Construction Sequences and Certifying 3-Connectedness

Technical Report B 09-01

Jens M. Schmidt*
Institute of Computer Science
Freie Universität Berlin, Germany

Abstract

Given two 3-connected graphs G and H, a *construction sequence* constructs G from H (e. g. from the K_4) with three basic operations, called the *Barnette-Grünbaum operations*. These operations are known to be able to construct all 3-connected graphs. We extend this result by identifying every intermediate graph in the construction sequence with a subdivision in G and showing under some minor assumptions that there is still a construction sequence to G when we start from an *arbitrary prescribed* H-subdivision. This leads to the first algorithm that computes a construction sequence in time $O(|V(G)|^2)$. As an application, we develop a certificate for the 3-connectedness of graphs that can be easily computed and verified. Based on this, a certifying test on 3-connectedness is designed.

keywords: construction sequence, 3-connected graph, nested subdivisions, inductive characterization, 3-connectedness, certifying algorithm

1 Introduction

Barnette and Grünbaum [1] proved 1969 that every 3-connected graph G can be constructed from the K_4 by iteratively applying three basic operations. Titov and Kelmans [9, 5] extended this result by allowing it to start with arbitrary 3-connected graphs H instead of K_4, as long as a subdivision of H is contained in G. This is a generalization of Barnette and Grünbaums theorem, since every 3-connected graph contains a subdivision of the K_4.

Although both theorems are used frequently in graph theory (see e. g. [8]), we are not aware of any computational results that find such a construction sequence. Moreover, efficient algorithms cannot be derived from the existence proofs, as both depend heavily on adding longest paths, which are NP-hard to find. Nevertheless, we show that it is possible to find a construction sequence

*This research was supported by the Deutsche Forschungsgemeinschaft within the research training group "Methods for Discrete Structures" (GRK 1408). Email: jens.schmidt@inf.fu-berlin.de).

of G in time $O(|V(G)|^2)$, if the subdivision of H is part of the input and intermediate graphs in the construction can have parallel edges.

The construction sequence leads to a certificate for the 3-connectedness of G that is easy to compute and can be verified in $O(|E(G)|)$. Blum and Kannan [2] introduced the concept of *certifying algorithms* that give a proof of correctness along with their output. Achieving such algorithms is a major goal for problems where the fastest solutions known are complicated and difficult to implement. Based on our certificate, we develop a *certifying* and simple 3-connectedness test for graphs with running time $O(|V(G)|^2)$. It remains open whether this certificate can even be found in linear time.

2 Preliminaries

Let $G = (V, E)$ be a finite graph with $n = |V|$, $m = |E|$, $V(G) = V$ and $E(G) = E$. Then G is *connected* if there is a path between every two nodes and *disconnected* otherwise. Let $k \geq 1$. Then a graph G is *k-connected* if $n > k$ and deleting every $k - 1$ nodes leaves a connected graph. A node (a pair of nodes) that leaves a disconnected graph upon deletion is called a *cut vertex* (*separation pair*). Note that k-connectedness here does not depend on parallel edges or self-loops. If not stated otherwise, all graphs can have parallel edges but have no self-loops, although all results can be adjusted to deal with them.

A *subdivision* of a graph G replaces each edge by a non-empty path. Conversely, we want a notation to get back to the graph without subdivided edges. For a node $v \in V(G)$ of degree two and not incident to a self-loop let $smooth_v(G)$ be the graph obtained from G by deleting v followed by adding an edge between its neighbors (we say v is *smoothed*). If v is not of degree two or incident to a self-loop let $smooth_v(G) = G$. Let $smooth(G)$ be the graph obtained by smoothing every node. Note that smoothing a graph can always be reversed by subdividing $smooth(G)$ again.

A path leading from node x to node y is denoted by $x \to y$. For a node x in G let $N(x) = \{y \mid xy \in E(G)\}$ denote its set of neighbors. For an edge e in G let $G \setminus e$ be the graph obtained from G by deleting the edge e. Let K_n be the complete graph on n nodes.

Lemma 1. (J. Isbell [1]) *Every 3-connected graph G contains a subdivision of the K_4.*

The following are well-known corollaries of Menger's theorem [6].

