_{x,y}(G \cup H)$, and
2. if $\lambda_{x,y}(C \cup H) < \mathcal{K}$, then $\lambda_{x,y}(G \cup H) \leq \lambda_{x,y}(C \cup H)$.

Now we would like to see that 1 holds. Let γ be a minimum edge-cut that disconnects x from y in $G \cup H$. Since it is a minimal edge-cut, it separates $G \cup H$ into exactly two pieces S_1 and S_2. Without loss of generality, assume that $x \in S_1$ and $y \in S_2$. Let $X_1 = S_1 \cap X$, $X_2 = S_2 \cap X$, $G_1 = S_1 \cap G$, $G_2 = S_2 \cap G$, $H_1 = S_1 \cap H$, and $H_2 = S_2 \cap H$. Note that $x \in H_1$ and $y \in H_2$. Further, let $\gamma_G = \gamma \cap E(G)$ and $\gamma_H = \gamma \cap E(H)$.

If $X_2 = \emptyset$, then $X = X_1$, $G_2 = \emptyset$, and $G = G_1$. So, $\gamma_G = \emptyset$ and $\gamma = \gamma_H$. In this case, $\gamma = \gamma_H$ disconnects $G \cup H$ into $G \cup H_1$ and H_2, with $x \in H_1$ and $y \in H_2$. Since $V(C) \cap V(H) \subseteq X = X_1$, replacing G with C yields that $\gamma = \gamma_H$ disconnects $C \cup H$ into $C \cup H_1$ and H_2, with $x \in (G \cup H_1)$ and $y \in H_2$. Hence, the very same edge-cut γ disconnects x and y in $C \cup H$. Similarly, we can assume that $X_1 \neq \emptyset$.

If $|\gamma| \geq \mathcal{K}$, there is nothing to prove. Otherwise, we need to construct γ' as an edge-cut of $C \cup H$ that disconnects x from y and such that $|\gamma'| \leq |\gamma|$. Note that γ_G disconnects X_1 from X_2 in G, so $|\gamma_G| \geq \lambda_{X_1,X_2}(G)$. Since $X = X_1 \cup X_2$, $\lambda_{X_1,X_2}(G) = \lambda_{X_1}(G)$. Moreover, since $|\gamma_G| \leq |\gamma| < \mathcal{K}$, $\lambda_{X_1}(G) = \lambda_{X_1}(C) = \lambda_{X_1,X_2}(C)$. This means that there is an edge-cut γ_C of C that disconnects X_1 from X_2 such that $|\gamma_C| \leq |\gamma_G|$. Thus, if $\gamma' = \gamma_H \cup \gamma_C$, then $|\gamma'| \leq |\gamma|$.

Now we would like to see that γ' disconnects x from y in $H \cup C$. Define C_1 and C_2 so that γ_C divides C into C_1 and C_2. Note that γ_C and γ_H both partition X in the same way, so we can assume that $X_1 \subseteq V(C_1)$ and $X_2 \subseteq V(C_2)$. Since $V(C) \cap V(H) \subseteq X$, this means that $(V(C_1) \cup V(H_1)) \cap (V(C_2) \cup V(H_2)) = \emptyset$, so Lemma 4.1 says that γ' is an edge-cut of $C \cup H$ that disconnects x from y.

To prove that 2 holds, we use exactly the same proof with the roles of G and C switched. Therefore, C is a local certificate of k-edge-connectivity for X in G. ∎

If the condition also holds when $R = \emptyset$ or $R = X$, then C is also a global certificate.

LEMMA 4.4. *Let $\mathcal{K}$ be a given integer. Let G be a given graph, with interesting vertices $X \subseteq V(G)$. If for any subset R of interesting vertices, $\emptyset \subseteq R \subseteq X$:*

$$\min\{\lambda_R(G), \mathcal{K}\} = \min\{\lambda_R(C), \mathcal{K}\},$$

then C is a global certificate of k-edge-connectivity for X in G, for every $k \leq \mathcal{K}$.

Proof. Let H be any graph such that $V(H) \cap V(G) \subseteq X$. Suppose that γ is a minimum k-edge-cut of $G \cup H$ that divides the graph into S_1 and S_2, and that $k < \mathcal{K}$. If both S_1 and S_2 contain vertices in H, then the fact that C is a local certificate of k-edge-connectivity for X in G shows that there is a k-edge-cut of $C \cup H$. Alternately, if S_1 (say) contains no vertices from H, then γ contains only edges from G, since every edge in a minimal edge-cut connects a vertex in S_1 to a vertex in S_2. This means that $\lambda_\emptyset(G) \leq k$. Since $k < \mathcal{K}$, $\lambda_\emptyset(G) = \lambda_\emptyset(C) \leq k$, so $C \cup H$ has a k-edge-cut as well.

Conversely, suppose that γ' is a minimum k-edge-cut of $C \cup H$ that divides the graph into S_1 and S_2, and that $k < \mathcal{K}$. If both S_1 and S_2 contain vertices in H, then the fact that C is a local certificate of k-edge-connectivity for X in G shows that there is a k-edge-cut of $G \cup H$. Alternately, if S_1 (say) contains no vertices from H, then γ' contains only edges from C. This means that $\lambda_\emptyset(C) \leq k$. Since $k < \mathcal{K}$, $\lambda_\emptyset(C) = \lambda_\emptyset(G) \leq k$, so $G \cup H$ has a k-edge-cut as well. Therefore, C is a global, as well as local, certificate of k-edge-connectivity provided that $k \leq \mathcal{K}$. ∎

Putting these three lemmas together, we have Theorem 4.1.

THEOREM 4.1. *Let $\mathcal{K}$ be a given integer. Let G be a given graph, with interesting vertices $X \subseteq V(G)$. C is a full certificate of k-edge-connectivity for X in G, for every $k \leq \mathcal{K}$, if and only if for any subset R of interesting vertices, $\emptyset \subseteq R \subseteq X$:*

$$\min\{\lambda_R(G), \mathcal{K}\} = \min\{\lambda_R(C), \mathcal{K}\}.$$

4.1. Split graphs and certificates. We now prove some properties about splitting 2-edge-connected graphs and computing their certificates for edge connectivity. Let G be a 2-edge-connected graph, and let $\{e_1, e_2\}$ be a 2-edge-cut in G. Let G_1 and G_2 be the two graphs obtained from G after the deletion of $\{e_1, e_2\}$, and let $e_1 = (u_1, u_2)$ and $e_2 = (v_1, v_2)$ be such that u_1, v_1 are in G_1 and u_2, v_2 are in G_2. We call this a *split* and call $G_1 \cup \{(u_1, v_1)\}$ and $G_2 \cup \{(u_2, v_2)\}$ the two *split graphs of G with respect to the 2-edge-cut $\{e_1, e_2\}$*. Note that $\{(u_1, v_1)\}$ and $\{(u_2, v_2)\}$ are not originally edges of G: they are called the *virtual edges* associated with the split. The operation inverse to a split is called a *merge*: it takes two split graphs with respect to the same 2-edge-cut and merges them back together, yielding the original graph. We observe that splits and merges preserve planarity. Indeed, the split graphs $G_1 \cup \{(u_1, v_1)\}$ and $G_2 \cup \{(u_2, v_2)\}$ can be obtained from G by means of edge contractions (for instance, $G_1 \cup \{(u_1, v_1)\}$ can be obtained after contracting e_1 and all the edges in G_2). Hence, if G is planar, the split graphs $G_1 \cup \{(u_1, v_1)\}$ and $G_2 \cup \{(u_2, v_2)\}$ obtained after a split are planar. Conversely, if the split graphs are planar, the graph obtained after a merge will be planar. Figure 3 illustrates splits and merges.

LEMMA 4.5. *Let G be a 2-edge-connected graph, and let $k \geq 2$ be an integer. Let $\{e_1, e_2\}$ be any 2-edge-cut of G. For $i = 1, 2$ let us denote by $G_i \cup \{(u_i, v_i)\}$ the split graphs of G with respect to the 2-edge-cut $\{e_1, e_2\}$. Let x and y be any two vertices in the same split graph $G_i \cup \{(u_i, v_i)\}$. Then x and y are k-edge-connected in $G_i \cup \{(u_i, v_i)\}$ if and only if they are k-edge-connected in G.*

Proof. Without loss of generality, let x and y be any two vertices of $G_1 \cup \{(u_1, v_1)\}$. Since G is 2-edge-connected, any two edges of G are contained in a cycle. In particular,

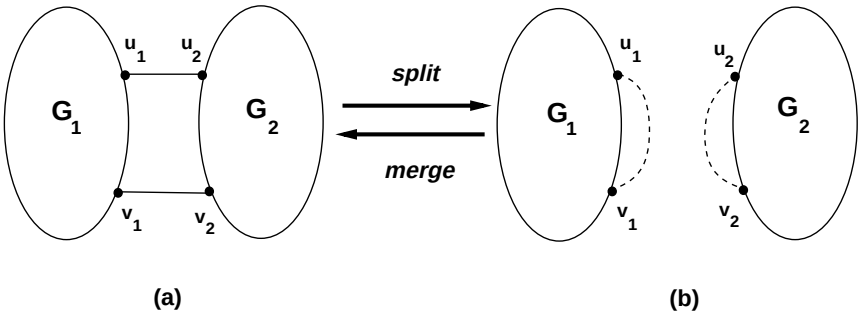

FIG. 3. *Splitting and merging a 2-edge-connected graph and its split graphs. Virtual edges are dashed.*

there is a cycle in G containing edges e_1 and e_2. This implies that there exists a path π_1 between vertices u_1 and v_1 that is entirely contained in G_1, and a path π_2 between u_2 and v_2 that is entirely contained in G_2.

Assume x and y are k-edge-connected in $G_1 \cup \{(u_1, v_1)\}$. Then there are k edge-disjoint paths in $G_1 \cup \{(u_1, v_1)\}$ between x and y, and at most one of them can use the edge (u_1, v_1). If none of these paths use (u_1, v_1), there will be k edge-disjoint paths between x and y in G. If one of these paths use (u_1, v_1), then the path in G containing e_1, π_2, and e_2 in place of (u_1, v_1) is a path between x and y that is edge-disjoint from the other $(k-1)$ paths in G_1. In both cases, x and y are k-edge-connected in G.

Conversely, assume that x and y are k-edge-connected in G. Again, at most one of the k edge-disjoint paths between x and y can go through G_2. If all the k-edge-disjoint paths between x and y are contained in G_1, x and y are k-edge-connected in $G_1 \cup \{(u_1, v_1)\}$ too. If one of these paths go through G_2, replacing this portion of the path with the edge (u_1, v_1) gives a path in $G_1 \cup \{(u_1, v_1)\}$ which is edge-disjoint from the other $(k-1)$ paths. □

The following corollary is an easy consequence of Lemma 4.5.

COROLLARY 4.1. *Let G be a 2-edge-connected graph, and let $\{e_1, e_2\}$ be any 2-edge-cut of G. The split graphs of G with respect to $\{e_1, e_2\}$ are 2-edge-connected.*

Since the split graphs of G are 2-edge-connected, the same splitting can be applied recursively to the split graphs of G, and to their split graphs, and so on. When no further splits are possible, each split graph left is 3-edge-connected and corresponds to exactly one 3-edge-connected class of the original graph. This gives another way of defining the cactus tree of a graph with edge connectivity 2. We call these final split graphs the 3-*edge-connected components* of G. We point out here a difference between 2-edge-connectivity and 3-edge-connectivity. For 2-edge-connectivity, the subgraph induced by a 2-edge-connected class is 2-edge-connected and coincides with a 2-edge-connected component. On the contrary, the subgraph induced by a 3-edge-connected class may differ from a 3-edge-connected component. Indeed, the subgraph induced by a 3-edge-connected class is not necessarily even connected, while the addition of the virtual edges makes a 3-edge-connected component 3-edge-connected. We remark that this decomposition of a 2-edge-connected graph is similar to the decomposition of a biconnected graph into its triconnected components [25], and it is implicit in the work of Dinitz [5].

Let G be a 2-edge-connected graph, and consider the following operation: split G and compute a full certificate of k-edge-connectivity, $k \geq 2$, for one of its split graphs; then merge this certificate with the other split graph. We will prove that by doing

so we obtain a full certificate of k-edge-connectivity, $k \geq 2$, for the original graph G. This property will be useful later on. We now summarize the notation used.

- G: a 2-edge-connected graph;
- $X \subseteq V(G)$: the interesting vertices of G;
- $\{e_1, e_2\}$: a 2-edge-cut in G, with $e_1 = (u_1, u_2)$ and $e_2 = (v_1, v_2)$;
- $\widehat{G}_1 = G_1 \cup \{(u_1, v_1)\}$, $\widehat{G}_2 = G_2 \cup \{(u_2, v_2)\}$: the split graphs of G with respect to $\{e_1, e_2\}$; $u_1, v_1 \in V(G_1)$ and $u_2, v_2 \in V(G_2)$;
- $X_1 = (X \cap V(G_1)) \cup \{u_1, v_1\}$: the interesting vertices in G_1 augmented with u_1 and v_1;
- $\widehat{C}_1 = C_1 \cup \{(u_1, v_1)\}$: a full certificate of k-edge-connectivity for X_1 in $\widehat{G}_1$ (note that this certificate keeps the virtual edge (u_1, v_1) of $\widehat{G}_1$);
- H: any graph such that $V(H) \cap V(G) \subseteq X$;
- $F = G_2 \cup H \cup \{e_1, e_2\}$. Namely, the vertices of F are u_1, v_1 plus the vertices of G_2 and H; the edges of F are e_1, e_2 plus the edges of G_2 and H.

Note that $G = G_1 \cup \{e_1, e_2\} \cup G_2$. We further define $C = C_1 \cup \{e_1, e_2\} \cup G_2$. The point of all of this is that C is a full certificate of k-edge-connectivity for X in G. To prove this, we first prove that it is a local certificate and then prove that it is a global certificate.

LEMMA 4.6. *Let $\mathcal{K}$ be a given integer. Let $\widehat{C}_1 = C_1 \cup \{(u_1, v_1)\}$ be a local certificate of k-edge-connectivity for X_1 in $\widehat{G}_1 = G_1 \cup \{(u_1, v_1)\}$ for every $k \leq \mathcal{K}$. Then $C = C_1 \cup \{e_1, e_2\} \cup G_2$ is a local certificate of k-edge-connectivity for X in G for every $k \leq \mathcal{K}$.*

Proof. Again, the idea behind the proof is to show that the appropriate edge-cuts exist. Specifically, we need to show that for any H such that $V(G) \cap V(H) \subseteq X$, $V(C) \cap V(H) \subseteq X$, and for any $x, y \in V(H)$:

1. if $\lambda_{x,y}(G \cup H) < \mathcal{K}$ then $\lambda_{x,y}(C \cup H) \leq \lambda_{x,y}(G \cup H)$, and
2. if $\lambda_{x,y}(C \cup H) < \mathcal{K}$ then $\lambda_{x,y}(G \cup H) \leq \lambda_{x,y}(C \cup H)$.

Suppose that γ is a minimum edge-cut that disconnects x from y in $G \cup H$ and that $|\gamma| < \mathcal{K}$. Since γ is a minimum edge-cut, it separates $G \cup H$ into two pieces $S^{(x)}$ and $S^{(y)}$, with $x \in S^{(x)}$ and $y \in S^{(y)}$. Let $G_1^{(x)} = G_1 \cap S^{(x)}$, $G_1^{(y)} = G_1 \cap S^{(y)}$, $F^{(x)} = F \cap S^{(x)}$, $F^{(y)} = F \cap S^{(y)}$, $X_1^{(x)} = X_1 \cap S^{(x)}$, and $X_1^{(y)} = X_1 \cap S^{(y)}$. Note that $G_1 = G_1^{(x)} \cup G_1^{(y)}$, $X_1 = X_1^{(x)} \cup X_1^{(y)}$, and $F = F^{(x)} \cup F^{(y)}$. Since $x \in (S^{(x)} \cap H)$ and $y \in (S^{(y)} \cap H)$, we must have $x \in F^{(x)}$ and $y \in F^{(y)}$. Also let $\gamma_{G_1} = \gamma \cap G_1$ and let $\gamma_F = \gamma - \gamma_{G_1}$. Note that γ_F disconnects $X_1^{(x)} \cap V(F)$ from $X_1^{(y)} \cap V(F)$ in F.

