Contractions, Removals and Certifying 3-Connectivity in Linear Time

Technical Report B 10-04

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

Abstract

As existence result, it is well known that every 3-connected graph $G = (V, E)$ on more than 4 vertices admits a sequence of contractions and a sequence of removal operations to K_4 such that every intermediate graph in the sequences is 3-connected. We show that both sequences can be computed in linear time, improving the previous best known running time of $O(|V|^2)$ to $O(|V| + |E|)$. This settles also the open question of finding a certifying 3-connectivity test in linear time and extents to certify 3-edge-connectivity in linear time as well.

1 Construction Sequences

Let $G = (V, E)$ be a finite graph with $n := |V|$ vertices and $m := |E|$ edges. Let G be *connected* if there is a path between every two vertices and *disconnected* otherwise. For $k \geq 1$, a graph G is *k-connected* if $n > k$ and deleting every set of $k-1$ vertices leaves a connected graph. A vertex (a pair of vertices) that leaves a disconnected graph upon deletion is called a *cut vertex* (a *separation pair*). Note that k-connectivity does neither depend on parallel edges nor on self-loops. A graph has *connectivity* k if it is k-connected but not $k+1$-connected. Let a path P from vertex v to vertex w in G be denoted by $v \to_G w$ and $s(P) = v$ and $t(P) = w$. For a vertex v in G, let $N(v) = \{w \mid vw \in E\}$ denote its set of neighbors and $deg(v)$ its degree. For a graph G, let $\delta(G)$ be the minimum degree of its vertices.

A *subdivision* of a graph G replaces each edge of G by a path of length at least one. Conversely, we want a notation to get back to the graph without subdivided edges. If $deg(v) = 2$, $|N(v)| = 2$ and $v \notin N(v)$, 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 one of the conditions is violated, we set $smooth_v(G) = G$. Let $smooth(G)$ be the graph obtained by smoothing every

*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).

vertex. For an edge $e \in E$, let $G \setminus e$ denote the graph obtained from G by deleting e.

Contracting an edge $e = xy$ in a graph deletes e, identifies vertices x and y and replaces iteratively two parallel edges by one single edge as long as possible. This way the simpleness of graphs is preserved by contractions. An edge e is called *contractible* if contracting e results in a 3-connected graph. Let K_n be the complete graph on n vertices and K_n^m be the complete graph on n vertices with m edges between each pair of vertices. For a rooted tree T and $x \in V(T)$, let $T(x)$ be the maximal subtree of T rooted at x.

For convenience, we assume the input graph G to be simple, although all results can be extended to multigraphs. The following operations are called *Barnette and Grünbaum operations* (*BG-operations*).

- add an edge xy (possibly a parallel edge)
- subdivide an edge ab by a vertex x and add the edge xy for a vertex $y \notin \{a, b\}$
- subdivide two non-parallel edges by vertices x and y, respectively, and add the edge xy

A sequence of BG-operations that starts with K_4 and ends with a graph G is called a *construction sequence* of G. Let $G_4, G_5, \ldots, G_z$ with $G_4 = K_4$ and $G_z = G$ be the graphs obtained in a construction sequence of G. As BG-operations preserve 3-connectivity and K_4 is 3-connected, every G_j, $4 \leq j \leq z$ is as well 3-connected. We can represent the construction sequence in a different, but equivalent way [1, 4]: Every G_j is identified with a G_j-subdivision S_j in G, giving the sequence $S_4, \ldots, S_z$ of subgraphs in G. In particular, S_4 is a K_4-subdivision in G. As with contractions, we want the inverse of a BG-operation. Let *removing* the edge $e = xy$ of a graph be the operation of deleting e followed by smoothing x and y. An edge $e = xy$ in G is called *removable*, if removing e yields a 3-connected graph.

The vertices v in S_j with $deg(v) \geq 3$ are called *real* vertices, because they correspond to vertices in G_j. Let $V_{real}(S_j)$ be the set of real vertices in S_j. Note that the graphs G_j may contain parallel edges, although every graph S_j is simple, because G is simple. We define the *links* of each S_j to be the unique paths in S_j with only their end vertices being real. The links of S_j partition $E(S_j)$, as S_j is simple, 2-connected and not a cycle and therefore two vertices of degree at least 3 must exist [6, Lemma 12.11]. Let two links be *parallel* if they share the same end vertices.