Lemma 2. (Fan Lemma) *Let x be a node in a graph G that is k-connected with $k \geq 1$ and let A be a set of at least k nodes in G with $x \notin A$. Then there are k internally node-disjoint paths $P_1, \ldots, P_k$ from x to distinct nodes $a_1, \ldots, a_k \in A$ such that for each of these paths $V(P_i) \cap A = a_i$.*

Lemma 3. (Expansion Lemma) *Let G be a k-connected graph. Then the graph obtained by adding a new node x with at least k neighbors in G is still k-connected.*

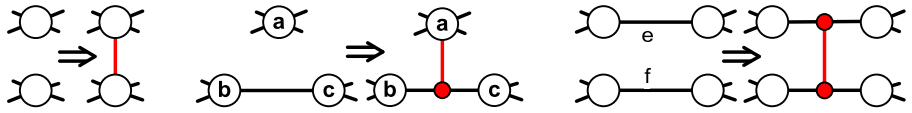
We define the Barnette and Grünbaum operations (*BG-operations*) as follows (see Figures 11(a)-1(c)).

(a) add an edge xy (possibly a parallel edge)

(b) subdivide an edge e by a node x and add the edge xy for a node $y \notin e$

(c) subdivide two distinct, non-parallel edges by nodes x and y, respectively, and add the edge xy

Let the *added edge* of an BG-operation be the edge xy.



(a) parallel edges allowed (b) a, b, c distinct (c) $e \neq f$, e and f not parallel

Figure 1: The operations of Barnette and Grünbaum.

Lemma 4. [1] *Performing a BG-operation on a* 3*-connected graph preserves* 3*-connectedness.*

Conversely, every 3-connected graph can be constructed using operations (a), (b) and (c) [1]. We call such a construction from another 3-connected graph using a fixed set of operations a *construction sequence* of G. If not stated otherwise, the set of operations is $\{(a), (b), (c)\}$. Then all intermediate graphs in the sequence are 3-connected by Lemma 4.

Let an operation be *basic*, if it does not create parallel edges and let a construction sequence be basic, if it only uses basic operations. Of course we cannot expect all operations to be basic when G is a multigraph. However, on simple graphs Barnett and Grünbaum proved that basic operations suffice.

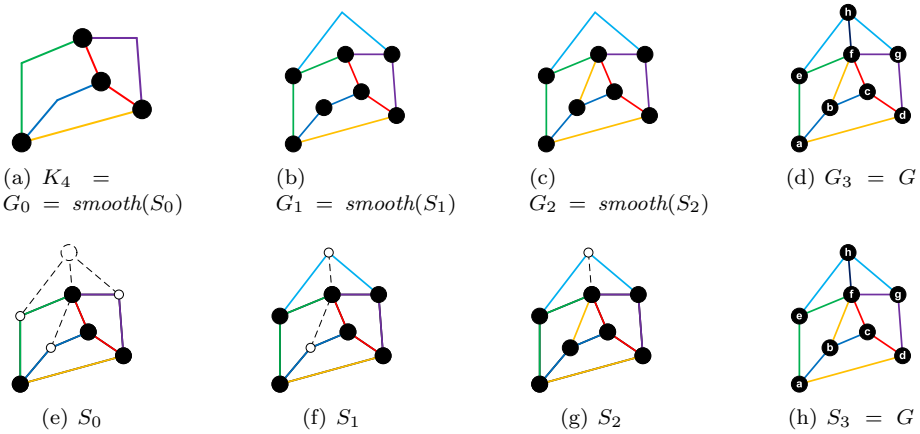
Theorem 5. [1] *A* simple *graph G is* 3*-connected if and only if G can be constructed from the K_4 using* basic *BG-operations.*

We want an operation inverse to a BG-operation. Let *removing* an edge $e = xy$, $x \neq y$ be that operation, consisting of deleting e followed by smoothing x and y.

Let $K_4 = G_0, G_1, \ldots, G_z = G$ be the 3-connected graphs obtained in a construction sequence Q. We can reverse Q by starting with G and removing the edges in the inverse order of how they were added. Suppose we would delete the edges instead of removing them and identify each emerging path with its corresponding added edge of the original construction sequence (see Figure 2). Then iteratively paths have to be deleted to get G_{i-1} from G_i and we obtain the sequence $S(Q)$ of subdivisions $S_0, \ldots, S_z$ in G with $S_z = G$ and S_0 being a subdivision of the K_4. We set this out as a proposition.