If γ_{G_1} is empty, then $\gamma_F = \gamma$. G_1 is connected by the assumption that G is biconnected, so one of $G_1^{(x)}$ or $G_1^{(y)}$ is empty and the other one contains all of G_1. Without loss of generality $G_1^{(x)} = \emptyset$. C_1 is not reachable from x in $C \cup H - \gamma$, since any path connecting x to C_1 in $C \cup H$ would connect x to G_1 in $G \cup H$, and so must be cut by γ. Then x is separated from y in $C \cup H - \gamma$, since no path from x can go through C_1, and since any path avoiding C_1 would also exist in $G \cup H - \gamma$.

Alternately, assume that γ_{G_1} is not empty. We claim that in this case neither $X_1^{(x)}$ nor $X_1^{(y)}$ can be empty. Indeed, if either $X_1^{(x)}$ or $X_1^{(y)}$ were empty, then the removal of all the edges of γ from $G \cup H$ would leave all the vertices of X_1 still connected. Since $V(H) \cap V(G_1) \subseteq X_1$, any path of $G \cup H$ between $x \in V(H)$ and $y \in V(H)$ that contains an edge of G_1 must also contain a vertex of X_1 (recall that, by definition, $u_1, v_1 \in X_1$). Thus, in this case, γ_F would be an edge-cut of $G \cup H$ that disconnects x from y. Since γ is by assumption a minimum edge-cut, this implies by contradiction that neither $X_1^{(x)}$ nor $X_1^{(y)}$ can be empty.

Without loss of generality, assume that $u_1 \in G_1^{(x)}$. We have two cases depending on whether $v_1 \in G_1^{(x)}$ or $v_1 \in G_1^{(y)}$. Suppose first that $v_1 \in G_1^{(x)}$. In this case, γ_{G_1} is an edge-cut of $\widehat{G_1}$ that disconnects $X_1^{(x)}$ from $X_1^{(y)}$. Consequently, $\lambda_{X_1^{(x)}}(\widehat{G_1}) \leq |\gamma_{G_1}|$. Since $\widehat{C_1}$ is a local certificate of k-edge-connectivity for X_1 in $\widehat{G_1}$, and $|\gamma_{G_1}| \leq |\gamma| < \mathcal{K}$, by Theorem 4.1 we have that $\lambda_{X_1^{(x)}}(\widehat{C_1}) = \lambda_{X_1^{(x)}}(\widehat{G_1}) \leq |\gamma_{G_1}|$. This is equivalent to saying that there is an edge-cut $\gamma_{\widehat{C_1}}$ of $\widehat{C_1}$ that disconnects $X_1^{(x)}$ from $X_1^{(y)}$ and such that $|\gamma_{\widehat{C_1}}| = \lambda_{X_1^{(x)}}(\widehat{C_1}) \leq |\gamma_{G_1}|$. Since we can take $\gamma_{\widehat{C_1}}$ to be a minimal edge-cut, it does not contain the edge (u_1, v_1), so it is an edge-cut of C_1. Thus $\gamma' = \gamma_{\widehat{C_1}} \cup \gamma_F$ is a set of edges of $C \cup H$ with $|\gamma'| \leq |\gamma|$. Moreover, $\gamma_{\widehat{C_1}}$ divides C_1 into $C_1^{(x)}$ and $C_1^{(y)}$. Since $X_1^{(x)} \subseteq V(C_1^{(x)})$ and $X_1^{(y)} \subseteq V(C_1^{(y)})$, we have that $C_1^{(x)} \cap F^{(y)} = \emptyset$ and $C_1^{(y)} \cap F^{(x)} = \emptyset$, so Lemma 4.1 shows that $\gamma' = \gamma_{\widehat{C_1}} \cup \gamma_F$ is an edge-cut of $C_1 \cup F = C \cup H$ that disconnects x from y.

Alternately, it could be the case that $v_1 \in G_1^{(y)}$. In this case, γ_{G_1} is not an edge-cut of $\widehat{G_1}$, but $\gamma_{G_1} \cup \{(u_1, v_1)\}$ is such an edge-cut. Again, Theorem 4.1 says that there is an edge-cut $\widehat{\gamma_{\widehat{C_1}}}$ that disconnects $X_1^{(x)}$ from $X_1^{(y)}$ in $\widehat{C_1}$ such that $|\widehat{\gamma_{\widehat{C_1}}}| \leq |\gamma_{\widehat{C_1}}| + 1$. Moreover, since $u_1 \in X_1^{(x)}$ and $v_1 \in X_1^{(y)}$, the edge-cut $\widehat{\gamma_{\widehat{C_1}}}$ contains the edge (u_1, v_1). Let $\gamma_{\widehat{C_1}} = \widehat{\gamma_{\widehat{C_1}}} - \{(u_1, v_1)\}$. Then $\gamma' = \gamma_{\widehat{C_1}} \cup \gamma_F$ is a set of edges in $C \cup H$ with $|\gamma'| \leq |\gamma|$. Moreover, $\gamma_{\widehat{C_1}}$ disconnects $X_1^{(x)}$ from $X_1^{(y)}$, so again, by Lemma 4.1, γ' is an edge-cut that disconnects x from y in $C \cup H$.

Therefore, in either case, γ' is the desired edge-cut, and the first property holds. Since $\widehat{C_1}$ is a certificate of k-edge-connectivity for X_1 in $\widehat{G_1}$, it is the case that $\widehat{G_1}$ is a certificate for k-edge-connectivity of X_1 in $\widehat{C_1}$. Therefore, the same proof can be used to prove property 2, and C is a global certificate of k-edge-connectivity for X in G. ∎

To prove that C is a full certificate, we also need to prove that it is a global certificate. The proof of this fact is annoyingly similar to the proof that C is a local certificate, but there seems to be no clear way to combine the proofs.

LEMMA 4.7. *Let $\mathcal{K}$ be a given integer. Let $\widehat{C}_1 = C_1 \cup \{(u_1, v_1)\}$ be a full certificate of k-edge-connectivity for X_1 in $\widehat{G}_1 = G_1 \cup \{(u_1, v_1)\}$ for every $k \leq \mathcal{K}$. Then $C = C_1 \cup \{e_1, e_2\} \cup G_2$ is a global certificate of k-edge-connectivity for X in G for every $k \leq \mathcal{K}$.*

Proof. Again it suffices to show that for any H with $V(H) \cap V(G) \subseteq X$, the following two properties hold:

1. if $\lambda(C \cup H) \leq \mathcal{K}$ then $\lambda(G \cup H) \leq \lambda(C \cup H)$, and
2. if $\lambda(G \cup H) \leq \mathcal{K}$ then $\lambda(C \cup H) \leq \lambda(G \cup H)$.

To prove property 1, let γ be a minimum edge-cut of $C \cup H$ that divides $C \cup H$ into S^* and S^{**}. Since γ is a minimum edge-cut in $C \cup H$, it must be of cardinality $\lambda(C \cup H)$: assume that $\lambda(C \cup H) \leq \mathcal{K}$. To prove that $\lambda(G \cup H) \leq \lambda(C \cup H)$, we distinguish several cases according to S^*, S^{**}, and H.

Assume first that both S^* and S^{**} contain vertices of H: thus, there exist two vertices, say x and y, in $V(H)$ that are separated by γ. By Lemma 4.6, we know that C is a local certificate of k-edge-connectivity for X in G, for every $k \leq \mathcal{K}$. Then, if $\lambda(C \cup H) \leq \mathcal{K}$, by Definition 3.2 there is an edge-cut γ' separating x and y in $G \cup H$ of cardinality exactly $\lambda(C \cup H)$. As a result, the minimum edge-cut of $G \cup H$ must have cardinality less than or equal to the cardinality of γ' and hence $\lambda(G \cup H) \leq \lambda(C \cup H)$.

Alternately, it may be the case that $V(H) \subseteq V(S^*)$. In this case, γ contains only edges in C. There are three subcases, depending on how many of the vertices v_1 and u_1 are in $V(S^{**})$. If neither vertex is in $V(S^{**})$, then either $\gamma \subseteq G_2 \cup \{e_1, e_2\}$, in which case γ is also an edge-cut of $G \cup H$, or $\gamma \subseteq C_1$. In the latter case, γ is also an edge-cut of $\widehat{C_1}$, so there is an edge-cut γ' of $\widehat{G_1}$ that is also an edge-cut of $G \cup H$ such that $|\gamma'| \leq |\gamma|$, since $\widehat{C_1}$ is a global certificate of k-edge-connectivity for X_1 in $\widehat{G_1}$.

Another possibility is that both u_1 and v_1 are in S^{**}. Here γ contains edges from both C_1 and G_2, since it is minimal. Let $\gamma_{C_1} = \gamma \cap C_1$ and $\gamma_F = \gamma - \gamma_{C_1}$. Then γ_{C_1} disconnects $\{u_1, v_1\}$ from the rest of X_1 in $\widehat{C_1}$. By Theorem 4.1, there is an edge-cut γ_{G_1} that disconnects $\{u_1, v_1\}$ from the rest of X_1 in $\widehat{G_1}$. Moreover, $|\gamma_{G_1}| \leq |\gamma_{C_1}|$, so if $\gamma' = \gamma_{G_1} \cup \gamma_F$, then $|\gamma'| \leq |\gamma|$. Moreover, since γ_F disconnects $\{u_1, v_1\}$ from X_2 in F, Lemma 4.1 says that γ' is an edge-cut of $G \cup H$.

The final possibility is that one vertex is in S^* and the other is in S^{**}. Without loss of generality, assume that $u_1 \in S^{**}$ but $v_1 \in S^*$. Again let $\gamma_{C_1} = \gamma \cap C_1$ and $\gamma_F = \gamma - \gamma_{C_1}$. Here, let $\widehat{\gamma_{C_1}} = \gamma_{c_1} \cup \{(u_1, v_1)\}$. Then $\widehat{\gamma_{C_1}}$ disconnects u_1 from the rest of X_1 in $\widehat{C_1}$. Therefore, by Theorem 4.1, there is an edge-cut $\widehat{\gamma_{G_1}}$ that disconnects u_1 from the rest of X_1 in $\widehat{G_1}$ such that $|\widehat{\gamma_{G_1}}| \leq |\widehat{\gamma_{C_1}}|$. Moreover, since $\widehat{\gamma_{G_1}}$ disconnects u_1 from v_1, $(u_1, v_1) \in \widehat{\gamma_{G_1}}$, so let $\gamma_{G_1} = \widehat{\gamma_{G_1}} - \{(u_1, v_1)\}$. Since $(u_1, v_1) \notin G_1$, the edge-cut γ_{G_1} disconnects u_1 from the rest of X_1 in G_1. Moreover, γ_F disconnects u_1 from X_2 in F, so, by applying Lemma 4.1 yet again, we discover that $\gamma' = \gamma_{G_1} \cup \gamma_F$ is an edge-cut of $G \cup H$. It is also the case that $|\gamma'| \leq |\gamma|$. Therefore, we have seen that in all cases $\lambda(G \cup H) \leq \lambda(C \cup H)$, and property 1 holds.

Since $\widehat{C_1}$ is a certificate of k-edge-connectivity for X_1 in $\widehat{G_1}$, it is the case that $\widehat{G_1}$ is a certificate for k-edge-connectivity of X_1 in $\widehat{C_1}$. Therefore, the same proof can be used to prove property 2, and C is a global certificate of k-edge-connectivity for X in G. □

THEOREM 4.2. *Let $\mathcal{K}$ be a given integer, let $X \subseteq V(G)$ be the interesting vertices of G, and let $X_1 = (X \cap V(G_1)) \cup \{u_1, v_1\}$: the interesting vertices in G_1 augmented with u_1 and v_1. Let $\widehat{C}_1 = C_1 \cup \{(u_1, v_1)\}$ be a full certificate of k-edge-connectivity for X_1 in $\widehat{G}_1 = G_1 \cup \{(u_1, v_1)\}$ for every $k \leq \mathcal{K}$. Then $C = C_1 \cup \{e_1, e_2\} \cup G_2$ is a full certificate of k-edge-connectivity for X in G for every $k \leq \mathcal{K}$.*

5. Edge connectivity. In this section we present compressed full certificates for 3- and 4-edge-connectivity. Using these certificates in Theorem 3.4 yields fast, fully dynamic algorithms for maintaining information about 3- and 4-edge-connectivity in a planar graph. Our certificates for edge connectivity will be constructed using the linear time algorithm of Lemma 3.4, so they will be required to preserve planarity as well as edge connectivity. To use this construction, we merely need to show that such certificates exist, by describing an algorithm for finding them. However, we need not analyze the time bounds of this algorithm, since Lemma 3.4 will then provide an alternate algorithm with linear complexity.

Let X be the set of interesting vertices in G. Our certificates are based upon a repeated compression of G during different phases. In the first phase we shrink some edges of G so as to reduce the number of 2-edge-connected classes (and thus components) to $O(|X|)$. After this phase we could easily get compressed full certificates for 2-edge-connectivity; however, as we mentioned earlier, these certificates for 2-edge-connectivity would yield time bounds that are worse than the polylogarithmic bounds we obtain in the companion paper [11], and so we will not describe them. In the second phase, we compress each 2-edge-connected component left so as to reduce

the total number of 3-edge-connected classes (and thus components) to $O(|X|)$: compressed full certificates for 3-edge-connectivity can be computed after this phase. In the third phase, we similarly reduce the number of 4-edge-connected classes. Finally, we compress each 4-edge-connected class left so as to reduce the overall size of the graph to $O(|X|)$.

To carry out these compressions, we use the decomposition of a k-edge-connected graph into its $(k+1)$-edge-connected classes described by the cactus tree (see section 2). In each phase we use a compression that follows many of the same ideas used in the companion paper [11] to compute the minimum spanning forest certificates of size $O(|X|)$. We recall here that for the minimum spanning forest certificate we started with a tree T and then repeatedly applied the following two rules until no more rule could be applied.

(1) If $v \in T - X$ touches a single edge (u, v) in T, remove both v and edge (u, v).
(2) If $v \in T - X$ touches two edges (u, v) and (v, w) in T, remove v and replace the two edges by a single edge (u, w).

Rule (1) cuts uninteresting branches (parts of the graph not containing interesting vertices) and rule (2) shortcuts uninteresting paths (paths not containing interesting vertices). If neither rule can be applied, the resulting tree has size $O(|X|)$.

A high-level description of our computation of certificates for 3- and 4-edge-connectivity follows. We proceed one level at the time, and at each level we compress the cactus tree. Namely, at level k, $1 \le k \le 3$, we have a tree or something like a tree describing the k-edge-cuts and $(k+1)$-edge-connected classes. To reduce the number of k-edge-cuts and $(k+1)$-edge-connected classes, we compress this tree by using rules which are the analog of rules (1) and (2) above. That is, we will cut uninteresting branches (parts of the graph not containing interesting vertices and separated by a k-edge-cut) and shortcut uninteresting paths (parts not containing interesting vertices and separated by two k-edge-cuts). There are only $O(1)$ ways a branch or path may be used to connect the rest of the graph, so we will replace each branch or path by the smallest possible graph having the same edge connectivity properties, and that graph will have size $O(1)$. Then we go on to the next level.