Definition 1. A *BG-path* for a subgraph $S_j \subset G$ is a path $P = x \to_G y$ with the following properties:

1. $S_j \cap P = \{x, y\}$
2. Every link of S_j that contains x and y, contains them as end vertices.
3. If x and y are inner vertices of links L_x and L_y of S_j, respectively, and $|V_{real}(S_j)| \geq 4$, then L_x and L_y are not parallel.

It is easy to see that every BG-path for a subdivision S_j corresponds to a BG-operation on G_j and vice versa. The choice of the K_4-subdivision S_4 is not

crucial [4]: At the expense of having additional parallel edges in intermediate graphs $G_j \subset G$, there exists a construction sequence to G from *each* prescribed K_4-subdivision in G.

Theorem 2. *The following statements are equivalent:*

$$
\begin{aligned}
& A \text{ simple graph } G \text{ is 3-connected} && (1)\\
\Leftrightarrow\ & \exists \text{ sequence of BG-operations from } K_4 \text{ to } G \text{ (see [1, 6])} && (2)\\
\Leftrightarrow\ & \exists \text{ sequence of BG-paths from each } K_4\text{-subdivision in } G \text{ to } G \text{ and} \\
& \delta(G) \geq 3 \text{ (see [4])} && (3)\\
\Leftrightarrow\ & \exists \text{ sequence of removals from } G \text{ to } K_4 \text{ on removable edges } e = xy \\
& \text{with } |N(x)| \geq 3,\ |N(y)| \geq 3 \text{ and } |N(x) \cup N(y)| \geq 5 \text{ (see [4])} && (4)\\
\Leftrightarrow\ & \exists \text{ sequence of contractions from } G \text{ to } K_4 \text{ on contractible edges } e = xy \\
& \text{with } |N(x)| \geq 3 \text{ and } |N(y)| \geq 3 \text{ (see [5])} && (5)
\end{aligned}
$$

Sequences 2.(2) and 2.(3) can be transformed into each other with a simple linear time algorithm [4]. Moreover, having sequence 2.(2) gives immediately sequence 2.(4) in linear time by removing the edges that were added as BG-operations in reverse order. The following result shows that sequences based on Barnette and Grünbaums' characterization are algorithmically at least as powerful as the well-known sequence of contractions and all other sequences of Theorem 2.

Lemma 3 ([4])**.** *There is a simple algorithm that transforms a given sequence of type* 2.(2) *or* 2.(3) *to a sequence of type* 2.(4) *and to a sequence of type* 2.(5) *in linear time.*

This allows us to focus only on computing a sequence of type 2.(3). To construct such a sequence, we will use the following Lemma for the special case that H is a K_4-subdivision.

Lemma 4 ([4])**.** *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 BG-path for H in G. Moreover, every link of H of length at least 2 contains an inner vertex on which a BG-path for H starts.*

2 Chain Decomposition

Let G be the simple, 3-connected input graph. According to Lemma 4, it suffices to add iteratively BG-paths to a K_4-subdivision in G to get sequence 2.(3). Note that we cannot make a wrong decision when choosing one BG-path in the construction sequence, since Lemma 4 can be applied on the new subdivision as well and therefore ensures a completion of the sequence. Instead of starting with a K_4-subdivision, we will w. l. o. g. start with a K_2^3-subdivision S_3 and demand that there is a BG-path for S_3 that results in a K_4-subdivision. We now construct S_3 in G and then describe a decomposition of G into special paths that allow us to find the BG-paths for each S_i, $3 \leq j \leq z$, efficiently.

An arbitrary Depth First Search (DFS) is performed on G, assigning a Depth First Index (DFI) to every vertex. Let T be the DFS-tree obtained, r be the

root of T and u be the vertex that is visited second in the DFS. Both, r and u, have exactly one child, as otherwise they would form a separation pair in G. For two vertices v and w in T, let v be a (*proper*) *ancestor* of w and w be a (*proper*) *descendant* of v if $v \in w \to_T r$ (and $v \neq w$). A *backedge* is an oriented edge $vw \in E(G) \setminus E(T)$ from v to w with v being an ancestor of w in T (note that the orientation differs from standard notation). A backedge vw is *entering* a subtree T' of a tree if $v \notin V(T')$ but $w \in V(T')$.

We choose two backedges ra and rb and denote the least common ancestor of a and b in T with x. The paths $x \to_T r$, $ra \cup a \to_T x$ and $rb \cup b \to_T x$ are the three subdivided edges of S_3 in G having real vertices r and x.