Proposition 6. *Let G and G_0 be graphs and let O be a construction sequence of G that starts with G_0 and uses BG-operations. Then G contains a subdivision of G_0. Furthermore, there is a unique subdivision $S_0(Q)$ in G that is obtained by deleting the paths corresponding to added edges of BG-operations in inverse order.*

Therefore, each graph G_i in our construction sequence can be identified with a unique subdivision $S_i(Q)$ contained in G. Conversely, $G_i = \mathit{smooth}(S_i)$ for



(a) $K_4 = G_0 = smooth(S_0)$ (b) $G_1 = smooth(S_1)$ (c) $G_2 = smooth(S_2)$ (d) $G_3 = G$

(e) S_0 (f) S_1 (g) S_2 (h) $S_3 = G$

Figure 2: The graphs $G_0, \ldots, G_z$ and $S_0, \ldots, S_z$ of a construction sequence of G. On graphs S_i the dashed edges and nodes are in G but not in S_i and nodes depicted in black are *real* nodes. The path $C_0 = e \to h \to g$ is a *chain* for S_0, yielding S_1. The *chain links* of S_1 are the paths C_0, $a \to b \to c$ and the edges in $E(S_1)$ with endnodes of degree 3.

all $0 \leq i \leq z$, since smoothing a graph is exactly the inverse operation of subdividing a graph without nodes of degree two. All nodes in S_i with degree at least three are called *real* nodes, because they correspond to nodes in G_i.

Note that, although S_i is simple if G is simple, $smooth(S_i)$ can have parallel edges in non-basic construction sequences. We define the *chain links* $L(S_i)$ of each S_i to be the unique paths in S_i with only their endnodes being real. The chain links $L(S_i)$ partition $E(S_i)$ because S_i is connected, has minimum degree two and is not a cycle. Let two chain links be *parallel* if they share the same endnodes.

Definition 7. A *chain* for S_i is a path $P = x \to y$ in G with the following properties:

1. $S_i \cap P = \{x, y\}$
2. x and y are not both contained in a chain link of S_i except as endnodes
3. x and y are not inner nodes of parallel chain links of S_i

Every chain for S_i corresponds to a BG-operation on G_i and possibly separates some chain links of S_i by introducing new real nodes. Construction sequences are not bound to start with the K_4. Titov and Kelmans [9, 5] extended Theorem 5 to sequences starting from subdivisions in G of arbitrary 3-connected graphs.

Theorem 8. [9, 5] *Let G_0 be a 3-connected graph. Then a simple graph G is 3-connected and contains a subdivision of G_0 if and only if G can be constructed from G_0 using basic BG-operations.*

3 Prescribing Subdivisions

Both theorems 5 and 8 choose a very special subdivision of the K_4 (resp. G_0) on which the construction sequence starts, in fact one in G having the maximum number of edges. The construction sequence is then obtained by adding longest chains. Unfortunately, computing these depends heavily on solving the longest paths problem, which is known to be NP-hard even in 3-connected graphs [3].

That gives rise to the question whether theorems 5 and 8 can be strengthened to start at a *prescribed* subdivision $H \subseteq G$ of G_0 instead of an arbitrary one. Note that this is equivalent to the constraint $S_0(Q) = H$ in each construction sequence Q. Such a result would provide an efficient computational access to construction sequences, since it would allow us to search the neighborhood of H for chains, yielding a new prescribed subdivision.

However, in general it is not possible to prescribe H, as the minimal counterexample in Figure 3 shows: Consider the graph G consisting of a $K_4 = H$ depicted in black and an additional node connected to three nodes of the K_4. Then every chain for H will create a parallel chain link, although G is simple. Thus, no basic operation can be applied.

But what if we drop the condition that construction sequences have to be basic? The following theorem shows that at this expense we can indeed start a construction sequence from any prescribed subdivision.

Theorem 9. *Let G be a 3-connected graph and $H \subset G$ with H being a subdivision of a 3-connected graph. Then there is a chain for H in G.*

Proof. We find a chain for $H \neq G$ by distinguishing two cases.