Our description of these compressions will be given in a top-down fashion. We will first abstract three different compression problems that we need to solve. Next, we will show how to solve these problems. Finally, we will combine the solutions to these problems to achieve our certificates. The three different problems we solve are the following.

PROBLEM 5.1. *Given a planar connected graph G_0 and a set $X_1 \subseteq V(G_0)$ of vertices in G_0, find a graph G_1 that satisfies the following properties:*

(a) *$X_1 \subseteq V(G_1)$;*
(b) *G_1 has $O(|X_1|)$ 2-edge-connected components;*
(c) *For every $k \ge 2$, G_1 is a full certificate of k-edge-connectivity for X_1 in G_0;*
(d) *G_1 preserves planarity and is connected.*

PROBLEM 5.2. *Given a planar 2-edge-connected graph G_0 and a set $X_2 \subseteq V(G_0)$ of vertices in G_0, find a graph G_2 that satisfies the following properties:*

(a) *$X_2 \subseteq V(G_2)$;*
(b) *G_2 has $O(|X_2|)$ 3-edge-connected components;*
(c) *For every $k \ge 2$, G_2 is a full certificate of k-edge-connectivity for X_2 in G_0;*
(d) *G_2 preserves planarity and is 2-edge-connected.*

PROBLEM 5.3. *Given a planar 3-edge-connected graph G_0, a set $X_3 \subseteq V(G_0)$ of vertices, and a set $Y_3 \subseteq E(G_0)$ of edges in G_0, denote by $Z_3 \subseteq V(G_0)$ the set of*

endpoints of edges in Y_3. *Find a graph* G_3 *that satisfies the following properties:*

(a) $X_3 \subseteq V(G_3)$ *and* $Y_3 \subseteq E(G_3)$*;*

(b) G_3 *is a compressed full certificate of* 3- *and* 4*-edge-connectivity for* $(X_3 \cup Z_3)$ *in* G_0*;*

(c) G_3 *preserves planarity and is* 3*-edge-connected.*

Note that Problems 5.1, 5.2, and 5.3 admit the trivial solutions $G_1 = G_0$, $G_2 = G_0$, $G_3 = G_0$ whenever, respectively, $X_1 = V(G_0)$, $X_2 = V(G_0)$, $X_3 \cup Z_3 = V(G_0)$. Hence, we look for nontrivial solutions when $X_1 \subset V(G_0)$, $X_2 \subset V(G_0)$, and $X_3 \cup Z_3 \subset V(G_0)$.

Let G be a graph and let X be a set of interesting vertices. We now give a very high-level description of our algorithm that computes a compressed full certificate for 3- and 4-edge-connectivity of X in G. We will first solve Problem 5.1 with $G = G_0$ and $X = X_1$. This will give us a graph G_1 that has only $O(|X|)$ 2-edge-connected components and bridges but is still a planarity-preserving full certificate for X in G. Next, we will solve Problem 5.2 for each 2-edge-connected component of G_1, so as to reduce the overall number of 3-edge-connected components to $O(|X|)$. Finally, we will obtain our planarity-preserving compressed full certificates by solving Problem 5.3 for each 3-edge-connected component left in the graph at this point.

In the next sections, we will fill in the low-level details of our approach. In section 5.1 we show how to solve Problem 5.1, in section 5.2 we deal with Problem 5.2, and in section 5.3 with Problem 5.3. The solutions to these three problems will then be combined in section 5.4, yielding our compressed full certificates for 3- and 4-edge-connectivity.

5.1. Compressing a connected graph. In this section we present our solution to Problem 5.1. Let G_0 be a planar connected graph, and let $X_1 \subseteq V(G_0)$. We start by computing the 2-edge-connected components of G_0 [34]. As said before, the 2-edge-connected components of a graph have a tree-like structure: indeed, shrinking each 2-edge-connected class of G_0 into a super-vertex yields a tree whose edges are all and only the bridges of G_0. This is called the *bridge-block tree* of G_0. To compute the graph G_1 we work on the bridge-block tree of G_0: namely, we will apply to the bridge-block tree rules (1) and (2) of section 5. This will reduce the total number of vertices and edges in the bridge-block tree to $O(|X_1|)$. In what follows, we will often interchange the term 2-edge-connected component of G_0 with the corresponding 2-edge-connected class and with the corresponding node of its bridge-block tree, and the term bridge of G_0 with the corresponding edge of its bridge-block tree.

We now give the details of the compression and prove that it yields a solution to Problem 5.1. Let (S, T) be a minimum edge-cut separating X_1 in G_0: without loss of generality, assume that $X_1 \subseteq S$, and pick arbitrarily a vertex $t_1 \in T$. Color red the vertices of $X_1 \cup \{t_1\}$. Note that the total number of red vertices of G_0 is $|X_1| + 1$. Define a 2-edge-connected component to be *red* if it contains at least one red vertex, and define it to be *black* otherwise. Clearly, there are $O(|X_1|)$ red 2-edge-connected components. Define the *degree of a* 2*-edge-connected component* to be the number of bridges incident to it (i.e., the degree of its corresponding node in the bridge-block tree). Black 2-edge-connected components of degree one are uninteresting leaves in the bridge-block tree, and adjacent black 2-edge-connected components of degree two yield uninteresting chains in the bridge-block tree. We compress the black leaves and black chains of the bridge-block tree by applying the following two rules (analogs of rules (1) and (2)).

(B1) Let B be a black 2-edge-connected component of degree one, and let $e = (u, v)$

be the bridge incident to B. Contract e. This corresponds to deleting a black leaf from the bridge-block tree.

(B2) Let B_1 and B_2 be two adjacent black 2-edge-connected components of degree two. Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ be the two bridges incident to B_1, and let e_2 and $e_3 = (u_3, v_3)$ be the two bridges adjacent to B_2. Contract e_2 identifying u_2 and v_2. This corresponds to merging two adjacent black nodes of degree two in the bridge-block tree.

Note that both rules (B1) and (B2) delete one bridge and keep the graph connected. After rule (B2) is applied, the subgraph $B_1 \cup B_2$ from the contraction of bridge e_2 is 2-edge-connected. Similarly, let B' be the 2-edge-connected component adjacent to B in rule (B1): after contracting e, $B \cup B'$ is 2-edge-connected.

Let G_1 be the graph obtained from G_0 after all the rules (B1) and (B2) have been applied. Let H be any graph with $V(H) \cap V(G_0) \subseteq X_1$: since $G_1 \cup H$ is obtained from $G_0 \cup H$ by means of edge contractions, $G_1 \cup H$ is planar whenever $G_0 \cup H$ is planar. Thus, G_1 preserves planarity according to Definition 3.5. Furthermore, red vertices of G are never contracted by (B1) or (B2), and therefore $X_1 \cup \{t_1\} \subseteq V(G_1)$. Consider the bridge-block tree of G_1: because of (B1) there are no black leaves, and because of (B2) there are no two adjacent degree-two black nodes. This implies that there are at most $O(|X_1|)$ nodes in the bridge-block tree of G_1, and therefore $O(|X_1|)$ 2-edge-connected components and $O(|X_1|)$ bridges in G_1. This shows that G_1 satisfies properties (a), (b), and (d) of Problem 5.1. The following two lemmas show also that property (c) is satisfied, and therefore G_1 is a solution to Problem 5.1.

LEMMA 5.1. *Let $\widehat{G}$ be obtained from G_0 by applying either rule* (B1) *or rule* (B2). *Let H be given with $V(G_0) \cap V(H) \subseteq X_1$, and let x and y be any two vertices of $X_1 \cup \{t_1\} \cup V(H)$. For every $k \geq 2$, x and y are k-edge-connected in $\widehat{G} \cup H$ if and only if they are k-edge-connected in $G_0 \cup H$.*

Proof. We observe that neither (B1) nor (B2) can contract edges incident to vertices of $X_1 \cup \{t_1\}$. Since $V(G_0) \cap (V(H) \cup X_1 \cup \{t_1\}) \subseteq X_1 \cup \{t_1\}$, this implies that $\widehat{G} \cup H$ is obtained from $G_0 \cup H$ by contracting an edge that is not incident to either x or y. So if x and y are k-edge-connected in $G_0 \cup H$, they are k-edge-connected in $\widehat{G} \cup H$. Assume now that x and y are k-edge-connected in $\widehat{G} \cup H$. Then there are k edge-disjoint paths between x and y in $\widehat{G} \cup H$.

If (B1) was applied, $e = (u, v)$ is a bridge and B is a black 2-edge-connected component of degree one. Since B is black, it contains no vertex of $X_1 \cup \{t_1\}$. Furthermore, since $V(G_0) \cap V(H) \subseteq X_1$, no edge of H is incident to a vertex in B. This implies that neither x nor y is in B. After applying rule (B1), e is contracted, identifying vertices u and v into a new vertex w. Conversely, $G_0 \cup H$ can be obtained from $\widehat{G} \cup H$ by the inverse operation of splitting vertex w so as to reconstruct $e = (u, v)$. Since e is a bridge of G_0 separating B from the rest of the graph, and no edge of H is incident to B, e is a bridge of $G_0 \cup H$ too. This implies that after contracting edge e, w will be an articulation point of $\widehat{G} \cup H$, again separating B from the rest of the graph. Since neither x nor y is in B, there are k edge-disjoint simple paths in $\widehat{G} \cup H$ that do not contain edges of B. Thus, after splitting vertex w, the same k paths are edge-disjoint in $G_0 \cup H$, and therefore x and y are k-edge-connected in $G_0 \cup H$ too.

If (B2) was applied, B_1 and B_2 are black 2-edge-connected components of degree two in G_0. Since they are black, neither B_1 nor B_2 contains vertices of $X_1 \cup \{t_1\}$. Furthermore, since $V(G_0) \cap V(H) \subseteq X_1$, no edge of H is incident to a vertex in either B_1 or B_2. This implies that x and y are outside both B_1 and B_2. After applying rule (B2), e_2 is contracted and u_2 and v_2 are identified into a new vertex, say w_2.

Conversely, $G_0 \cup H$ can be obtained from $\widehat{G} \cup H$ by the inverse operation of splitting vertex w_2 into vertices u_2 and v_2 joined by edge e_2. Since no edge of H is incident to either B_1 or B_2, $\{e_1, e_3\}$ is a 2-edge-cut in $\widehat{G} \cup H$. Since neither x nor y is in $B_1 \cup B_2$, at most one of the k edge-disjoint paths in $\widehat{G} \cup H$ contains e_1 and e_3 and goes through B_1 and B_2. If w is split into vertices u_2 and v_2 joined by edge e_2, there is a new path containing edges e_1, e_2, and e_3 and going through B_1 and B_2 that is still edge-disjoint from the other $k-1$ paths: there are still k edge-disjoint paths between x and y in $G_0 \cup H$, and therefore x and y are k-edge-connected. □

The following corollary follows from Lemma 5.1.

COROLLARY 5.1. *Let $\widehat{G}$ be obtained from G_0 by applying either rule* (B1) *or rule* (B2). *For every $k \geq 2$, $\widehat{G}$ is a local certificate of k-edge-connectivity for X_1 in G_0.*

LEMMA 5.2. *Let $\widehat{G}$ be obtained from G_0 by applying either rule* (B1) *or rule* (B2). *Then for every $k \geq 2$, $\widehat{G}$ is a global certificate of k-edge-connectivity for X_1 in G_0.*

Proof. Let H be given with $V(G_0) \cap V(H) \subseteq X_1$. Rules (B1) and (B2) can only increase the edge connectivity of $G_0 \cup H$, and thus $\lambda(\widehat{G} \cup H) \geq \lambda(G_0 \cup H)$. To prove the lemma, it remains to show that if there is a k-edge-cut in $G_0 \cup H$, then in $\widehat{G} \cup H$ there must be an edge-cut of cardinality $k' \leq k$.

Let (S, T) be a k-edge-cut in $G_0 \cup H$. If both S and T contain vertices of $V(H) \cup X_1 \cup \{t_1\}$, a k-edge-cut in $\widehat{G} \cup H$ exists by Lemma 5.1. Otherwise, assume without loss of generality that $V(H) \cup X_1 \cup \{t_1\} \subseteq S$. This implies that T contains only black vertices of G_0 and that all the edges of (S, T) are in G_0: thus, (S, T) is an edge-cut separating X_1 in G_0. Let (S', T') be a minimum edge-cut separating X_1 from t_1 in G_0, with $X_1 \subseteq S'$ and $t_1 \in T'$, and let k' denote the cardinality of (S', T'). Recall that by the definition of t_1, (S', T') has the smallest cardinality among all of the edge-cuts separating X_1 in G_0: thus $k' \leq k$. Since $V(G_0) \cap V(H) \subseteq X_1 \subseteq S'$, $(S' \cup V(H), T')$ is itself a k'-edge-cut in $G_0 \cup H$. Since t_1 was colored red, a k'-edge-cut, $k' \leq k$, must exist in $\widehat{G} \cup H$ because of Lemma 5.1. □

It is clear from the proof of Lemma 5.2 why we needed to choose vertex t_1. Indeed, if G_0 has a bridge that leaves all the vertices of X_1 on one side, then $\lambda(G_0 \cup H) = 1$ for every H such that $V(H) \cap V(G_0) \subseteq X_1$. Picking vertex t_1 and coloring it red guarantees that there will be a bridge that leaves all the vertices of X_1 on one side in $\widehat{G} \cup H$ too.

Corollary 5.1 and Lemma 5.2 imply that G_1 is a full certificate for G. However, we remark that G_1 is not necessarily a *compressed* certificate, since it can have more than $O(|X_1|)$ vertices and edges: indeed, each 2-edge-connected component of G_1 might contain many vertices and edges. In the next section we show how to compress a 2-edge-connected graph, solving Problem 5.2.

5.2. Compressing a 2-edge-connected graph. We now turn to Problem 5.2. Let G_0 be a 2-edge-connected graph, and let $X_2 \subseteq V(G_0)$ be a set of vertices of G_0. As said before, if each 3-edge-connected class of G_0 is shrunk into one super-vertex, the resulting graph consists of a collection of simple cycles such that no two cycles share more than one super-vertex. This graph is the *cactus tree* of G_0: each node in the cactus tree corresponds to a 3-edge-connected class (and thus component) of G. Let p be the number of 3-edge-connected classes of G_0: the cactus tree has the property of having only $O(p)$ edges, and the edges in the cactus tree define all the possible 2-edge-cuts of G_0. Indeed, any two edges in the same cycle of the cactus tree define a 2-edge-cut of G_0. Consequently, even though the number of 2-edge-cuts can be $\Omega(p^2)$ (for instance, if G_0 is a simple cycle of p edges), the cactus tree of G_0 always

has size $O(p)$.

Let (S,T) be a minimum edge-cut separating X_2 in G_0, and without loss of generality, assume that $X_2 \subseteq S$. Pick arbitrarily one vertex $t_2 \in T$ and color red all the vertices of $X_2 \cup \{t_2\}$: note that the total number of red vertices of G_0 is $|X_2|+1$. Define a 3-edge-connected class of G_0 to be *red* if it contains at least one red vertex, and define it to be *black* otherwise. Clearly, there are $O(|X_2|)$ red 3-edge-connected classes. Our compression follows many of the same ideas used in section 5.1, but this time applies rules that are the analogs of rules (1) and (2) to the cactus tree of a 2-edge-connected graph rather than to the bridge-block tree of a connected graph. However, there are more technicalities involved, since the cactus tree here is not really a tree, but rather a tree of cycles.