Dependent on T, we describe the decomposition of G into special paths $C := \{C_0, C_1, \ldots, C_{m-n+1}\}$, called *chains*, whose edge sets partition $E(G)$. Additionally, the decomposition yields a total order $<$ on C with $C_0 < C_1 < \ldots < C_{m-n+1}$ that reflects the order in which the chains were computed. Let $low(P)$ for a path P in G be the vertex in P with maximal DFI value. We set $C_0 := x \to_T r$, $C_1 := ra \cup a \to_T x$ and $C_2 := rb \cup b \to_T x$. The remaining chains are obtained by applying the following procedure subsequently for $i = 0, 1, \ldots, n-m+1$: We iterate over all vertices $v \in V(C_i)$ from $t(C_i)$ to $low(C_i)$. For every backedge vw that is not already in a chain, we traverse the path $w \to_T r$ until a vertex x is found that is contained in a chain. The traversed path including vw forms the new chain $C_j = v \to_G x$, with j being the next free index.

Note that every chain $C_i \neq C_0$ contains exactly one backedge and that $s(C_i)$ is always a proper ancestor of $t(C_i)$. For an edge $e \in E(G)$, let $C(e)$ be the chain that contains e. The next two lemmas show that C partitions the edge set of G and admits a tree structure.

Lemma 5. *The edge sets of the chains in C partition $E(G)$, hence $|C| = m - n + 2$. Let $e = e_1e_2$ be an edge in T and $f = f_1f_2$ be an edge in G with $e_2 = low(e)$, $f_2 = low(f)$ and e_2 being an ancestor of f_2 in T. Then $C(e) \leq C(f)$ holds.*

Proof. Whenever a vertex v is processed in the chain decomposition, all its ancestors and their attached backedges have already been processed. At the same time, for every child w of v, the tree edge vw is already contained in a chain, since otherwise all backedges entering $T(w)$ have to start at v, implying that v is a cut vertex separating $T(w)$. Therefore, w will be processed later. This implies that every vertex and every edge in G will be processed by the decomposition. Since all chains are pairwise edge-disjoint, the edge sets of chains in C partition $E(G)$. Since every chain in $C \setminus \{C_0\}$ contains exactly one backedge, C contains $m - n + 2$ chains.

The same line of argument settles the second claim for the case when e_1 is an ancestor of $s(C(f))$, since then e is already contained in a chain at the time $s(C(f))$ is processed and $C(e) < C(f)$ holds. The only remaining case is when $s(C(f))$ is an ancestor of e_1 with $s(C(f)) \neq e_1$. If e is not part of a chain when $C(f)$ is found, e will be contained in $C(f)$, because e_2 is an ancestor of f_2, and $C(e) = C(f)$. Otherwise, e is part of a chain that was found before $C(f)$ and $C(e) < C(f)$ holds. □

Definition 6. Let the *parent of a chain* $C_i \neq C_0$ be the chain that contains the edge from $t(C_i)$ to the parent of $t(C_i)$ in T.

Lemma 7. *The parent relation defines a tree* U *with* $V(U) = C$ *and root* C_0.

Proof. Let $D_0 \neq C_0$ be a chain in C and let $D_1, \ldots, D_k$ be the sequence of chains containing the edges of $t(D_0) \rightarrow_T r$ in that order, omitting double occurrences. By definition of the parent relation, each D_i, $0 \leq i < k$, has the parent D_{i+1} and $D_k = C_0$ holds. It follows that U is connected. Moreover, for each parent C_i of a chain C_j holds $C_i < C_j$ with Lemma 5 and, thus, no cycle can occur in U. □

2.1 Classifying Chains

In order to find BG-paths efficiently, we extend the chain decomposition by assigning each chain in $C \setminus \{C_0\}$ immediately after it was found to one of five types: 1, $2a + b$ and $3a + b$. These types form a partition of $C \setminus \{C_0\}$ and are defined by Algorithm 1. Note that the classification for C_i is dependent on the type of the parent C_k of C_i. At the beginning, all chains are unmarked.