- $H \neq \mathit{smooth}(H)$.
  Then some chain link $T \in L(H)$ contains an inner node x, which has degree two in H. Let Q be the set of paths in G from x to a node in $V(H) \setminus V(T)$ avoiding the endnodes of T. By the 3-connectedness of G, Q cannot be empty and every path in Q fulfills 7.2. Moreover, there is at least one path $P = x \to y$ in Q with y being not contained in a parallel chain link of T, because otherwise the endnodes of T would be a separation pair. Let x' be the last node in P that is in T or in a parallel chain link of T and let y' be the first node after x' that is in $V(H)$. Then $x' \to y'$ is a chain for H, since it has properties 7.1 and 7.3.

- $H = \mathit{smooth}(H)$.
  Then H consists only of real nodes and since $H \neq G$, there is a node in $V(G) \setminus V(H)$ or an edge in $E(G) \setminus E(H)$. At first, assume that there is a node $x \in V(G) \setminus V(H)$. Then, by the 2-connectedness of G and Fan Lemma 2 we can choose a path $P = y_1 \to x \to y_2$ with no other nodes in H than y_1 and y_2. For P the properties 7.1-7.3 hold, because no chain link in $L(H)$ can have inner nodes. Let now $V(G) = V(H)$ and e an edge in $E(G) \setminus E(H)$. Then e must be a chain for H, since both endnodes are real.

□

In Theorem 9, non-basic operations can only occur in case $H = \mathit{smooth}(H)$ when a path through a node of $V(G) \setminus V(H)$ is chosen. Although we cannot avoid that, it is possible to yield a construction that is basic by augmenting $\{(a), (b), (c)\}$ with the new operation (d), which is called the *expand* operation:

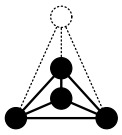


Figure 3: Every possible BG-operation adds a parallel edge.

(d) connect a new node to three distinct nodes

Whenever we encounter a node in $V(G) \setminus V(H)$ in Theorem 9, we know by the Fan Lemma 2 and the 3-connectedness of G that there are three internally node-disjoint paths to real nodes in H. Adding those paths corresponds to an expand operation, which is basic, because each path ends on the new node. This gives the following result (note that G is only assumed to be simple because some operations have to be basic).

Theorem 10. *Let G be a simple graph and let H be a graph with* smooth(H) *being* 3*-connected. Then*

$$
\begin{aligned}
& G \text{ is 3-connected and } H \subseteq G && (1)\\
\Leftrightarrow\ & \exists \text{ construction sequence } Q \text{ of } G \text{ with } S_0(Q) = H \text{ using BG-operations} && (2)\\
\Leftrightarrow\ & \exists \text{ basic construction sequence } Q' \text{ of } G \text{ with } S_0(Q') = H \text{ using BG-operations and (d)} && (3)
\end{aligned}
$$

Proof. The suffiency parts hold with Lemmas 3, 4 and 6. Necessity follows immediately from applying Theorem 9 iteratively with or without the additional expand operation. □

4 Computing Construction Sequences

Let G be a 3-connected (multi-)graph and let the prerequisites of Theorem 10 and 10.(1) hold with $H \neq G$. Then there is a non-empty construction sequence Q of G starting with H and there exists one last BG-operation in Q adding an edge $e \in E(G) \setminus E(H)$. If we would know e we could undo that operation by removing e and yield a smaller graph that is still 3-connected. Repeating this procedure until only graph H is left would give us Q.

Unfortunately we do not know e, but a straight-forward approach is to try each edge $e \in E(G) \setminus E(H)$ and check the graph G' obtained by removing e on 3-connectedness and on having a BG-operation that gets back to G by adding e. The following Lemma gives a necessary and sufficient characterization of these edges.

Lemma 11. *Let G' be a 3-connected graph and G be a graph with an edge $e = xy$ not in G'. Then the following statements are equivalent.*

- *There is an BG-operation on G' that adds e and builds G.*
- *Removing e in G yields G' and either $|V(G)| = 4$ or in G holds*
  - $|N(x)| \geq 3$,
  - $|N(y)| \geq 3$ *and*

– $|N(x) \cup N(y)| \geq 5$.