Define the *degree of a 3-edge-connected class* to be the number of 2-edge-cuts incident to it (i.e., the number of cycles in the cactus tree that the corresponding node belongs to). If C is a 3-edge-connected class of G_0, there is a node in the cactus tree corresponding to C. Since there is no danger of ambiguity, we call this cactus tree node C also, and we color it with the same color we used for the 3-edge-connected class C.

Given a set S of vertices in G_0, define the graph $\Gamma_G(S)$ having as vertices S and all neighbors of S, and having as edges all edges in G_0 with one or both endpoints in S.

To compress "uninteresting leaves" of the cactus tree, we delete black nodes that appear in only one cycle. To compress "uninteresting chains," we take a chain of black nodes of the cactus tree that appear in the same 2-cycle and merge them into a smaller certificate for that chain. The two rules (analogs of rules (1) and (2)) that we use are the following.

(T1) Let C be a black 3-edge-connected class of degree one, and let $\{e_1, e_2\}$ be the 2-edge-cut incident to C. Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$, with u_1 and u_2 in C. Replace e_1, e_2, and C with an edge $e = (v_1, v_2)$.

(T2) Let $C_1, C_2, \ldots, C_\ell$ be a set of one or more black 3-edge-connected classes of degree two, such that each adjacent pair C_i, C_{i+1} shares an edge-cut $\{e_i, e_i'\}$. Let $\{e_0, e_0'\}$ and $\{e_\ell, e_\ell'\}$ be the two 2-edge-cuts separating $\bigcup C_i$ from the rest of the graph, and let $Y = \{v_0, v_0', v_\ell, v_\ell'\}$ be the endpoints of these cut edges in the rest of the graph. Let $B = \Gamma_G(C_1 \cup C_2 \cup \cdots \cup C_\ell)$; i.e., B consists of the edges induced by each C_i together with all cut edges e_i, e_i', $0 \le i \le \ell$. Then replace B by the minimum size planarity-preserving certificate B' of k-edge-connectivity (for all k) for B with the vertices of Y denoted as interesting.

The definition of rule (T2) may seem circular, as we are invoking the existence of certificates in the definition of an algorithm we are trying to use to prove that certificates exist. However, we know that there will always exist some planarity-preserving certificate: B itself is such a graph. Therefore, (T2) is well defined, although it may not be clear how to implement it efficiently. The reason for stating (T2) in this form is threefold: it is very general, and applies equally well to other forms of connectivity in other cactus trees; it motivates the seemingly more complicated rule (T2′) below, and it is immediately apparent that the result is a certificate. A general argument based on the possible connectivity requirements through vertices in Y, and on the different ways these vertices can be placed in an embedding of B, shows that only $O(1)$ different replacement certificates are needed in rule (T2), and hence B' has size $O(1)$. We do not elaborate, as this argument does not give good bounds on $|B'|$. Instead, we replace (T2) by rule (T2′) below. With rule (T2′), the size of B' is clearly

$O(1)$, but it is less clear that the resulting graph is a certificate. We prove this by showing that (T2) and (T2′) are equivalent.

Our description is based on *flows* in G. If we assign orientations and nonnegative *flow amounts* to the edges of G, the *excess* at any vertex is the sum of the flow amounts on incoming edges minus the sum of flow amounts on outgoing edges. A *flow* is an assignment of orientations and flow amounts such that, except at certain designated *terminals*, the excess at each vertex is zero. We will assume *integer flows* and *unit capacity edges*; i.e., all flow amounts should be zero or one. A *flow requirement* is the designation of values on certain terminal vertices; we say that a flow requirement is *satisfied* if there exists a flow such that, at each terminal, the excess equals the designated value. The *total flow* is the sum of positive flow requirement values; we call a flow with total flow k a *k-unit flow*. Any k-unit flow can be partitioned into k single-unit flows; a single-unit flow is just a path connecting its two terminals, possibly together with some cycles. This partitioning of a flow into paths forms one-half of the *max-flow min-cut theorem* [3], that x and y are k-connected if and only if there exists a flow (with integer edge capacities, and which can be restricted to an integer flow without loss of generality) with x and y as terminals having designated values $-k$ and k; for a proof of this connection between flows and connectivity, see any graph theory text.

First, we define the following term. Suppose that $\ell = 1$, so that B consists of the subgraph induced by a single black 3-edge-connected class C_1 together with the four vertices Y and the edges connecting Y to C_1. We say that B is *well connected* if, for any way of partitioning Y into two pairs of vertices, there is a two-unit flow in B with the first pair as the two sources and the second pair as the two sinks. For instance, if B consists of a star $K_{1,4}$, it is well connected. We are now ready to define rule (T2′).

(T2′) Let $C_1, C_2, \ldots, C_\ell$ be a set of one or more black 3-edge-connected classes of degree two, such that each adjacent pair C_i, C_{i+1} shares an edge-cut $\{e_i, e_i'\}$. Let $\{e_0, e_0'\}$ and $\{e_\ell, e_\ell'\}$ be the two 2-edge-cuts separating $\bigcup C_i$ from the rest of the graph, and let $Y = \{v_0, v_0', v_\ell, v_\ell'\}$ be the endpoints of these cut edges in the rest of the graph. Let $B = \Gamma_G(C_1 \cup C_2 \cup \cdots \cup C_\ell)$.

- (i) If $\ell > 1$, form B' from B by contracting $C_1 \cup C_2 \cup \cdots \cup C_\ell$ to a single vertex.
- (ii) If $\ell = 1$, and C_1 is well connected, form B' from B by contracting C_1 to a single vertex.
- (iii) If $\ell = 1$, and C_1 is not well connected, let T be a spanning tree of B. (We show below that B is connected, so T exists.) Contract any edge in T adjacent to a vertex in C_1 of degree one or two, until no more contractions are possible; let B' be the resulting contracted tree.

In all cases replace B by B'.

LEMMA 5.3. *The graph B specified in rules* (T2) *and* (T2′) *is connected; moreover, the subgraph induced by each 3-edge-connected class C_i is connected.*

Proof. The connectedness of the subgraph induced by C_i follows because each pair of vertices in C_i is connected by three edge-disjoint paths, and only two such paths can pass through the four edges separating C_i from the rest of G. For each pair of classes C_i, C_{i+1} there is an edge connecting that pair, from which the overall connectedness of B follows. □

LEMMA 5.4. *Rule* (T2′) *preserves planarity.*

Proof. In all three cases of rule (T2′), the resulting graph B' is a minor of B and hence preserves planarity. □

We next show that the graph B' resulting from rule (T2′) is a certificate of k-edge-connectivity for B. A pair of vertices is k-edge-connected if and only if there is a k-unit flow with one vertex as source and the other as sink. If we consider the orientations and flow amounts of this flow, restricted to the edges of B, we can think of the excesses at each vertex of Y as giving flow requirements satisfied by this flow. (The excesses at vertices of $B - Y$ must be zero since all incident edges of such vertices are in B.) Replacing this flow on B by any other flow satisfying the same requirements will produce a k-unit flow in B, so it is enough to show that, whenever a flow requirement F with terminals in Y can be satisfied in B, it can also be satisfied in B'.

Note that we need only consider flow requirements of at most one unit at each vertex of Y, since each vertex of Y is adjacent to a single edge in B. Thus larger flow requirements could not be satisfied in B. So, overall, there are at most two single-unit sources and two single-unit sinks. If there is one source or sink only, the satisfiability of the flow requirements follows from the connectedness of B (Lemma 5.3) and from the connectedness of B' (obvious from the definition of rule (T2′)).

Let the remaining two-unit flow requirements be denoted F_1, having sources $\{v_0, v_0'\}$ and sinks $\{v_\ell, v_\ell'\}$; F_2, having sources $\{v_0, v_\ell\}$ and sinks $\{v_0', v_\ell'\}$; and F_3, having sources $\{v_0, v_\ell'\}$ and sinks $\{v_0', v_\ell\}$. (Any other collection of two sources and two sinks can be found by switching sources and sinks in some F_i.)

LEMMA 5.5. *Flow requirement F_1 is satisfiable in B.*

Proof. By assumption, G is 2-edge-connected, so there exists a pair of edge-disjoint paths in G from v_0 to v_ℓ. Each path must cross B, and the union of these two paths intersected with B forms a two-unit flow satisfying F_1. ∎

LEMMA 5.6. *At least one of F_2 or F_3 is satisfiable in B.*

Proof. Let F_1 be satisfied by a 2-unit flow. Then the flow can be partitioned into two edge-disjoint paths, each connecting a source with a sink. If the paths connect v_0 with v_ℓ and v_0' with v_ℓ', then F_3 is also satisfiable. If the paths connect v_0 with v_ℓ' and v_0' with v_ℓ, then F_2 is also satisfiable. ∎

LEMMA 5.7. *If some flow requirement F_i is not satisfiable by a flow in B, then B consists of the subgraph induced by a single class C_1.*

Proof. By Lemma 5.6, we can without loss of generality let the unsatisfiable flow requirement be F_2. If $\ell > 1$, this flow requirement would be satisfied by finding one path in the subgraph induced by C_1 connecting v_0 and v_0', and a path in the subgraph induced by C_ℓ connecting v_ℓ and v_ℓ', both of which exist by Lemma 5.3. ∎

LEMMA 5.8. *When part* (iii) *of rule* (T2′) *is applied to a graph B in which the flow requirement F_2 is not satisfiable by any flow, the resulting graph B' consists of Y together with two vertices u and u', with edges v_0u, uv_1, uu', $v_0'u'$, and $u'v_1'$.*

Proof. Since B' is a contraction of a tree, it is itself a tree. Since F_2 is unsatisfiable, B contains a bridge edge e between $\{v_0, v_1\}$ and $\{v_0', v_1'\}$ by the max-flow min-cut theorem [3]. This edge e must be in T. Construct a tree T' by contracting out all degree-one black vertices in T. Then B' remains a contraction of T'. Since e is on a path between two vertices of Y, it is not contracted in forming T'. On either side of e, T' consists of paths connecting v_0 with v_1, and v_0' with v_1'. Further, note that all vertices in Y must be leaves in T' since they have degree one in B and their degree does not increase in forming T'. The only compressed tree possible with four leaves and a bridge separating the two pairs of leaves is the one described in the lemma. ∎

LEMMA 5.9. *The graph B' resulting from rule* (T2′) *is a certificate of k-edge-*

connectivity for B.

Proof. We show that for every flow requirement F on terminals in Y, F is satisfiable by a flow in B if and only if it is satisfiable in B'. Requirements with more than two units of flow are never satisfiable in either graph. One-unit requirements are satisfiable in B by Lemma 5.3, and in B' since B' is clearly connected. Thus we need only worry about requirements F_1, F_2, and F_3.

If all three of the requirements F_i are satisfiable by flows in B, then either $\ell > 1$ or B consists of a single well-connected component C_1. In either case parts (i) or (ii) of rule (T2′) contract B to a star $K_{1,4}$ in which all three collections remain satisfiable.

If only two of the three requirements are satisfiable, then by Lemma 5.7, B consists of the subgraph induced by the single component C_1 which must not be well connected. By Lemma 5.6, we can assume without loss of generality that requirement F_2 is not satisfiable by a flow in B. Then by Lemma 5.8, B' consists of a tree with four leaves and a bridge, containing edge-disjoint paths from v_0 to v_1 and v_0' to v_1'; in this graph as in B, F_1 and F_3 are satisfiable and F_2 is not satisfiable. ∎

LEMMA 5.10. *Rule* (T2) *is equivalent to rule* (T2′).

Proof. The fact that B' is a certificate is shown in Lemma 5.9. It can be shown that it is the minimum certificate for B by noting that any certificate must contain a spanning tree with at least as many vertices as are in B'; we omit the details, as they are not necessary for the overall correctness of our connectivity algorithm. ∎

Next, we analyze the size of the cactus tree for G' and show that the graph obtained at the end of this compression is a solution to Problem 5.2. Recall that the meaning of our rules in the cactus tree is the following. Rule (T1) takes as input a black node that is contained in only one cycle and deletes it, while rule (T2) compresses a chain of black nodes in the cactus tree. The following lemmas show that the graph obtained at the end of this compression is a solution to Problem 5.2.

LEMMA 5.11. *Let G_2 be the graph obtained from a 2-edge-connected graph G_0 by repeated applications of rules* (T1) *and* (T2)*, until no further rule can be applied. Then there are $O(|X_2|)$ 3-edge-connected components in G_2, and its cactus tree contains $O(|X_2|)$ nodes and edges.*

Proof. Since rules (T1) and (T2) can be described in terms of the cactus tree, and each node of the cactus tree corresponds to a 3-edge-connected class (and thus component), we refer to nodes of the cactus tree as components, and show that when no rule can be applied any more, we are left with a cactus tree that contains $O(|X_2|)$ nodes and edges.

We observe that the cactus tree can be represented by a tree. We form a black or red node for every component in the cactus tree, and a blue node for every cycle in the cactus tree. A node representing a component is adjacent to a node representing a cycle if and only if the cycle contains that component.

Because of rule (T1), a black component must be shared by more than one cycle in the cactus tree. Therefore, all leaves of the tree are red, and there are at most $|X_2| + 1$ leaves.

Next, we examine the degree-two nodes in the tree. Form maximal chains of black and blue degree-two nodes. Each such chain is terminated either by a high-degree node or by a red node; therefore, as in any tree, there can be at most $2|X_2| - 1$ chains. Each chain can consist of at most one black node and two blue nodes by rule (T2), so there are at most $2|X_2| - 1$ black nodes of degree two and at most $4|X_2| - 2$ blue nodes of degree two.

Finally, as in any tree, there are at most two fewer nodes of degrees higher than

two than there are leaves, so there are at most $|X_2| - 1$ such nodes.

In total we find at most $3|X_2| - 2$ black nodes representing black components of the cactus tree, and at most $5|X_2| - 3$ blue nodes representing cycles in the cactus tree. Any cactus tree with k components has at most $2k - 2$ edges, so there are at most $8|X_2| - 4$ edges in the cactus tree. ∎

LEMMA 5.12. *Let $\widehat{G}$ be obtained from G_0 by repeatedly applying rules* (T1) *and* (T2). *Let H be given with $V(G_0) \cap V(H) \subseteq X_2$. Any two vertices of $X_2 \cup V(H) \cup \{t_2\}$ are k-edge-connected in $\widehat{G} \cup H$ if and only if they are k-edge-connected in $G_0 \cup H$, $k \geq 2$.*

Proof. By transitivity of certificates, we need only show that each step produces a certificate for the previous graph. Steps involving rule (T1) yield a split graph with respect to the 2-edge-cut $\{e_1, e_2\}$, and Lemma 4.5 shows that this produces a certificate of k-edge-connectivity. Steps involving rule (T2) yield a certificate by definition. ∎

Lemma 5.12 implies the following corollary.

COROLLARY 5.2. *Let $\widehat{G}$ be obtained from G_0 by repeatedly applying rules* (T1) *and* (T2). *For every $k \geq 2$, $\widehat{G}$ is a local certificate of k-edge-connectivity for X_2 in G_0.*

LEMMA 5.13. *Let $\widehat{G}$ be obtained from G_0 by applying rule* (T1) *or* (T2). *Then $\widehat{G}$ is a global certificate of k-edge-connectivity for X_2 in G_0, $k \geq 2$.*

Proof. Let H be given with $V(G_0) \cap V(H) \subseteq X_2$. Rules (T1) and (T2) can only increase the connectivity of $G_0 \cup H$, and thus $\lambda(\widehat{G} \cup H) \geq \lambda(G_0 \cup H)$. Hence, it remains to show that any k-edge-cut (S, T) in $G_0 \cup H$ corresponds to an edge-cut in $\widehat{G} \cup H$ of cardinality at most k.