Algorithm 1 classify($C_i \in C \setminus \{C_0\}$, DFS-tree T)

```
1:  C_k := parent(C_i)                                   ▷ the parent of C_i in U: C_k < C_i
2:  if t(C_i) →_T s(C_i) is contained in C_k then                                  ▷ type 1
3:      assign C_i to type 1
4:  else if s(C_i) = s(C_k) then   ▷ type 2: C_k ≠ C_0, t(C_i) is inner vertex of C_k
5:      if C_i is a backedge then
6:          assign C_i to type 2a                                                  ▷ type 2a
7:      else
8:          assign C_i to type 2b; mark C_i                                        ▷ type 2b
9:  else          ▷ type 3: s(C_i) ≠ s(C_k), C_k ≠ C_0, t(C_i) is inner vertex of C_k
10:     if C_k is not marked then
11:         assign C_i to type 3a                                                  ▷ type 3a
12:     else                                                                ▷ C_k is marked
13:         assign C_i to type 3b; create a list L_i = {C_i}; C_j := C_k          ▷ type 3b
14:         while C_j is marked do          ▷ the chains in L_i form a caterpillar
15:             unmark C_j; append C_j to L_i; C_j := parent(C_j)
```

Lemma 8. *Computing a chain decomposition of a* 3*-connected graph and classifying each chain with Algorithm 1 takes running time* $O(n + m)$.

Proof. The DFS tree T can be obtained in time $O(n + m)$. The subdivision S_3 can be found in time linearly dependent on $E(S_3)$ by taking two arbitrary backedges ra and rb with r being the root of T and finding the lowest common ancestor of a and b by traversing T upwards. The computation of each remaining chain C_i, $i > 2$, takes time linearly dependent on its length, too, which gives a running time of $O(n + m)$ for the chain decomposition.

In order to obtain a fast classification in Algorithm 1, we store the following information on each chain C_i: A pointer to its parent C_k (for $C_i \neq C_0$), pointers

to $s(C_i)$ and $t(C_i)$ and the information whether C_i is a backedge. In addition, for each inner vertex of C_i a pointer to C_i is stored. That allows us to check vertices on being contained as inner vertices or end vertices in arbitrary chains in $O(1)$. If $C_k = C_0$, we can check the condition on C_i being of type 1 in constant time by testing whether $s(C_i)$ and $t(C_i)$ are contained in C_0. If $C_k \neq C_0$, we check in constant time whether $s(C_i)$ and $t(C_i)$ are contained in $C_k \setminus \{s(C_k)\}$. The condition for type 2 needs constant time as well. Every chain is marked at most once, therefore unmarked as most once in line 15 of Algorithm 1, which gives a total running time of $O(n + m)$. □

2.2 Restrictions

Definition 9. Let a subdivision $S_j \subseteq G$ be *upwards-closed* if for each vertex in S_j the edge to its parent is in $E(S_j)$. Let S_j be *modular* if S_j is the union of chains.

In order to find BG-paths efficiently, we want to restrict every subdivision S_l to be upwards-closed and modular.

Lemma 10. *Let S_l be upwards-closed and modular. Then a BG-path P for S_l is a chain if and only if S_{l+1} is upwards-closed and modular.*

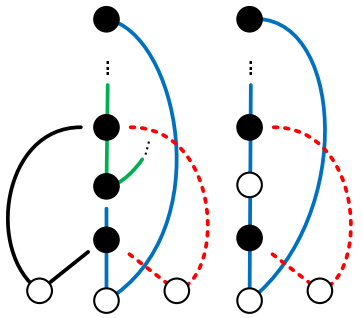
Proof. If P is a chain, $t(P)$ is contained in S_l and S_{l+1} must be upwards-closed and modular due to the DFS structure. If P is not a chain, we assume to the contrary that S_{l+1} is upwards-closed and modular. Then P must be the union of $t > 1$ chains; let C_i be the first chain in P. Now P cannot start with $t(C_i)$, since $s(C_i)$ is in S_l and property 1.1 contradicts $t > 1$. Thus, P starts with $s(C_i)$, which contradicts $t > 1$ as well, as S_{l+1} is upwards-closed and a second chain in P would include another backedge in P at a vertex that is already incident to two DFS tree edges. □

Lemma 10 shows that this restriction implies every BG-path to be a chain. Unfortunately, configurations exist where no BG-path for a subdivision S_l is a chain (show an example here) and we have to weaken the restriction in order to ensure the existence of a construction sequence. Instead of forcing every subdivision to be upwards-closed and modular, we will only assume that the current subdivision S_l has that properties and then find t BG-paths that result in an upwards-closed subdivision when applied successively and whose union is the union of t distinct chains. This ensures S_{l+t} to be modular and implies $z = |C| = m-n+2$. Since S_3 is upwards-closed and modular, we now can assume that whenever we are searching for a new BG-path, the current subdivision S_l is upwards-closed, modular and consists of exactly l chains (and at least l links).