Proof. If there is an BG-operation on G' that adds e, it clearly can be undone by removing e. G must be 3-connected with Lemma 4, thus $|N(x)| \geq 3$ and $|N(y)| \geq 3$ in G. Only one of $|V(G)| = 4$ and $|N(x) \cup N(y)| \geq 5$ can be true. If $|N(x)| > 3$ or $|N(y)| > 3$, $|N(x) \cup N(y)| \geq 5$ follows, since x and y are neighbors and no self-loops exist. Thus, let $|N(x)| = |N(y)| = 3$. Having $N(x) \setminus \{y\} \neq N(y) \setminus \{x\}$ yields $|N(x) \cup N(y)| \geq 5$ as well, so let $N(x) \setminus \{y\}$ and $N(y) \setminus \{x\}$ contain the same two nodes a and b. If $|V(G)| = 4$ does not hold, a or b is adjacent to a node c that is not adjacent to x and y. But then deleting a and b separates G into at least the connected component containing x and y and the component containing c, contradicting the 3-connectedness of G. Note that the case $|V(G)| = 4$ only occurs if e is a parallel edge.

Now let removing e yield G' and assume at first that $|V(G)| = 4$. Since G' is 3-connected and arises from G by removing e, G can only be the K_4 with e being a parallel edge. Then adding e equates to the BG-operation (a).

Let the neighborhood constraints in G be satisfied. Then both nodes x and y have degree at least 3 in G. Assume that neither x nor y have exactly degree 3. Then the removing operation does not smooth x and y and this again gives a BG-operation (a). If either x or y has degree 3, let w. l. o. g. x be that node. Then x becomes a node of degree 2 during the removal and is smoothed to form the edge f in G'. This edge cannot be a self-loop, since $|N(x)| \geq 3$ in G. The same argument shows that y cannot be an endnode of f. This ensures that e can be added by the BG-operation (b).

Finally, let both nodes x and y have degree 3 in G and let f (resp. g) be the edge in G' that is formed by smoothing x (resp. y). We show that then e is added by a BG-operation (c) that subdivides f and g. The constraints $|N(x)| \geq 3$ and $|N(y)| \geq 3$ prevent f and g from being identical and from being self-loops. Furthermore, f and g cannot be parallel, because $|N(x) \cup N(y)| \geq 5$. □

We phrased Lemma 11 quite general, but we know a bit more about G than it assumes, namely that G is 3-connected itself. In that case the neighborhood constraints follow directly from the 3-connectedness of G when $V(G) \neq 4$. Therefore, it suffices to check the graph G' on being 3-connected for every edge $e \in E(G) \setminus E(H)$. Testing a connected graph on 3-connectedness can be done in time $O(m)$ [4], so we need time $O(m^2)$ to find the right edge and $O(m^3)$ to find the whole construction sequence.

However, we can slightly improve the total running time to $O(n^3)$ with a $O(m)$ preprocessing [7] that preserves G to be 3-connected while reducing the number of edges to $O(n)$. Computing chains instead of BG-operations allows us to obtain better running times, but at first we need to know how exactly construction sequences can be represented.

4.1 Representations

An obvious representation of a construction sequence Q would be to store the graph $G_0 = \mathit{smooth}(H)$ and in addition every BG-operation, yielding the sequence $G_0, \ldots, G_z = G$. Unfortunately, the graphs G_i are not necessarily subgraphs of G_{i+1}, so we have to take care of relabeled edges when specifying each operation.

Whenever an edge e is subdivided as part of an operation (b) or (c), we specify it by its index in G_i followed by assigning new indices for the new degree-two node and one of the two new separated edge parts in G_{i+1}. The other edge part keeps the index of e.

Similarly, on operations (a) and (b), real endnodes of the added edge are specified by their indices in G_i. We assign a new index for the added edge in G_{i+1}, too. Finally, we have to impose the constraint that G_z is not just isomorphic but identical to G, meaning that nodes and edges of G_z and G are labeled by exactly the same indices, since otherwise we would have to solve the graph isomorphism problem to check that Q really constructs G.

On the other hand we know with Theorem 10.(2) that we can prescribe $H = S_0(Q)$ in G while preserving the construction sequence. That allows us to represent Q without indexing issues by storing just $S_0(Q)$ and the chains $C_0, \ldots, C_{z-1}$. Hence, we can represent a construction sequence Q of G in the following two ways.