If both S and T contain vertices in $V(H) \cup X_2 \cup \{t_2\}$, a corresponding edge-cut can be found by Lemma 5.12. Otherwise assume without loss of generality that T contains only black vertices of G_0. Then all edges of cut (S, T) are in G_0. Let (S', T') be the minimum edge-cut separating X_2 from t_2 in G_0. Then $(S' \cup V(H), T')$ has edge cardinality at most that of (S, T), and since t_2 was colored red, a corresponding edge-cut exists in $\widehat{G} \cup H$ by Lemma 5.12. ∎

5.3. Compressing a 3-edge-connected graph. In this section we describe our solution to Problem 5.3. Since it relies heavily on the notion of cactus tree, we first list some properties of a cactus tree of a 3-edge-connected graph, referring the interested reader to [4, 28] for the full details. It might be helpful to refer to Figure 2 while we discuss these properties.

Let G_0 be a 3-edge-connected graph, and let $\mathcal{T}(G_0)$ denote its cactus tree. Since 3 is odd, $\mathcal{T}(G_0)$ is an actual tree (and not a tree of edges and cycles). Edges of $\mathcal{T}(G_0)$ correspond to 3-edge-cuts of G_0. Each 4-edge-connected class of G_0 corresponds to a node in $\mathcal{T}(G_0)$, while the converse is not necessarily true. Indeed, there can be nodes that do not correspond to any 4-edge-connected class: the reason for having these nodes is to represent the tree structure of the 3-edge-cuts (see Figure 2). We call these nodes *empty* since they do not contain any vertex of G_0. In what follows, since there is no danger of ambiguity, we will use interchangeably the terms 3-edge-cut of G_0 and edge of $\mathcal{T}(G_0)$, and the terms 4-edge-connected class of G_0 and nonempty node of $\mathcal{T}(G_0)$. Note that an edge $e = (u, v)$ of G_0 might take part in different 3-edge-cuts of G_0. However, all these 3-edge-cuts must form a path in $\mathcal{T}(G_0)$: endpoints of this path are the two nodes corresponding to the 4-edge-connected classes containing u and v. We label each cactus edge with the three edges of the corresponding 3-edge-cut.

We now describe our solution to Problem 5.3. Let G_0 be a 3-edge-connected graph, and let $X_3 \subseteq V(G_0)$ and $Y_3 \subseteq E(G_0)$. Let Z_3 be the set of endpoints of edges in Y_3. We color red the vertices of $X_3 \cup Z_3$ and the edges of Y_3. Note that both the endpoints of a red edge are colored red. We define a 3-edge-cut of G_0 to be an *inter* 3-*edge-cut* if it separates two different red vertices of G_0, and define it to be an *extra* 3-*edge-cut* otherwise. If G_0 has at least one *extra* 3-edge-cut, we pick one such 3-edge-cut and we choose arbitrarily one vertex that is separated from all the red vertices by this 3-edge-cut. We call this vertex the *chosen extra vertex*, and we call the 3-edge-cut the *chosen extra* 3-*edge-cut*. We color the chosen extra vertex red: the total number of red vertices of G_0 is therefore $(|X_3 \cup Z_3| + 1)$. Our computation of a solution for Problem 5.3 consists of the following three phases.

`Phase 1:` Compute the cactus tree $\mathcal{T}(G_0)$ of G_0 (see Figure 4(b)).

`Phase 2:` Compute a new cactus tree $\mathcal{T}'$ by compressing $\mathcal{T}(G_0)$, so that $\mathcal{T}'$ will have size linear in the number of red vertices (see Figure 4(c)).

`Phase 3:` Compute G_3 from $\mathcal{T}'$.

We now give the low-level details of these phases. Phase 1 is simply accomplished by computing the cactus tree of G_0. Let R be the set of red vertices in G_0, and let R_1 and R_2 be any two nonempty sets of red vertices in G_0, $R_1 \cap R_2 = \emptyset$. We say that an edge of $\mathcal{T}(G_0)$ separates R_1 and R_2 if it corresponds to a 3-edge-cut separating R_1 and R_2 in G_0. The following lemma is an immediate consequence of the notion of cactus tree [4].

LEMMA 5.14. *Let R_1 and R_2 be any two nonempty sets of red vertices in G_0, $R_1 \cap R_2 = \emptyset$. The* 3*-edge-cut $\{e_1, e_2, e_3\}$ of G_0 separates R_1 and R_2 if and only if the edge (α, β) labeled with $\{e_1, e_2, e_3\}$ in $\mathcal{T}(G_0)$ separates R_1 and R_2.*

In Phase 2 we use a compression similar to the one used for the MSF (minimum spanning forest) certificate. Given the graph G_0 and its cactus tree $\mathcal{T}(G_0)$, we define the *degree of a* 4*-edge-connected class* of G_0 to be the tree degree of the corresponding (nonempty) node in $\mathcal{T}(G_0)$. Intuitively, the degree of a 4-edge-connected class denotes the number of different 3-edge-cuts that are "incident" to the class. Again, we define a 4-edge-connected class to be *red* if it contains at least one red vertex, and define it to be *black* otherwise. We color red the (nonempty) nodes of $\mathcal{T}(G_0)$ that correspond to red 4-edge-connected classes of G_0 (see Figure 4(b)). Black 4-edge-connected classes of degree one are uninteresting leaves in the cactus tree, and adjacent black nodes of degree two give rise to uninteresting chains in the cactus tree. Our rules to compress $\mathcal{T}(G_0)$ are the following.

(Q1) Let Q be a black cactus node of degree 1. Delete Q and its incident edge from the cactus tree.

(Q2) Let Q be a black cactus node of degree 2. Let e_1 and e_2 be the cactus edges incident to Q, and let Q_1 and Q_2 be the cactus nodes adjacent to Q. Delete Q and replace e_1 and e_2 with an edge e between Q_1 and Q_2. Edge e gets the same label as e_1.

We call $\mathcal{T}'$ the cactus tree obtained after all the rules (Q1) and (Q2) have been applied. Note that $\mathcal{T}'$ contains all the red nodes of $\mathcal{T}(G_0)$ and may contain some black nodes of $\mathcal{T}(G_0)$. However, because of (Q1) and (Q2), the total size of $\mathcal{T}'$ is linear in the number of red nodes. Note that edges of $\mathcal{T}'$ are also edges of $\mathcal{T}(G_0)$, and therefore correspond to 3-edge-cuts of G_0. Again, given two nonempty disjoint sets of red vertices R_1 and R_2, we say that an edge of $\mathcal{T}'$ separates R_1 and R_2 if it corresponds to a 3-edge-cut separating R_1 and R_2 in G_0. As the following lemma shows, $\mathcal{T}'$ has the property of preserving 3-edge-cuts separating any two sets of red

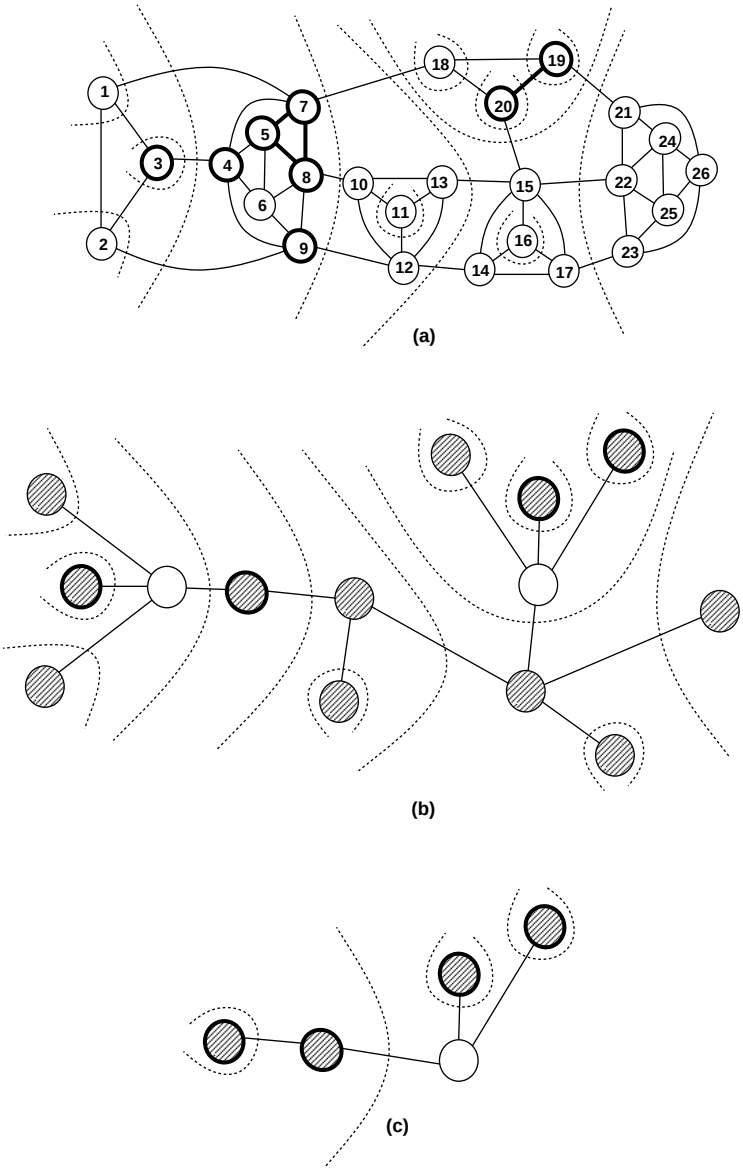

FIG. 4. *Compressing a 3-edge-connected graph G_0.* (a) *The original graph G_0: red vertices ($\{3,4,5,7,8,9,19,20\}$) and red edges ($\{(5,7),(5,8),(7,8),(19,20)\}$) are shown in bold, 3-edge-cuts are dashed.* (b) *The cactus tree $\mathcal{T}(G_0)$: red 4-edge-connected classes are shown in bold.* (c) *The compressed cactus tree $\mathcal{T}'$ obtained from $\mathcal{T}(G_0)$.*

vertices in G_0.

LEMMA 5.15. *Let R_1 and R_2 be any two nonempty sets of red vertices in G_0, $R_1 \cap R_2 \neq \emptyset$. There is an edge (α, β) in $\mathcal{T}(G_0)$ separating R_1 and R_2 if and only if there is an edge (α', β') in $\mathcal{T}'$ separating R_1 and R_2.*

Proof. To prove the lemma, we show that rules (Q1) and (Q2) maintain the following invariant. There is an edge (α, β) separating R_1 and R_2 before the rule is

applied if and only if there is an edge (α', β') separating R_1 and R_2 after the rule is applied.

If rule (Q1) is applied, Q is a black node. Since the edge incident to Q cannot separate red vertices, the invariant must hold. If rule (Q2) is applied, Q is a black node. Consequently, e_1 and e_2 separate exactly the same set of red vertices. But then the edge e inserted by rule (Q2) still separates the same set of red vertices previously separated by e_1 and e_2. Hence, the invariant still holds. □

Now, we describe our implementation of Phase 3, namely, how to compute from $\mathcal{T}'$ the graph G_3 that is a solution to Problem 5.3. By Lemmas 5.14 and 5.15, we know that, given any two sets R_1 and R_2 in G_0, they can be separated by a 3-edge-cut of G_0 if and only if they can be separated by an edge of $\mathcal{T}'$. In Phase 3, we build a graph G_3 that contains the red vertices and red edges of G_0, and has all the 3-edge-cuts corresponding to edges of $\mathcal{T}'$. We will use this property to prove that any two sets of red vertices R_1 and R_2 can be separated by a 3-edge-cut in G_0 if and only if they can be separated by a 3-edge-cut in G_3.

We show how to obtain G_3. For each edge (α', β') of $\mathcal{T}'$, we do the following. Let $\{e_1, e_2, e_3\}$ be the label of (α', β') (corresponding to the 3-edge-cut $\{e_1, e_2, e_3\}$ of G_0): we mark the edges e_1, e_2, and e_3 and their endpoints in G_0. In other words, all the edges that are in a 3-edge-cut corresponding to an edge of $\mathcal{T}'$ are *marked*. Next, we delete all the marked edges from G_0, and denote by G_0^1, G_0^2, ..., G_0^p, $p \geq 1$, the connected components left (see Figure 5(a)). The edges of G_0^i, $1 \leq i \leq p$, are referred to as *unmarked*.

Our graph G_3 will keep all the marked edges and replace each G_0^i with a smaller graph that preserves 3- and 4-edge-connectivity between the red vertices. We compute this smaller graph as follows. We denote the red and marked vertices of G_0^i *interesting*. We force 4-edge-connectedness between any two interesting vertices in G_0^i by replacing G_0^i with a smaller graph that contains all the interesting vertices and the red edges originally in G_0^i and that is 4-edge-connected. We call this graph $\mathcal{C}(G_0^i)$. Note that this might change the edge connectivity inside G_0^i, since it may change the number of edge-disjoint paths inside G_0^i. Namely, two vertices x and y of G_0^i may be ℓ-edge-connected, $\ell \geq 4$ in G_0, and ℓ'-edge-connected, $\ell' \geq 4$, $\ell' \neq \ell$, in the new graph obtained after replacing G_0^i with $\mathcal{C}(G_0^i)$. However, as we will show, this is not a problem, since we are interested in certificates for 3- and 4-edge-connectivity only, and not for higher edge connectivity. We make all the interesting vertices of G_0^i 4-edge-connected in the new graph by compressing the unmarked edges as follows. We first compute the minimal subgraph S_0^i of G_0^i that is connected and contains all the red edges and the interesting vertices in G_0^i (see Figure 5(b)). Note that S_0^i is not necessarily a tree, since it might contain cycles: however, because of the minimality of S_0^i, the possible cycles of S_0^i can have only red edges. We collapse each red cycle in a single node, and then compress the tree obtained, using as interesting vertices the collapsed cycles, the red vertices and the marked vertices. The compression of the unmarked edges left in S_0^i is the same as used in the MSF algorithm: we compress uninteresting branches and uninteresting chains of degree-two vertices (see Figure 5(c)). Next, we expand the red cycles back, and finally, we quadruplicate each unmarked edge left in the compressed graph, to force 4-edge-connectivity. We call the graph obtained $\mathcal{C}(G_0^i)$, and we call G_3 the graph obtained after replacing G_0^i with $\mathcal{C}(G_0^i)$, $1 \leq i \leq p$ (see Figure 5(d)).