We impose the additional restriction (R_2) (see Figure 1), which will prevent some BG-paths in the sequence from violating property 1.3. In total, we obey the following restrictions in a construction sequence:

(R_1) For each subdivision S_l, BG-paths are only added as

- single chains of type 1, $2a$ or $3a$, respectively, with S_{l+1} being upwards-closed and modular or as



(a) allowed (b) forbidden

Figure 1: The effect of restriction (R_2) on the dashed BG-path.

- set of $t > 1$ successive BG-paths that construct an upwards-closed modular subdivision S_{l+t} differing from S_l in exactly t chains of types $2b$ and $3b$.

(R_2) For each subdivision S_l, every link of S_l that contains only tree edges of T has no parallel link, except C_0 in S_3.

We will prove the existence of a construction sequence restricted by (R_1) and (R_2) in Section 3. Now it is shown that restriction (R_2) implies property 1.3 on upwards-closed modular subdivisions.

Lemma 11. *Let S_l be a subdivision constructed under (R_1) and (R_2) (in particular, S_l is upwards-closed and modular). Then each path P for S_l having properties 1.1 and 1.2 is a BG-path. If P is additionally a chain of type $2a$ or $3a$, (R_1) and (R_2) are preserved.*

Proof. For the first claim, assume to the contrary that P violates property 1.3. Then $|V_{real}(S_l)| \geq 4$ must hold and $S_l \neq S_3$ follows. Let R and Q be the parallel links of S_l that contain the end vertices of P as inner vertices, respectively. At least one of them, say R, contains a backedge, since otherwise T would contain a cycle. Let $C_i \neq C_0$ be the chain in S_l that contains R. Since C_i contains exactly one backedge, $s(C_i)$ is an end vertex of R. If $R \subset C_i$, Q must contain a backedge, as $t(C_i)$ is an inner real vertex of $t(R) \to_T s(R)$. In that case, all inner vertices of Q lie in a subtree of T that cannot be reached by P due to property 1.1 and S_l being upwards-closed. Thus, $R = C_i$ and with the same argument $Q = t(C_i) \to_T s(C_i)$ holds. With (R_2), S_l must be S_3 and $Q = C_0$, which contradicts our assumption.

For the second claim, each chain C_i of type 2 or 3 has by definition an inner real vertex in $t(C_i) \to_T s(C_i)$ and therefore preserves (R_2). If C_i is of type $2a$ or $3a$, (R_1) is preserved as well, as S_{l+1} is upwards-closed and modular with Lemma 10. □

We now show that the chains of type $3a$ help to find BG-paths efficiently.

Lemma 12. *Let C_k be the parent of a chain $C_i \neq C_0$.*

- *If C_i is not of type 1, $C_k \neq C_0$ and $t(C_i)$ is an inner vertex of C_k.*

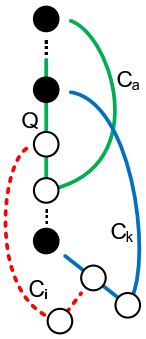


Figure 2: A chain $C_i \not\subseteq S_l$ of type 3.

- *Let C_k but not C_i be contained in a subdivision S_l constructed under (R_1) and (R_2). If C_i is either of type 3a or of type 1 with an inner real vertex in $t(C_i) \to_T s(C_i)$, C_i is a BG-path for S_l preserving (R_1) and (R_2).*

Proof. Assume to the contrary that C_i is not of type 1 and $C_k = C_0$. Because $t(C_i)$ is contained in C_0, $s(C_i)$ must be in C_0 as well. But then C_i would be of type 1, since $t(C_i) \to_T s(C_i) \subseteq C_0$. Therefore, if C_i is not of type 1, $C_k \neq C_0$ holds and C_k must start with a backedge. Then the definition of the parent relation implies that $t(C_i)$ is an inner vertex of C_k.

For the second claim, let C_i first be of type 3*a*. Since S_l is upwards-closed, modular and contains C_k, C_i satisfies the property 1.1 of BG-paths. In addition, $s(C_i) \neq s(C_k)$ holds by definition and with $C_k < C_i$, $s(C_i)$ must be an inner vertex of the path $t(C_k) \to_T s(C_k)$ (see Figure 2). Therefore, the only chains C_j that contain $s(C_i)$ and $t(C_i)$ are different from C_0 and fulfill $C_i \cap C_j = \{s(C_i), t(C_i)\} = \{s(C_j), t(C_j)\}$. This implies C_i having property 1.2. Using Lemma 11, C_i is a BG-path for S_l that preserves (R_1) and (R_2).