- Represent Q by G_0 and a sequence of BG-operations, along with specifying new and old indices for each operation, such that G_z and G are labeled the same.
- Represent Q by S_0 and chains $C_0, \ldots, C_{z-1}$.

Both representations refer to the same sequence of graphs $G_0, \ldots, G_z$ and are of size $\theta(m)$, assuming the uniform cost model. We show that it does not matter which of the two representations we compute.

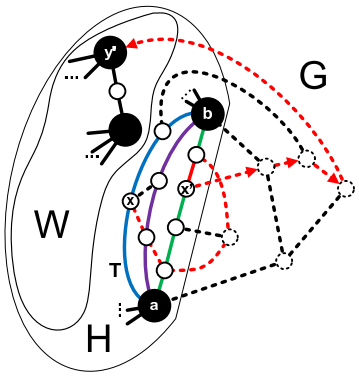


Figure 4: The case $H \neq \mathit{smooth}(H)$. Dashed edges are in $E(G)\backslash E(H)$, arrows depict the chain $x' \to y'$.

Lemma 12. *Both representations of a construction sequence Q can be transformed into each other in $O(m)$. Moreover, the representation computed is a unique representation of Q.*

Proof. Let G_0 and a sequence of BG-operations along with their specified indices on edges and nodes be given. If an operation O' subdivides an edge e', we define $\beta(e', O')$ to be the edge of the two new ones that gets a new index. Let e be the added edge of an operation in Q. With the preliminary considerations e corresponds to a chain C, which will therefore be subdivided $|C| - 1$ times in the construction sequence. To compute the chain C we have to keep track of the $|C| - 1$ operations that involve subdividing e and glue the parts together.

Whenever an operation $O \in Q$ subdivides e we store a pointer from $\beta(e, O)$ to e. Moreover, on all edges f that point to e and are subdivided we store a pointer from $\beta(f, O'')$ to e. In both cases, we put $\beta(e, O)$ resp. $\beta(f, O'')$ in a list stored on the edge e. Each $\beta(e, O)$ can be found in constant time and by augmenting the list of e with e itself we get all the edges in which C got subdivided in the end, hence exactly the set of edges in the chain C. Since G_z has the same labeling as G, the indices of e and all other edges in C are still contained in G.

The set of edges is not necessarily in the order of appearance in C, but this can be easily fixed in time $O(|C|)$ by temporarily storing the incidence information of this set on the endnodes and extracting the chain C from a degree-one node. In order to compute S_0, we analogously maintain pointers for each edge of G_0 to get the paths with endnodes having degree three in S_0. Those paths partition $E(S_0)$ and since $C_0, \ldots, C_{z-1}$ partition $E(G) \setminus E(S_0)$, the running time is $O(m)$.

Conversely, let S_0 and the sequence $C_0, \ldots, C_{z-1}$ of chains be given. We *remove* chains in reversed order from G by deleting their edge followed by smoothing their endnodes (there is only one edge left this way, the one added in the corresponding BG-operation). Therefore, we pass through the graph sequence $G_z, \ldots, G_0$ and get G_0. If both endnodes of the removed chain C_i are real before smoothing them, we can keep their index and construct the corresponding BG-operation (a).

Otherwise let a be an endnode having degree two before smoothing it, incident to the edges e and f. We smooth a, assign the lowest index of e and f to the new edge and construct the operation (b) or (c) with the involved indices in constant time. It remains to show that always unique representations of Q are computed. The representation with chains is by definition unique. The other representation can vary in the indices, but picking the incident edge with lowest index before smoothing a node creates a unique representation, since G is given and every encountered edge is in $E(G)$. □

4.2 A quadratic time algorithm

Let G and S_0 be given. We follow the lines of Theorem 9 and construct $C_0, \ldots, C_{z-1}$ in $O(n^2)$, which then can be transformed to a BG-operation sequence in $O(m)$. An index for every chain link is assigned and stored on each of its inner nodes. Moreover, a pointer to its two endnodes is saved for every chain link.