Note that G_3 is obtained from G_0 by means of a suitable sequence of the following operations:

(1) delete unmarked edges inside a component G_0^i; and

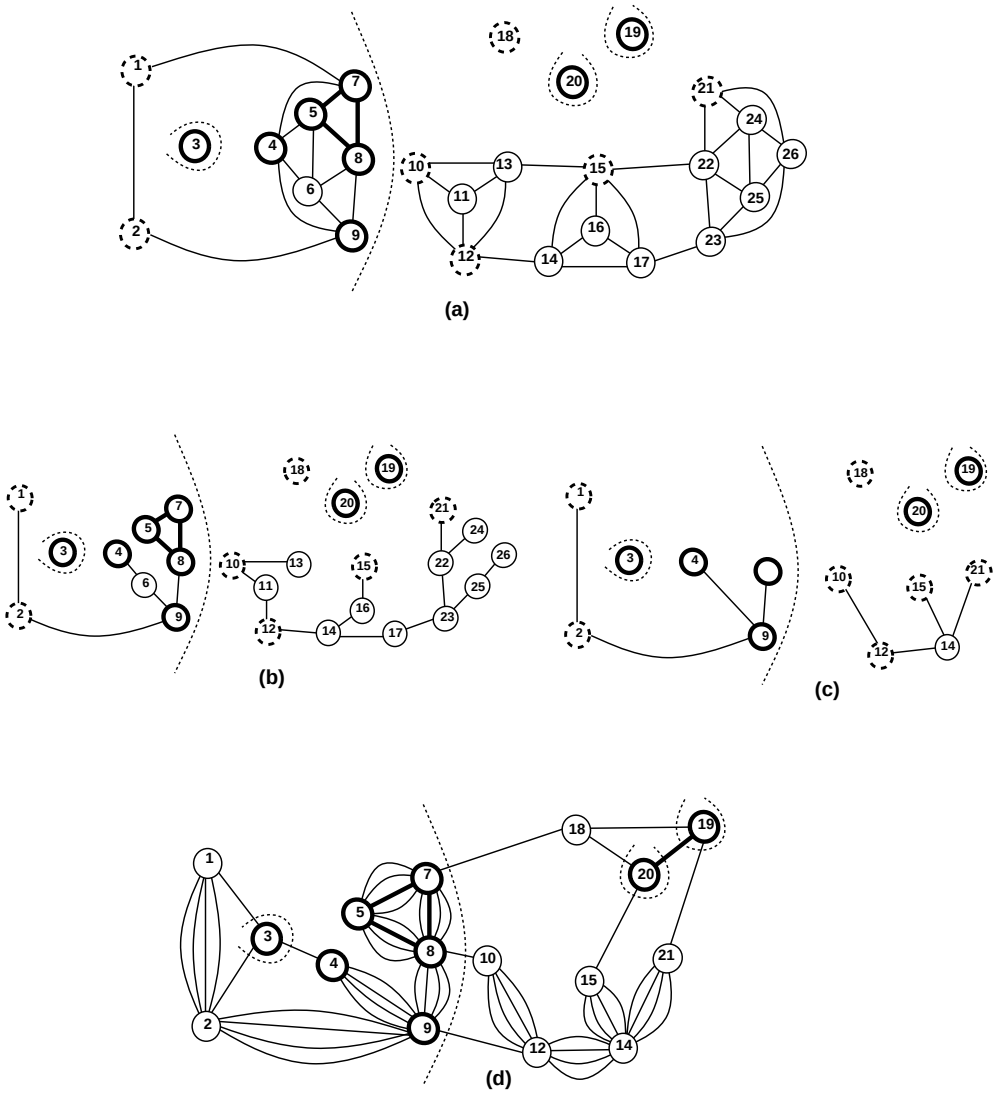

FIG. 5. *Compressing G_0; 3-edge-cuts corresponding to edges of $\mathcal{T}'$ are always shown dashed.* (a) *The connected components G_0^i, $1 \le i \le p$, obtained after the removal of 3-edge-cuts in $\mathcal{T}'$. Marked (non-red) vertices are shown dashed.* (b) *The minimal subgraphs S_0^i, $1 \le i \le p$.* (c) *The graphs obtained after compressing S_0^i, $1 \le i \le p$: the red cycle containing vertices* 5, 7, *and* 8 *has been collapsed into a single node.* (d) *The graph G_3 obtained from G_0 by replacing G_0^i with $\mathcal{C}(G_0^i)$, $1 \le i \le p$. The red cycle has been expanded.*

(2) contract unmarked edges; and
(3) quadruplicate unmarked edges.

Conversely, G_0 can be recomputed from G_3 by means of a suitable sequence of the following operations:

(4) delete multiple unmarked edges;
(5) expand unmarked edges;

(6) insert new unmarked edges inside a component $\mathcal{C}(G_0^i)$.

We now prove that G_3 is a solution to Problem 5.3.

LEMMA 5.16. *G_3 is 3-edge-connected, and $X_3 \subseteq V(G_3)$ and $Y_3 \subseteq E(G_3)$.*

Proof. We first prove that G_3 is 3-edge-connected. Assume by contradiction that the minimum edge-cut γ of G_3 is of cardinality ℓ, $\ell \leq 2$. Recall that G_3 consists of marked edges and unmarked edges. Since each unmarked edge is quadruplicated in G_3, its endpoints are 4-edge-connected. This implies that no unmarked edge can be in γ, and therefore, γ consists of marked edges only. Due to the minimality of γ, deleting the edges of γ disconnects G_3 into two graphs: say G_3' and G_3''. Since γ contains only marked edges, and each $\mathcal{C}(G_0^i)$ is connected, each $\mathcal{C}(G_0^i)$ is contained in either G_3' or G_3''. Recall that G_0 can be obtained from G_3 by means of operations of type (4), (5), and (6) above. Each such operation modifies only the graphs $\mathcal{C}(G_0^i)$ by deleting multiple edges, expanding unmarked edges, and inserting new unmarked edges. Marked edges will be unaffected by these operations. This implies that all these operations will be local to the graphs $\mathcal{C}(G_0^i)$, and therefore will still keep the vertices in G_3' and in G_3'' disconnected. As a result, γ is an ℓ-edge-cut in G_0. This is clearly a contradiction, since $\ell \leq 2$ and G_0 is 3-edge-connected. Thus, there cannot be an ℓ-edge-cut, $\ell \leq 2$, in G_3, and therefore G_3 must be 3-edge-connected.

We now prove that G_3 contains all the red vertices and edges of G_0. Since red vertices are kept in each $\mathcal{C}(G_0^i)$, we have that $X_3 \subseteq V(G_3)$. As for red edges, recall that the edges of G_0 are partitioned into marked and unmarked. Each $\mathcal{C}(G_0^i)$ preserves all the red edges in G_0^i: this shows that all the unmarked red edges are in G_3. Since G_3 preserves all the marked edges, it will preserve also the marked red edges. This shows that $Y_3 \subseteq E(G_3)$. ∎

Before proving the other properties that make G_3 a solution to Problem 5.3, we need some technical lemmas.

LEMMA 5.17. *Let R_1 and R_2 be any two nonempty sets of red vertices in G_0, $R_1 \cap R_2 = \emptyset$. There is a 3-edge-cut in G_0 separating R_1 and R_2 if and only if there is a 3-edge-cut in G_3 separating R_1 and R_2.*

Proof. Assume that there is a 3-edge-cut $\{e_1, e_2, e_3\}$ separating R_1 and R_2 in G_0. By Lemmas 5.14 and 5.15 there must be an edge (α', β') in $\mathcal{T}'$ separating R_1 and R_2. Let $\{e_1', e_2', e_3'\}$ be the label of (α', β'). We claim that $\{e_1', e_2', e_3'\}$ is a 3-edge-cut of G_3 separating R_1 and R_2. Note that, as a consequence of our computation of $\mathcal{T}'$, $\{e_1', e_2', e_3'\}$ must have been the label of one edge of $\mathcal{T}(G_0)$. By definition of cactus tree, this implies that $\{e_1', e_2', e_3'\}$ is a 3-edge-cut of G_0. Note that it is not necessarily the case that $\{e_1, e_2, e_3\} = \{e_1', e_2', e_3'\}$. Both $\{e_1, e_2, e_3\}$ and $\{e_1', e_2', e_3'\}$ are 3-edge-cuts separating R_1 and R_2 in G_0, and $\{e_1', e_2', e_3'\}$ is kept in $\mathcal{T}'$ (and therefore in G_3).

Denote by G_0' and G_0'' the graphs obtained from G_0 after removing e_1', e_2', and e_3', and without loss of generality, assume that $R_1 \subseteq V(G_0')$ and $R_2 \subseteq V(G_0'')$. Note that $\{e_1', e_2', e_3'\}$ is a 3-edge-cut corresponding to an edge of $\mathcal{T}'$. As a result, the edges e_1', e_2', and e_3' are marked edges, and each G_0^i is contained in either G_0' or G_0''. Recall that G_3 is obtained from G_0 by means of operations (1)–(3) above, which are local to the graphs G_0^i and therefore will still keep G_0' and G_0'' disconnected. Since $R_1 \subseteq V(G_0')$ and $R_2 \subseteq V(G_0'')$, this implies that $\{e_1', e_2', e_3'\}$ is a 3-edge-cut separating R_1 and R_2 in G_3.

Assume now that there is a 3-edge-cut $\{e_1, e_2, e_3\}$ separating R_1 and R_2 in G_3. Note that no unmarked edge can be in a 3-edge-cut, since each unmarked edge left after the compression of S_0^i is quadruplicated, therefore making its endpoints 4-edge-connected. As a result, e_1, e_2, and e_3 must all be marked edges. This implies that

e_1, e_2 and e_3 are all edges of G_0.

Since by Lemma 5.16 G_3 is 3-edge-connected, $\{e_1, e_2, e_3\}$ is a minimum edge-cut in G_3, and therefore the removal of e_1, e_2, and e_3 splits G_3 into two graphs, say G'_3 and G''_3. Without loss of generality assume that $R_1 \subseteq V(G'_3)$ and $R_2 \subseteq V(G''_3)$. Recall that G_0 can be obtained from G_3 by means of operations (4)–(6) above, which still keep G'_3 and G''_3 disconnected. Since $R_1 \subseteq V(G'_3)$ and $R_2 \subseteq V(G''_3)$, this implies that $\{e_1, e_2, e_3\}$ is a 3-edge-cut separating R_1 and R_2 in G_0. □

LEMMA 5.18. *Let γ be a 3-edge-cut of G_3, separating V' and V'', with $V', V'' \subseteq V(G_3)$. Then γ is a 3-edge-cut of G_0, separating V' and V'' in G_0.*

Proof. Let G'_3 and G''_3 be the two graphs obtained after the deletion of γ from G_3, with $V' = V(G'_3)$ and $V'' = V(G''_3)$. Recall that G_0 can be obtained from G_3 by means of operations of types (4), (5), and (6). None of these operations adds a path between V' and V'', and therefore γ is a 3-edge-cut of G_0, separating V' and V''. □

LEMMA 5.19. *G_3 has $O(|X_3 \cup Z_3|)$ vertices and edges.*

Proof. As said before, there are exactly $(|X_3 \cup Z_3| + 1)$ red vertices in G_0. Recall that a 4-edge-connected class Q_j of G_0 is red if it contains at least one red vertex. Let $r_j \geq 1$ be the number of red vertices contained in a red 4-edge-connected class Q_j, and let ρ be the total number of red 4-edge-connected classes of G_0. Since any two 4-edge-connected classes are disjoint, we have

$$\sum_{j=1}^{\rho} r_j = |X_3 \cup Z_3| + 1$$

and

$$\rho \leq |X_3 \cup Z_3| + 1.$$

Let $\mathcal{T}(G_0)$ be the cactus tree of G_0. We color red a node of $\mathcal{T}(G_0)$ only if it corresponds to a red 4-edge-connected class of G_0. Hence, there are exactly ρ red nodes in $\mathcal{T}(G_0)$. Let $\mathcal{T}'$ be the compressed cactus tree. Because of rules (Q1) and (Q2), $\mathcal{T}'$ has no black leaves and no black chains of degree-two nodes. Consequently, $\mathcal{T}'$ has $O(\rho)$ nodes and edges. Since there are at most three marked edges in G_0 for each edge in $\mathcal{T}'$, the total number of marked edges and vertices in G_0 will be $O(\rho)$. For $1 \leq i \leq p$, let ρ_i be the total number of marked vertices in G_0^i, and let σ_i be the total number of red vertices in G_0^i. Note that

$$\sum_{i=1}^{p} (\rho_i + \sigma_i) = O(\rho + |X_3 \cup Z_3|) = O(|X_3 \cup Z_3|).$$

Since each $\mathcal{C}(G_i)$ has size $O(\rho_i + \sigma_i)$, and there are $O(\rho)$ marked edges in G_3, the lemma follows. □

LEMMA 5.20. *Let H be a graph such that $V(H) \cap V(G_0) \subseteq X_3 \cup Z_3$. If $\lambda(G_0 \cup H) = \ell$, $\ell \leq 3$, then $\lambda(G_3 \cup H) \leq \ell$.*

Proof. Assume that $\lambda(G_0 \cup H) = \ell \leq 3$, and let γ be a minimum edge-cut of $G_0 \cup H$, with $|\gamma| = \ell$. Let $\gamma_{G_0} = \gamma \cap E(G_0)$ and $\gamma_H = \gamma \cap E(H)$ be, respectively, the edges of γ in G_0 and in H. Due to the minimality of γ, if $\gamma_{G_0} \neq \emptyset$, γ_{G_0} must be an edge-cut in G_0. Similarly, if $\gamma_H \neq \emptyset$, γ_H must be an edge-cut in H. In other words, denote by $G'_0 \cup H'$ and $G''_0 \cup H''$ the two graphs obtained from $G_0 \cup H$ after deleting the edges of γ, with $G'_0, G''_0 \subseteq G_0$, and $H', H'' \subseteq H$: γ_{G_0} splits G_0 into G'_0

and G_0'', while γ_H splits H into H' and H''. If $\gamma_{G_0} = \emptyset$, then γ uses no edges of G_0, and therefore it does not split G_0: in this case we assume that $G_0' = G_0$ and $G_0'' = \emptyset$. Since γ_{G_0} is an edge-cut of G_0, and G_0 is 3-edge-connected, either $\gamma_{G_0} = \emptyset$ or $|\gamma_{G_0}| \geq 3$. Since $|\gamma| \leq 3$, we are left with only two possibilities: either

(a) $\gamma_{G_0} = \emptyset$ (and therefore $\gamma = \gamma_H$), or

(b) $|\gamma_{G_0}| = 3$ and $\gamma_H = \emptyset$ (and therefore $\gamma = \gamma_{G_0}$ and $\ell = 3$).

In case (a), since $\gamma_{G_0} = \emptyset$, $G_0 = G_0'$. Since $\gamma = \gamma_H$, no path between H' and H'' can go through G_0. Since G_0 is 3-edge-connected, and therefore connected, this implies that either H' or H'' is not connected to G_0: without loss of generality, assume that $V(H') \cap V(G_0) = \emptyset$. Then $\gamma = \gamma_H$ splits $G_0 \cup H$ into $G_0 \cup H''$ and H'. Since $V(H) \cap V(G_0) = V(H) \cap V(G_3)$, we also have that $V(H') \cap V(G_3) = \emptyset$, and therefore $\gamma = \gamma_H$ splits $G_3 \cup H$ into $G_3 \cup H''$ and H'. Thus, $\lambda(G_3 \cup H) \leq \ell$, and the lemma holds in case (a).

Assume now that we are in case (b) and $\gamma = \gamma_{G_0}$ is a 3-edge-cut in $G_0 \cup H$. Since $\ell = 3$, we have to prove that $\lambda(G_3 \cup H) \leq 3$. To do this, we distinguish two subcases.

(b1) γ_{G_0} is separating two nonempty sets of red vertices, say R_1 and R_2, in G_0, or

(b2) no two red vertices are separated by γ_{G_0}.