If C_i is of type 1, property 1.1 follows from the same argument as before. Additionally, the inner real vertex in $t(C_i) \to_T s(C_i)$ prevents any link containing $s(C_i)$ and $t(C_i)$ from having $s(C_i)$ or $t(C_i)$ as an inner vertex and therefore ensures property 1.2. Lemma 11 implies that C_i is a BG-path for S_l and C_i must preserve (R_1) and (R_2), the latter due to the inner real vertex in $t(C_i) \to_T s(C_i)$. □

2.3 Caterpillars

While chains of type 3*a* form BG-operations under the conditions of Lemma 12, chains of types 1 and 2 in general do not. For every chain C_i of type 3*b*, Algorithm 1 collects a list L_i that contains only C_i and chains of type 2*b* (see line 15). We call each list L_i a *caterpillar*. Caterpillars bundle the single chains of type 2*b*, which cannot immediately be added as BG-paths, and will offer a simple decomposition into successive BG-paths later.

Proposition 13. *Every caterpillar L_i contains exactly one chain of type 3b, namely the chain C_i. All other chains in L_i are of type 2b.*

Lemma 14. *$C \setminus \{C_0\}$ is partitioned into the chains of types 1, 2a, 3a and the chains being contained in caterpillars. Moreover, no chain is contained in two caterpillars.*

Proof. With Proposition 13, it remains to show that every chain C_i of type $2b$ or $3b$ is contained in exactly one caterpillar. If C_i is of type $3b$, C_i is part of the caterpillar L_i (see Algorithm 1, line 13) and will not be assigned to a second caterpillar afterwards, as it is not marked. Otherwise, C_i is of type $2b$ and was therefore marked. We show that, after all chains in C have been classified, C_i is not marked anymore. This forces C_i to be contained in exactly one caterpillar, as the only way to unmark chains is to assign them to a caterpillar (see Algorithm 1, line 15) and no chain is marked twice.

Let C_k be the parent of C_i. Because C_i is of type $2b$, $s(C_i) = s(C_k)$ holds and C_i is not a backedge, implying that the last edge e of C_i is in T. Let x be the end vertex of e different from $t(C_i)$. Using Lemma 12, $C_k \neq C_0$ holds and $t(C_i)$ is an inner vertex of C_k. Then at least one backedge vw with $v \notin \{s(C_i), t(C_i)\}$ must enter $T(x)$, since otherwise $s(C_i)$ and $t(C_i)$ would be a separation pair of G. Let C_j be the minimal chain with respect to $<$ that contains such a backedge.

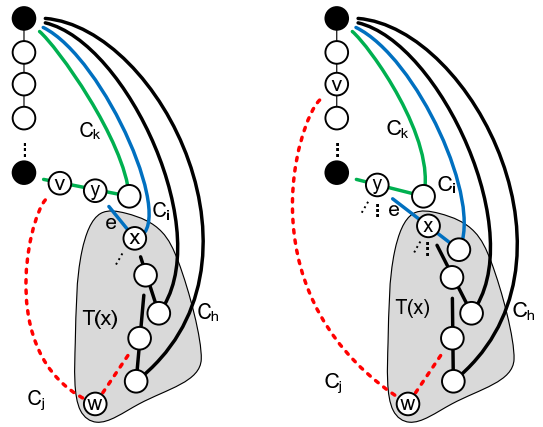
As $C_j > C_i$ holds due to Lemma 5, the vertex v is an inner vertex of $t(C_i) \rightarrow_T s(C_i)$, implying that C_j is not of type 2. In addition, C_j is not of type 1, since $t(C_j) \rightarrow_T v$ contains edges from C_i and C_k. At the time C_j is found in the chain decomposition, every chain that already ends at a vertex in $T(x)$ starts at $s(C_i)$ and is therefore of type $2a$ or $2b$. Since chains that are backedges cannot have children, the parent of C_j is marked and C_j is of type $3b$. Moreover, every chain corresponding to an inner vertex of the path $C_j \rightarrow_U C_i$ is marked. This concludes C_i to become unmarked due to line 15 of Algorithm 1. □

Definition 15. Let the *parent of a caterpillar* L_j be the parent of the chain in L_j that is minimal with respect to $<$. Let a caterpillar L_j with parent C_k be *bad* if $s(C_j)$ is a descendant of $t(C_k)$ and $s(C_k) \rightarrow_{C_k} s(C_j)$ contains no inner real vertex (see Figure 3(a)). Otherwise, L_j is called a *good* caterpillar (see Figure 3(b)).