In case $H \neq \mathit{smooth}(H)$ of Theorem 9 we pick an arbitrary node x of degree two. Let $T = a \to b$ be the chain link that contains x and let W be the set of nodes $V(H) \setminus V(T)$ minus all nodes in parallel chain links of T (see Figure 4). We compute the path $P = x \to y'$ by temporarily deleting a and b and performing a Depth First Search (DFS) on x that stops on the first node $y' \in W$. We can check whether a node lies in a parallel chain link of T in constant time by comparing the endnodes of its containing chain link with a and b. Thus, the subpath $x' \to y'$ with x' being the last node in T or in a parallel chain link of T is a chain and can be found efficiently. The chain links can be updated in $O(n)$.

Similarly, in case $H = \mathit{smooth}(H)$ we start a DFS on a node $x \in V(H)$ that has incident edges in $E(G) \setminus E(H)$. This DFS-traversal only traverses edges in $E(G) \setminus E(H)$ and stops at the first node $y \in V(H) \setminus \{x\}$. The path $x \to y$ is then the desired chain. Applying the preprocessing of [7] in advance gives a running time of $O(n)$ for each chain, thus $O(n^2)$ in total. We can extend this algorithm to construct the basic construction sequence 10.(3) on simple graphs G with the following Lemma.

Lemma 13. *For simple graphs G, the construction sequences 10.(2) and 10.(3) can be transformed into each other in $O(m)$.*

Proof. It is straightforward to split the three internally node-disjoint paths of each expand operation into two chains, possibly inducing non-basic operations.

Conversely, let the construction sequence 10.(2) be given and represented with BG-operations, otherwise we can compute this representation in $O(m)$. For each added edge of a BG-operation its position in the construction sequence and, if exists, the BG-operation that subdivides it is stored. Performing a bucket sort on either endnode of all added edges gives a list of added edges sorted in lexicographic order, which can be used to group edges with the same endnodes.

For each set S of added edges having the same endnodes we apply the following procedure: If there is an edge in S that is not going to be subdivided and that does not have the first position of all edges in this set, we move it to the end of the construction sequence and remove it from S. The edge with the first position in S is already part of a basic operation. All other edges $e \in S$ can be non-basic, but are at some point subdivided by the added edge f of a BG-operation. We move $e = ab$ to the position of $f = cd$ without harming the construction sequence. If f subdivides only the edge e with c and no other edge, we can glue e and f together to an expand operation, which is basic due to the new node c.

Otherwise, f subdivides another edge g with d (see Figure 5) and e and f can be replaced with two BG-operations of type (b), namely by adding the edge da followed by adding the edge cb. Again, these operations are basic, since they involve subdividing edges and all steps can be carried out in $O(m)$. □

Theorem 14. *The construction sequences* 10.(2) *and* 10.(3) *can be computed in* $O(n^2)$.

5 A Certifying 3-Connectedness Test

We use construction sequences in the chain representation as a certificate for the 3-connectedness of graphs. This leads to a new, certifying method for testing graphs on being 3-connected. The total running time of this method is $O(n^2)$, however this is dominated by the time needed for finding the construction sequence and every improvement made there will automatically result in a faster algorithm for testing 3-connectedness. The test has the advantage that the input graph does not have to be biconnected nor connected.

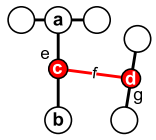


Figure 5: No expand operation can be formed.

For an arbitrary (multi-)graph, we follow the steps:

- Apply preprocessing of Nagamochi & Ibaraki and get G in $O(n + m)$ (this is only used to improve the running time)
- Try to compute a K_4-subdivision S_0 in G and prescribe it in $O(n)$
- Try to compute a construction sequence of G from S_0 in $O(n^2)$
  - Success: Return the construction sequence
  - Failure: Return a separation pair

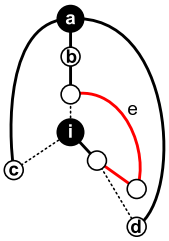


Figure 6: Finding a K_4-subdivision. Dashed edges can be (empty) paths, arcs depict backedges.

The preprocessing step preserves the graph to be 3-connected or to be not 3-connected. We first describe how to find a K_4-subdivision by one DFS-traversal, which as a byproduct can sort out graphs that are not connected or have nodes with less than 3 neighbors. Let a (resp. b) be the node in the DFS-tree T that is visited first (resp. second). If G is 3-connected, then a and b have exactly one child, otherwise they form a separation pair. We choose the two neighbors c and d of a that are visited last (see Figure 6). Traversing the paths $c \to b$ and $d \to b$ in T gives the least common ancestor $i \neq b$ of c and d.