Assume we are in case (b1). Since $\gamma_H = \emptyset$, and γ_{G_0} is a 3-edge-cut of $G_0 \cup H$ separating R_1 and R_2, there is no path in H between R_1 and R_2. As a consequence of $R_1, R_2 \neq \emptyset$, by Lemma 5.17 there is a 3-edge-cut γ' separating R_1 and R_2 in G_3. Since γ' separates R_1 and R_2 in G_3, with $R_1, R_2 \neq \emptyset$, and there is no path in H between R_1 and R_2, γ' is a 3-edge-cut of $G_3 \cup H$. If we are in case (b2), $\gamma = \gamma_{G_0}$ is a 3-edge-cut of G_0 that does not separate red vertices. Since all the vertices in $X_3 \cup Z_3$ are red, this implies that γ is an *extra* 3-edge-cut. But then there is a chosen extra 3-edge-cut $\widehat{\gamma}$ in G_0. Let v_0 be the chosen extra vertex: $\widehat{\gamma}$ separates $\widehat{R_1} = X_3 \cup Z_3$ from $\widehat{R_2} = \{v_0\}$, and v_0 is colored red. By Lemma 5.17, there is a 3-edge-cut $\widehat{\gamma}'$ separating $X_3 \cup Z_3$ and $\{v_0\}$ in G_3. Since $V(H) \cap V(G_0) \subseteq X_3 \cup Z_3$, and $v_0 \notin (X_3 \cup Z_3)$, there cannot be a path in H between $\widehat{R_1} = X_3 \cup Z_3$ and $\widehat{R_2} = \{v_0\}$. This implies that $\widehat{\gamma}'$ separates $X_3 \cup Z_3$ and v_0 in $G_3 \cup H$, and therefore $\widehat{\gamma}'$ is a 3-edge-cut of $G_3 \cup H$. In summary, in both cases (b1) and (b2), there is a 3-edge-cut in $G_3 \cup H$. Then, $\lambda(G_3 \cup H) \leq 3$, and the lemma also holds in case (b), since $\ell = 3$. ∎

LEMMA 5.21. *Let H be a graph such that $V(H) \cap V(G_0) \subseteq X_3 \cup Z_3$. If $\lambda(G_3 \cup H) = \ell \leq 3$, then $\lambda(G_0 \cup H) \leq \ell$.*

Proof. We use an argument similar to the one used in the proof of Lemma 5.20. Assume that $\lambda(G_3 \cup H) = \ell \leq 3$, and let δ be a minimum edge-cut of $G_3 \cup H$, with $|\delta| = \ell$. Let $\delta_{G_3} = \delta \cap E(G_3)$ and $\delta_H = \delta \cap E(H)$ be the edges of δ in G_3 and in H. Due to the minimality of δ, if $\delta_{G_3} \neq \emptyset$, δ_{G_3} must be an edge-cut in G_3. Similarly, if $\delta_H \neq \emptyset$, δ_H must be an edge-cut in H. Since by Lemma 5.16, G_3 is 3-edge-connected, either $\delta_{G_3} = \emptyset$ or $|\delta_{G_3}| \geq 3$. As in the proof of Lemma 5.20, $|\delta| \leq 3$ leaves only two possibilities: either (a) $\delta_{G_3} = \emptyset$ or (b) $|\delta_{G_3}| = 3$, $\delta_H = \emptyset$, and $\ell = 3$. In case (a), the same proof previously used for case (a) of Lemma 5.20 shows that $\delta = \delta_H$ is an ℓ-edge-cut in $G_0 \cup H$ too. If we are in case (b), $\delta_H = \emptyset$, and $\delta = \delta_{G_3}$. Let V_1 and V_2 be the two sets of vertices separated by δ in G_3. Because of Lemma 5.18, δ is a 3-edge-cut separating V_1 and V_2 in G_0, which implies that all the paths between V_1 and V_2 in G_0 go through δ. Since $\delta_H = \emptyset$, there is no path in H between V_1 and V_2. The latter two statements imply that all the paths between V_1 and V_2 in $G_0 \cup H$ go through δ. Thus, δ is a 3-edge-cut of $G_0 \cup H$ too. ∎

LEMMA 5.22. *For $2 \leq k \leq 4$, G_3 is a compressed global certificate of k-edge-connectivity for $(X_3 \cup Z_3)$ in G_0.*

Proof. Let H be any graph with $V(H) \cap V(G_0) \subseteq X_3 \cup Z_3$. By Lemmas 5.20 and 5.21, $\lambda(G_0 \cup H) = \ell \leq 3$ if and only if $\lambda(G_3 \cup H) = \ell \leq 3$. This implies that $\lambda(G_0 \cup H) > 3$ if and only if $\lambda(G_3 \cup H) > 3$. Hence, G_3 is a global certificate of k-edge-connectivity for $(X_3 \cup Z_3)$ in G_0, $2 \leq k \leq 4$. This certificate is compressed because of Lemma 5.19. ∎

LEMMA 5.23. *Let H be given with $V(G_0) \cap V(H) \subseteq X_3 \cup Z_3$, and let $u \in X_3 \cup Z_3 \cup V(H)$. For any vertex $v \in V(G_0)$, all the paths between u and v in $G_0 \cup H$ must contain at least one red vertex. Similarly, for any vertex $w \in V(G_3)$, all the paths between u and w in $G_3 \cup H$ must contain at least one red vertex.*

Proof. Recall that all the vertices in $X_3 \cup Z_3$ are colored red. If $u \in X_3 \cup Z_3$, u is itself a red vertex, and therefore the lemma is trivially true. Assume now that $u \in (V(H) - (X_3 \cup Z_3))$. Since $V(G_0) \cap V(H) \subseteq X_3 \cup Z_3$, G_0 and H share only red vertices, and therefore each path between a vertex in H and a vertex in G_0 must contain at least one red vertex. The same argument applies to G_3. ∎

LEMMA 5.24. *For $2 \leq k \leq 4$, G_3 is a local certificate of k-edge-connectivity for $X_3 \cup Z_3$ in G_0.*

Proof. Let H be given with $V(G_0) \cap V(H) \subseteq X_3 \cup Z_3$, and let x and y be any two vertices of $X_3 \cup Z_3 \cup V(H)$. To prove the lemma, we basically follow the same ideas used for proving Lemma 5.22, but this time we consider edge-cuts that separate x and y instead. Note that since $x, y \in X_3 \cup Z_3 \cup V(H)$, x and y are in both $G_0 \cup H$ and $G_3 \cup H$. We prove the following two properties, which are the analogs of Lemmas 5.20 and 5.21.

(i) If $\lambda_{x,y}(G_0 \cup H) = \ell \leq 3$, then $\lambda_{x,y}(G_3 \cup H) \leq \ell$.

(ii) If $\lambda_{x,y}(G_3 \cup H) = \ell \leq 3$, then $\lambda_{x,y}(G_0 \cup H) \leq \ell$.

Properties (i) and (ii) imply that $\lambda_{x,y}(G_0 \cup H) = \ell \leq 3$ if and only if $\lambda_{x,y}(G_3 \cup H) = \ell \leq 3$. Note that any two vertices whose edge connectivity is not ℓ, $\ell \leq 3$, are at least 4-edge-connected. Hence, the lemma will follow from (i) and (ii).

We first prove (i). If $\lambda_{x,y}(G_0 \cup H) = \ell \leq 3$, then there is a minimum edge-cut $\gamma(x, y)$ separating x and y in $G_0 \cup H$ such that $|\gamma(x, y)| = \ell \leq 3$. Denote by $G_0' \cup H'$ and $G_0'' \cup H''$ the two graphs obtained from $G_0 \cup H$ after deleting the edges of $\gamma(x, y)$, with $G_0', G_0'' \subseteq G_0$, and $H', H'' \subseteq H$. Without loss of generality, assume that x is in $G_0' \cup H'$ and y is in $G_0'' \cup H''$. Let $\gamma_{G_0}(x, y) = \gamma(x, y) \cap E(G_0)$ and $\gamma_H(x, y) = \gamma(x, y) \cap E(H)$ be, respectively, the edges of $\gamma(x, y)$ in G_0 and in H. Due to the minimality of $\gamma(x, y)$, if $\gamma_{G_0}(x, y) \neq \emptyset$, $\gamma_{G_0}(x, y)$ must be an edge-cut of G_0 (separating G_0' and G_0''). If $\gamma_{G_0}(x, y) = \emptyset$, then $\gamma(x, y)$ uses no edges of G_0, and therefore it does not split G_0: in this case we assume that $G_0' = G_0$ and $G_0'' = \emptyset$. Similarly, if $\gamma_H \neq \emptyset$, $\gamma_H(x, y)$ must be an edge-cut of H (separating H' and H'').

Since $\gamma_{G_0}(x, y)$ is an edge-cut of G_0, and G_0 is 3-edge-connected, either $\gamma_{G_0}(x, y) = \emptyset$ or $|\gamma_{G_0}(x, y)| \geq 3$. Since $|\gamma(x, y)| \leq 3$, we are left with only two possibilities:

(a) $\gamma_{G_0}(x, y) = \emptyset$ (and therefore $\gamma(x, y) = \gamma_H(x, y)$);

(b) $|\gamma_{G_0}(x, y)| = 3$ and $\gamma_H(x, y) = \emptyset$ (and therefore $\gamma(x, y) = \gamma_{G_0}(x, y)$).

Assume we are in case (a). Since $\gamma_{G_0}(x, y) = \emptyset$, $G_0 = G_0'$ and $\gamma_H(x, y)$ splits $G_0 \cup H$ into $G_0 \cup H''$ and H'. Due to the fact that $\gamma_H(x, y)$ separates x and y, one of these vertices must be in H', and the other must be in $G_0 \cup H''$. Since $V(G_3) \subseteq V(G_0)$, we have that $V(H') \cap V(G_3) = \emptyset$. Thus, $\gamma_H(x, y)$ splits $G_3 \cup H$ into $G_3 \cup H''$ and H', and therefore it separates x and y in $G_3 \cup H$ too. This yields (i) for this case.

Assume now that we are in case (b). Since $\gamma(x, y) = \gamma_{G_0}(x, y)$ is a 3-edge-cut separating x and y in $G_0 \cup H$, it splits G_0 into G_0' and G_0'', which are both non-empty. Assume that x is in the same side as G_0' and y is in the same side as G_0''.

Let R' and R'' be the sets of red vertices, respectively, in G'_0 and in G''_0. Recall that $V(H) \cap V(G_3) \subseteq X_3 \cup Z_3$, and vertices of $X_3 \cup Z_3$ are colored red. Because of Lemma 5.23, there is a red vertex (of R') in each path between x and a vertex in G'_0, and there is a red vertex (of R'') in each path between y and a vertex in G''_0. This shows that $R', R'' \neq \emptyset$, and therefore by Lemma 5.17, there is a 3-edge-cut $\widehat{\gamma}$ separating R' and R'' in G_3. Again, because of Lemma 5.23, any path between x and y that goes through G_3 must contain at least one vertex in R' and one vertex in R''. Since R' and R'' are separated by $\widehat{\gamma}$ in G_3, any such path between x and y must contain at least one edge of $\widehat{\gamma}$. As a consequence of $\gamma_H(x, y) = \emptyset$, no path between x and y can go through H. Then, every path between x and y in $G_3 \cup H$ must go through $\widehat{\gamma}$, thus giving (i).

We now turn to (ii). Assume that the minimum edge-cut $\delta(x, y)$ separating x and y in $G_3 \cup H$ is of cardinality $|\delta(x, y)| = \ell \leq 3$. Let $\delta_{G_3}(x, y) = \delta(x, y) \cap E(G_3)$ and $\delta_H(x, y) = \delta(x, y) \cap E(H)$ be the edges of $\delta(x, y)$ in G_3 and in H. Due to the minimality of $\delta(x, y)$, if $\delta_{G_3}(x, y) \neq \emptyset$, $\delta_{G_3}(x, y)$ must be an edge-cut in $G_3(x, y)$. Similarly, if $\delta_H(x, y) \neq \emptyset$, $\delta_H(x, y)$ must be an edge-cut in H. Since G_3 is 3-edge-connected, either $\delta_{G_3}(x, y) = \emptyset$ or $|\delta_{G_3}(x, y)| \geq 3$. Once again, we have two possibilities: either (a) $\delta_{G_3}(x, y) = \emptyset$ or (b) $|\delta_{G_3}(x, y)| = 3$ and $\delta_H(x, y) = \emptyset$. In case (a), the same argument used for case (a) of (i) shows that $\delta(x, y) = \delta_H(x, y)$ is an ℓ-edge-cut separating x and y in $G_0 \cup H$ too, and therefore (ii) holds. If we are in case (b), $\delta_H(x, y) = \emptyset$, and $\delta(x, y) = \delta_{G_3}(x, y)$, $|\delta_{G_3}(x, y)| = 3$. Let R' and R'' be the red vertices separated by $\delta(x, y)$ in G_3. Exactly as in (i), any path between x and y that goes through G_3 must contain at least one (red) vertex in R' and one (red) vertex in R''. As before, this implies two things. First, by Lemma 5.17, there is a 3-edge-cut $\widehat{\delta}$ separating R' and R'' in G_0. Second, any path between x and y in G_0 must contain at least one edge of $\widehat{\delta}$. As a consequence of $\gamma_H(x, y) = \emptyset$, no path between x and y can go through H. Then, every path between x and y in $G_0 \cup H$ must go through $\widehat{\delta}$. Thus, $\widehat{\delta}$ is a 3-edge-cut separating x and y in $G_0 \cup H$. Since $\ell = 3$ in case (b), this proves (ii). □

LEMMA 5.25. *G_3 preserves planarity.*

Proof. Let H be any graph such that $V(H) \cap V(G_0) \subseteq X_3 \cup Z_3$. $G_3 \cup H$ can be obtained from $G_0 \cup H$ by means of operations (1), (2), and (3), which consist of deletion of edges, contraction of edges, and insertion of parallel edges. Since all these operations preserve planarity, if $G_0 \cup H$ is planar, then $G_3 \cup H$ is planar. Thus, G_3 preserves planarity according to Definition 3.5. □

Lemmas 5.16, 5.22, 5.24, and 5.25 prove that G_3 is a solution to Problem 5.3.

5.4. Compressed certificates for 3- and 4-edge-connectivity. In this section, we combine the results obtained in sections 5.1, 5.2, and 5.3 to derive compressed full certificates for 3- and 4-edge-connectivity. Let $G = (V, E)$ be a planar connected graph, and let X be a set of interesting vertices in G. Let H be given, with $V(G) \cap V(H) \subseteq X$. Our algorithm for finding compressed full certificates consists of the following steps.

Step 1 (*decreasing the number of* 2*-edge-connected components*). Set $X_1 = X$, $G_0 = G$, and compute a solution G_1 to Problem 5.1 as shown in section 5.1.

Step 2 (*decreasing the number of* 3*-edge-connected components*). Let N_1 be the set of endpoints of bridges left in G_1 at the end of Step 1. Because of the previous step, there are $O(|X|)$ bridges in G_1 and therefore $|N_1| = O(|X|)$. For each 2-edge-connected component B left in G_1 do the following: set $X_2(B) = (X_1 \cup N_1) \cap V(B)$, $G_0(B) = B$, and compute a solution $G_2(B)$ to Problem 5.2 as shown in section 5.2.

Let $X_2 = \sum_B X_2(B)$. Note that $|X_2| = \sum_B |X_2(B)| = |X_1 \cup N_1| = O(|X|)$, so this compression uses $O(|X|)$ interesting vertices overall. Denote by G_2 the graph obtained from G_1 by replacing each 2-edge-connected component B with $G_2(B)$. Since each $G_2(B)$ is a solution to Problem 5.2, $G_2(B)$ has $O(|X_2(B)|)$ 3-edge-connected components. Hence, the overall number of 3-edge-connected components left in G_2 will be $\sum_B O(|X_2(B)|) = O(|X|)$.

Step 3 (*final compression*). Let N_2 be the set of endpoints of 2-edge-cuts left in G_2 at the end of Step 2. Since the total number of endpoints of 2-edge-cuts is linear in the number of 3-edge-connected components, because of the previous step, $|N_2| = O(|X|)$. Let T be a 3-edge-connected component of G_2: recall that T is obtained by means of splits (as defined in section 4.1), and it contains virtual edges corresponding to 2-edge-cuts. Let $M_2(T)$ be the set of virtual edges in T. For each 3-edge-connected component T left in G_2 do the following: set $X_3(T) = (X_2 \cup N_2) \cap V(T)$, $G_0(T) = T$, $Y_3(T) = M_2(T)$, and compute a solution $G_3(T)$ to Problem 5.3 as shown in section 5.3. Let $X_3 = \sum_T X_3(T)$ and $M_2 = \sum_T M_2(T)$ be, respectively, the total number of red vertices and red edges. Note that $|X_3| = \sum_T |X_3(T)| = |X_2 \cup N_2| = O(|X|)$ and $|M_2| = \sum_T |M_2(T)| \leq \sum_T |N_2 \cap V(T)| \leq \sum_T |X_3(T)| = O(|X|)$, so again the compression uses $O(|X|)$ interesting vertices overall. Then merge back the compressed 3-edge-connected components along their original 2-edge-cuts. This is well defined, since all the virtual edges, and hence the endpoints of the original 2-edge-cuts, are colored red and therefore preserved in the graphs $G_3(T)$. Denote by G_3 the graph obtained from G_2 in this way.