We now show that, under some minor conditions, caterpillars can be decomposed into multiple BG-paths nicely.

Lemma 16. *Let L_j be a caterpillar that contains t chains and has parent C_k. Let C_k but no chain in L_j be contained in a subdivision S_l that was constructed under (R_1) and (R_2). Then the chains in L_j can be efficiently decomposed into t successive BG-paths preserving (R_1) and creating subdivisions $S_{l+1}, S_{l+2}, \ldots, S_{l+t}$, each of which satisfies (R_2), if and only if L_j is good.*

Proof. Let L_j be good and let y be the last vertex of the minimal chain in L_j, thus $y \in V(C_k)$. We assume at first that $s(C_j)$ is a proper ancestor of $t(C_k)$ (see Figure 3(b)). Then the path $P = C_j \cup (t(C_j) \rightarrow_T y)$ fulfills properties 1.1 and 1.2 and is a BG-path for S_l with Lemma 11. Adding P preserves S_l to be upwards-closed but not modular. The restriction (R_2) is also preserved, as $t(C_k)$ is real and, for $S_l = S_3$, C_k must be either C_1 or C_2, implying that $s(P)$ becomes



(a) A *bad* caterpillar L_j with parent C_k.

(b) A *good* caterpillar L_j with parent C_k.

Figure 3: Two kinds of caterpillars.

an inner real vertex of C_0. Successively, for each chain C_i of the $t-1$ chains in $L_i \setminus \{C_j\}$, we now add $C_i \setminus P$, which is a BG-path yielding an upwards-closed subdivision for analogue reasons.

Now assume that $s(C_j)$ is a descendant of $t(C_k)$ (see Figure 3(a)). Then $s(C_j) \in V(C_k)$ and since L_j is good, there is a real vertex a strictly between $s(C_j)$ and $s(C_k)$ in C_k. We first show that $t(C_k) \to_T s(C_k)$ contains an inner real vertex as well. Assume the contrary. Then C_k must be of type 1 and has been added before, contradicting restriction (R_2) unless $S_l = S_3$. But S_l must be different from S_3, since a exists, and it follows that $t(C_k) \to_T s(C_k)$ contains an inner real vertex b. Let C_h be the parent of C_j. Then $(C_j \cup C_h) \setminus ((t(C_j) \to_T y) \setminus \{t(C_j)\})$ is a BG-path due to the real vertices a and b and we add it, although it neither preserves S_{l+1} to be upwards-closed nor modular. We next add $t(C_j) \to_T y$, which restores upwards-closedness. The resulting subdivisions S_{l+1} and S_{l+2} both satisfy (R_2), as b is real in S_{l+1} and S_{l+2} and y is real in S_{l+2}. We proceed with adding successively paths, namely for each chain C_i of the $t-2$ remaining chains in $L_i \setminus \{C_j, C_h\}$ the path $C_i \setminus (t(C_j) \to_T y)$. With the same line of argument, these paths obtain upwards-closed subdivisions $S_{l+3}, \ldots, S_{l+t}$, each of which satisfies (R_2).

In both cases, S_{l+t} is modular, since L_j is a list of chains. Moreover, the t chosen BG-paths preserve (R_1), as the chains in L_j are of types $2b$ and $3b$ only, $t > 1$ holds and S_{l+t} is upwards-closed. All paths can be computed in time linearly dependent on the total number of edges in L_j.

For the only if part, let P_1 and P_2 be the first two BG-paths in a decomposition of the chains in L_j; these exist, since $t > 1$ holds in every caterpillar. Let L_j be bad, as otherwise the claim follows. Then $s(C_j) \in V(C_k)$. We show that L_j cannot be bad, as S_l contains a real vertex in C_k strictly between $s(C_j)$ and $s(C_k)$. Because of properties 1.1 and 1.2, $P_1 \cap S_l = \{s(C_k), s(C_j)\}$ must hold and P_1 is a link of S_{l+1} being parallel to $s(C_j) \to_{C_k} s(C_k)$. Since only the chain of type $3b$ in L_j starts at $s(C_j)$, both end vertices of P_2 must be different from $s(C_j)$. Then, due to properties 1.1 and 1.2, P_2 joins inner vertices of the parallel links P_1 and $s(C_j) \to_{C_k} s(C_k)$ in S_{l+1}, contradicting property 1.3, as

$|V_{real}(S_{l+1})| \geq 4$. □

3 Existence of the Restricted Construction Sequence

We show that there still exists a construction sequence under restrictions (R_1) and (R_2).