Let j be the child of i that leads to d. If G is 3-connected, we can find a backedge e that starts on a node z in the subtree rooted at j and ends on an inner node z' of $a \to i$ in time $O(n)$. If e does not exist, a and i form a separation pair, otherwise we have found a K_4-subdivision with real nodes a, i, z and z'. The paths connecting this real nodes are determined by T and the three visited backedges.

Once we have found the K_4-subdivision, we try to compute the construction sequence and succeed if G is 3-connected. Otherwise no construction sequence can exist with Lemma 4. In that case a DFS-traversal starting at node x fails to find a new chain for some subdivision $H \subset G$. If $H \neq \mathit{smooth}(H)$, the endnodes of the chain link that contains x form a separation pair. Otherwise $H = \mathit{smooth}(H)$ and x must be a cut vertex. Thus, if G is not 3-connected, the algorithm returns always a separation pair or cut vertex.

5.1 Verifying the Construction Sequence

It is essential for a certificate that it can be easily validated. We could do this by transforming the chain representation to BG-operations using Lemma 12 and checking them on being valid by comparing indices, but there is a more direct way. First, it can be checked in linear time that all chains $C_0, \ldots, C_{z-1}$ are paths in G and that these paths partition $E(G) \setminus E(S_0)$. We try to remove the chains $C_{z-1}, \ldots, C_0$ from G in that order. If the certificate is valid, this is well defined as all removed chains are then edges. On the other hand we can detect longer chains $|C_i| \geq 2$ before their removal, in which case the certificate is not valid, since inner nodes of C_i are not attached to chains C_j, $j > i$.

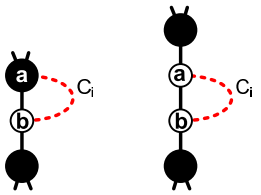
We verify that every removed $C_i = ab$ corresponds to a BG-operation by using the definition 7 of chains, and start with checking that a and b lie in our current subgraph for condition 7.1.

The conditions 7.2 and 7.3 can now be checked in constant time: Consider the situation immediately after the deletion of ab, but before smoothing a and b. Then almost all chain links in our subgraph are single edges, except possibly ones containing a and b as inner nodes.

Therefore, 7.2 is not met if and only if a is a neighbor of b and at least one of a and b has degree two (see Figures 7 for possible configurations). Condition 7.3 is not met if and only if $N(a) = N(b)$ and both a and b have degree two. Both conditions can be easily checked in constant time. Note that encountering proper chains $C_{z-1}, \ldots, C_i$ does not necessarily imply that the current subgraph is 3-connected, since false chains C_j, $j < i$, can exist.

It remains to validate that the graph after removing all chains is the K_4. This can done in constant time by checking it on being simple and having exactly 4 degree three nodes.

Theorem 15. *A construction sequence can be checked on validity in time linearly dependent on its length.*



(a) Either a or b has degree 2.

(b) Both a and b have degree 2.

Figure 7: Cases where 7.2 fails when $a \in N(b)$.

References

[1] D. Barnette and B. Grünbaum. On Steinitz's theorem concerning convex 3-polytopes and on some properties of 3-connected graphs. *Many Facets of Graph Theory, Lecture Notes in Mathematics*, 110:27–40, 1969.

[2] M. Blum and S. Kannan. Designing programs that check their work. In *STOC '89*, pages 86–97, New York, 1989.

[3] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is NP-complete. *Siam J. Comp.*, 5(4):704–714, 1976.

[4] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

[5] A. K. Kelmans. Graph expansion and reduction. *Algebraic methods in graph theory, Szeged, Hungary*, 1:317–343, 1978.

[6] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math.*, 10:96–115, 1927.

[7] H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(1-6):583–596, 1992.

[8] C. Thomassen. Reflections on graph theory. *Journal of Graph Theory*, 10(3):309–324, 2006.

[9] V. K. Titov. *A constructive description of some classes of graphs.* PhD thesis, Moscow, 1975.