We now prove that G_3 is a planarity-preserving compressed full certificate of 3- and 4-edge-connectivity for X in G.

LEMMA 5.26. *Let $G = (V, E)$ be an undirected planar graph with a set $X \subseteq V(G)$. Let G_3 be the graph defined at the end of Step* 3 *above. Then G_3 is a planarity-preserving compressed full certificate for X in G with respect to* 3- *and* 4*-edge-connectivity.*

Proof. Let G_1 be the graph obtained after Step 1. Since G_1 is a solution to Problem 5.1, then for every $k \geq 2$, G_1 is a planarity-preserving full certificate of k-edge-connectivity for X in G and has $O(|X|)$ 2-edge-connected components.

Let G_2 be the graph obtained from G_1 by replacing each 2-edge-connected component B with $G_2(B)$, where each $G_2(B)$ is a solution to Problem 5.2 with interesting vertices $X_2(B)$ as described in Step 2. We claim that for every $k \geq 2$, G_2 is a planarity-preserving compressed full certificate for k-edge-connectivity of X in G. Since each $G_2(B)$ is planarity-preserving, as shown in section 5.2, and G_2 is obtained by only adding bridges between the graphs $G_2(B)$, G_2 will be planarity-preserving too. The fact that G_2 is compressed follows easily from repeated applications of Lemma 5.11. It remains to show that G_2 is indeed a certificate for X in G. To prove this, fix a particular 2-edge-connected component B of G_1. Let $X_2(B)$ be the interesting vertices of B as defined in Step 2, and let $G_2^{(1)}$ be the graph obtained after replacing B with $G_2(B)$. By Lemma 5.13 and Corollary 5.2, for every $k \geq 2$, $G_2(B)$ is a full certificate for k-edge-connectivity of $X_2(B)$ in B. By Lemma 3.2 applied with $C' = G_2(B)$, $X' = X_2(B)$, $G' = B$, $C'' = G'' = G_1 - B$, and $X'' = V(G_1 - B)$, we have that for every $k \geq 2$, $G_2^{(1)}$ is a full certificate for k-edge-connectivity of $X_2(B) \cup V(G_1 - B)$ in G_1. Since $X \cup X_2(B) \subseteq X_2(B) \cup V(G_1 - B)$, by Lemma 3.3 we have that for every $k \geq 2$, $G_2^{(1)}$ is a full certificate for k-edge-connectivity of $X \cup X_2(B)$ in G_1. Repeating this argument for each 2-edge-connected component B of G_1 yields that for every $k \geq 2$, G_2 is a full certificate for k-edge-connectivity of X_2 in G_1. Since G_1 is a full

certificate of k-edge-connectivity for X in G, and $X \subseteq X_2$, by Lemma 3.1 we have that G_2 is a full certificate of k-edge-connectivity for X in G.

Let G_3 be the graph obtained at the end of Step 3. We claim that G_3 is a planarity-preserving compressed full certificate for 3- and 4-edge-connectivity of X in G. By Lemma 5.25, each $G_3(T)$, which is a solution to Problem 5.3, is planar. Recall that G_3 is obtained by splitting G_2, replacing each 3-edge-connected component T with $G_3(T)$, and by merging back the compressed 3-edge-connected components $G_3(T)$. Since splits and merges preserve planarity, G_3 will be planarity-preserving. The fact that G_3 is compressed follows from repeated applications of Lemma 5.19. To show that G_3 is a certificate of 3- and 4-edge-connectivity for X in G, we proceed as follows. Fix a particular 3-edge-connected component T of G_2, and let $B(T)$ be the 2-edge-connected component containing T, whose interesting vertices (defined in step 2) are in $X_2(B)$. Let $X_3(T)$ be the interesting vertices of T as defined in Step 3, and let $G_3^{(1)}$ be the graph obtained from G_2 after replacing T with $G_3(T)$. By Lemmas 5.22 and 5.24, for $2 \le k \le 4$, $G_3(T)$ is a full certificate for k-edge-connectivity of $X_3(T)$ in T. Let $B(T)^{(1)}$ be the 2-edge-component of $G_3^{(1)}$ containing $G_3(T)$. By Theorem 4.2 applied to the 2-edge-connected graph $B(T)$, we have that for $2 \le k \le 4$, $B(T)^{(1)}$ is a full certificate for k-edge-connectivity of $X_2(B)$ in $B(T)$. Applying Lemma 3.2 as before yields that for $2 \le k \le 4$, $G_3^{(1)}$ is a full certificate for k-edge-connectivity of $X_2(B) \cup V(G_2 - B(T))$ in G_2. Again, by Lemma 3.3, we have that for $2 \le k \le 4$, $G_3^{(1)}$ is a full certificate for k-edge-connectivity of $X \cup X_2(B)$ in G_2. Repeating this argument for each 3-edge-connected component T of G_2 yields that for $2 \le k \le 4$, G_3 is a full certificate for k-edge-connectivity of X in G_2. Since we have already proved that, for any $k \ge 2$, G_2 is a full certificate of k-edge-connectivity for X in G, by Lemma 3.1 we have that G_3 is a full certificate of k-edge-connectivity for X in G, $2 \le k \le 4$. □

THEOREM 5.1. *We can maintain a planar graph subject to insertions and deletions that preserve planarity, and allow testing the* 3- *and* 4*-edge-connectivity of the graph in $O(n^{1/2})$ time per update, or test whether two vertices are* 3- *or* 4*-edge-connected, in $O(n^{1/2})$ time per update or query.*

Proof. Lemma 5.26 shows that planarity-preserving compressed full certificates exist, so we can use Lemma 3.4 to find such certificates in linear time. We then apply Theorem 3.4 with $T(n) = Q(n) = O(n)$. □

6. Vertex connectivity. As another application of our basic sparsification technique, we describe an algorithm for 2- and 3-vertex-connectivity.

THEOREM 6.1. *We can maintain a planar graph, subject to insertions and deletions that preserve planarity, and allow queries that test the* 2- *or* 3*-vertex-connectivity of the graph, or test whether two vertices belong to the same* 2- *or* 3*-vertex-connected component, in $O(n^{1/2})$ amortized time per update or query.*

Proof. Galil, Italiano, and Sarnak [21] show that compressed certificates for 2- and 3-vertex-connectivity can be found in linear time. It can be verified that their certificates comply with our Definition 3.3. Using their compressed certificates in our separator tree gives $O(n^{1/2})$ amortized time per update by Theorem 3.2. □

We remark that the same $O(n^{1/2})$ bound can be achieved for the problem of maintaining minimum spanning forests and for local connectivity. This does not improve the $O(n^{1/2})$ bounds that can be achieved using the algorithms for general graphs [9], however. In the companion paper [11], we achieve better $O(\log^2 n)$ bounds for these problems.

7. Conclusions and open problems. We have introduced a new and general technique for designing dynamic planar graph algorithms. This technique is based upon sparsification, compressed certificates, and balanced separator trees, and improves many known bounds. In the companion paper [11], we have showed how this technique produces fast, fully dynamic algorithms for minimum spanning forests, for 2-edge-connectivity on planar graphs, and for dynamic planarity testing. In this paper we have applied this technique to 2-vertex-, 3-vertex-, 3-edge-, and 4-edge-connectivity. For edge connectivity, we have developed certificates which may be interesting on their own. There are a number of related and perhaps interesting questions.

The algorithms described in [11] exploit a stability property in their certificates to support polylogarithmic time updates in planar graphs. For the problems tackled in this paper, we have not been able to apply stable sparsification, and our bounds are $O(n^{1/2})$. Is it possible to exploit stability and to achieve polylogarithmic bounds for these problems too?

Furthermore, are there compressed certificates for other problems? For higher edge connectivity, for instance, the cactus tree gets more complicated (see, for instance, [6]) and makes our way of computing compressed certificates for edge connectivity difficult to generalize. Are there compressed certificates for edge connectivity which are not based on cactus trees?

Finally, can we exploit known partially dynamic algorithms to speed up the insertion times in our bounds? This would be particularly appealing, since there are very fast, partially dynamic algorithms already available in the literature for edge and vertex connectivity [6, 20, 27, 29, 30, 36].

Acknowledgments. We are deeply indebted to the anonymous referees, whose thorough reading of the paper lead to a substantial improvement in its presentation.

REFERENCES

[1] J. Cheriyan and S. N. Maheshwari, *Finding nonseparating induced cycles and independent spanning trees in* 3*-connected graphs*, J. Algorithms, 9 (1988), pp. 507–537.

[2] J. Cheriyan, M. Y. Kao, and R. Thurimella, *Scan-first search and sparse certificates: An improved parallel algorithm for k-vertex connectivity*, SIAM J. Comput., 22 (1993), pp. 157–174.

[3] G. B. Dantzig and D. R. Fulkerson, *On the max-flow min-cut theorem of networks*, in Linear Inequalities and Related Systems, Ann. Math. Study 38, Princeton University Press, Princeton, NJ, 1956, pp. 215–221.

[4] E. A. Dinitz, A. V. Karzanov, and M. V. Lomonosov, *A structure of the system of all minimal cuts of a graph*, in Studies in Discrete Optimization, A. A. Fridman, ed., Nauka, Moscow, 1976, pp. 290–306. (In Russian.)

[5] E. A. Dinitz, *The* 3*-edge-components and a structural description of all* 3*-edge-cuts in a graph*, in Proc. 18th Int. Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science 657, Springer-Verlag, New York, 1992, pp. 145–157.

[6] E. A. Dinitz, *Maintaining the* 4*-edge-connected components of a graph on-line*, in Proc. 2nd Israel Symp. on Theory of Computing and Systems, Nethania, Israel, IEEE Computer Society Press, Piscataway, NJ, 1993, pp. 88–97.

[7] D. Eppstein, *Subgraph isomorphism for planar graphs and related problems*, in Proc. 6th ACM-SIAM Symp. on Discrete Algorithms, 1995, pp. 189–196; J. Graph Algorithms Appl., to appear.

[8] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, *Sparsification—A technique for speeding up dynamic graph algorithms*, in Proc. 33rd IEEE Symp. on Foundations of Computer Science, 1992, pp. 60–69.

[9] D. Eppstein, Z. Galil, and G. F. Italiano, *Improved Sparsification*, Tech. Report 93-20, Department of Information and Computer Science, University of California, Irvine, 1993.
[10] D. Eppstein, Z. Galil, G. F. Italiano, and T. H. Spencer, *Separator based sparsification for dynamic planar graph algorithms*, in Proc. 25th Annual ACM Symp. on Theory of Computing, 1993, pp. 208–217.
[11] D. Eppstein, Z. Galil, G. F. Italiano, and T. H. Spencer, *Separator based sparsification* I*: Planarity testing and minimum spanning trees*, J. Comput. System Sci., Special issue on STOC 93, 52 (1996), pp. 3–27.
[12] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung, *Maintenance of a minimum spanning forest in a dynamic plane graph*, J. Algorithms, 13 (1992), pp. 33–54.
[13] A. Frank, T. Ibaraki, and H. Nagamochi, *On sparse subgraphs preserving connectivity properties*, J. Graph Theory, 17 (1993), pp. 275–281.
[14] G. N. Frederickson, *Data structures for on-line updating of minimum spanning trees, with applications*, SIAM J. Comput., 14 (1985), pp. 781–798.
[15] G. N. Frederickson, *Ambivalent data structures for dynamic* 2*-edge-connectivity and k smallest spanning trees*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 632–641.
[16] H. N. Gabow, *A matroid approach to finding edge connectivity and packing arborescences*, in Proc. 23rd ACM Symp. on Theory of Computing, 1991, pp. 112–122.
[17] H. N. Gabow, *Applications of a poset representation to edge connectivity and graph rigidity*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 812–821.
[18] H. N. Gabow and M. Stallman, *Efficient algorithms for graphic matroid intersection and parity*, in Proc. 12th Int. Coll. Automata, Languages, and Programming, Lecture Notes in Computer Science 194, Springer-Verlag, New York, 1985, pp. 210–220.
[19] Z. Galil and G. F. Italiano, *Maintaining biconnected components of dynamic planar graphs*, in Proc. 18th Int. Colloq. Automata, Languages, and Programming, Lecture Notes in Computer Science 510, Springer-Verlag, New York, 1991, pp. 339–350.
[20] Z. Galil and G. F. Italiano, *Maintaining the* 3*-edge-connected components of a graph on-line*, SIAM J. Comput., 22 (1993), pp. 11–28.
[21] Z. Galil, G. F. Italiano, and N. Sarnak, *Fully dynamic planarity testing*, in Proc. 24th ACM Symp. on Theory of Computing, 1992, pp. 495–506.
[22] D. Giammarresi and G. F. Italiano, *Decremental* 2*- and* 3*-connectivity on planar graphs*, Algorithmica, 16 (1996), pp. 263–287.
[23] M. T. Goodrich, *Planar separators and parallel polygon triangulation*, in Proc. 24th ACM Symp. on Theory of Computing, 1992, pp. 507–516.
[24] J. Hershberger, M. Rauch, and S. Suri, *Data structures for two-edge connectivity on planar graphs*, Theoret. Comput. Sci., 130 (1994), pp. 139–161.
[25] J. Hopcroft and R. E. Tarjan, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158.
[26] A. Itai and M. Rodeh, *The multi-tree approach to reliability in distributed networks*, Inform. and Comput., 79 (1988), pp. 3–59.
[27] A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen, *On-line maintenance of the four-connected components of a graph*, in Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991, pp. 793–801.
[28] A. V. Karzanov and E. A. Timofeev, *Efficient algorithm for finding all minimal edge-cuts of a nonoriented graph*, Cybernetics, 1986, pp. 156–162. (Trans. from Kibernetica, 2(1986), pp. 8–12.)
[29] J. A. La Poutré, *Dynamic Graph Algorithms and Data Structures*, Ph.D. Thesis, Utrecht University, the Netherlands, September 1991.
[30] J. A. La Poutré, *Maintenance of triconnected components of graphs*, in Proc. 19th Int. Coll. Automata, Languages, and Programming, Lecture Notes in Computer Science 623, Springer-Verlag, New York, 1992, pp. 354–365.
[31] H. Nagamochi and T. Ibaraki, *A linear-time algorithm for finding a sparse* k*-connected spanning subgraph of a* k*-connected graph*, Algorithmica, 7 (1992), pp. 583–596.
[32] M. Rauch, *Fully dynamic biconnectivity in graphs*, in Proc. 33rd IEEE Symp. on Foundations of Computer Science, 1992, pp. 50–59.
[33] R. Tamassia, *A dynamic data structure for planar graph embedding*, in Proc. 15th Int. Colloq. Automata, Languages, and Programming, Lecture Notes in Computer Science 317, Springer-Verlag, New York, 1988, pp. 576–590.
[34] R. E. Tarjan, *Depth-first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.

[35] R. Thurimella, *Techniques for the Design of Parallel Graph Algorithms*, Ph.D. Thesis, Univ. of Texas, Austin, 1989.

[36] J. Westbrook and R. E. Tarjan, *Maintaining bridge-connected and biconnected components on-line*, Algorithmica, 7 (1992), pp. 433–464.