Definition 17. Let S_l be a subdivision of G constructed under (R_1) and (R_2). We define the equivalence relation $\sim$ on $E(G) \setminus E(S_l)$ with

- $\forall e \in E(G) \setminus E(S_l) : e \sim e$ and
- $\forall e, f \in E(G) \setminus E(S_l) : e \sim f$ if there is a path $e \to_G f$ without an inner vertex in S_l.

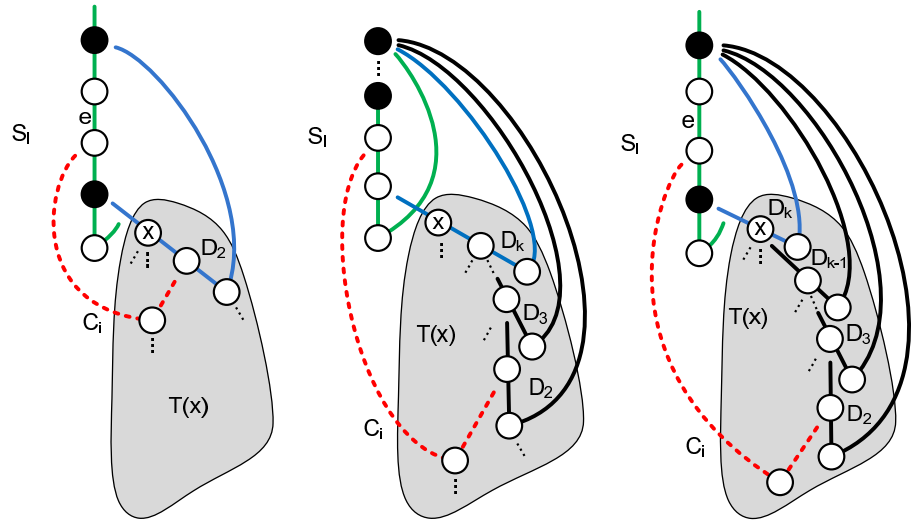
Let the equivalence classes of $\sim$ be the *segments* of S_l. Note that every segment consists of a disjoint union of chains. For a chain C_i that is not contained in S_l, let the *segment* of C_i be the segment of S_l that contains C_i.

Lemma 18. *Let S_l be a subdivision constructed under (R_1) and (R_2) and let C_i be a chain of type 3 such that $s(C_i) \in V(S_l)$, $C_i \not\subseteq S_l$ and C_i is minimal among the chains of type 3 in its segment H. Let $D_1 > \ldots > D_k$ be all ancestors of C_i in H with $D_1 = C_i$ and D_k being the minimal chain in H. Then all chains $D_1, \ldots, D_k$ can be successively added as BG-paths preserving (R_1) and (R_2) (possibly being part of caterpillars), unless one of the following exceptions holds:*

1. *C_i is of type 3a, $k = 2$, D_k is of type 1, $s(C_i)$ is an inner vertex of $t(D_k) \to_T s(D_k)$ and there is no inner real vertex in $t(D_k) \to_T s(D_k)$ (Figure 4(a)),*
2. *C_i is of type 3b, D_k is of type 2b and $L_i = \{D_1, \ldots, D_k\}$ with L_i being bad (Figure 4(b)),*
3. *C_i is of type 3b, $L_i = \{D_1, \ldots, D_{k-1}\}$, D_k is of type 1, $s(C_i)$ is an inner vertex of $t(D_k) \to_T s(D_k)$ and there is no inner real vertex in $t(D_k) \to_T s(D_k)$ (Figure 4(c)).*

Proof. Let $D \in \{D_2, \ldots, D_k\}$. Then D is not of type 3 by assumption and not of type 2a, as chains of that type cannot have children. Assume that D is of type 2b and let L_j be the caterpillar containing D due to Lemma 14. If $C_j \neq C_i$, $C_j < C_i$ holds, as otherwise C_j would not be the chain of type 3b in L_j. But then C_j contradicts the minimality of C_i, since C_j is not contained in S_l and of type 3b. We conclude that every chain in $\{D_2, \ldots, D_k\}$ of type 2b is contained in L_i and forces C_i to be of type 3b. This is used in the following case distinction.

Let C_i be of type 3a. If $k = 1$, C_i is a BG-path for S_l with Lemma 12 and the claim follows. Otherwise, $k > 1$ and all chains in $\{D_2, \ldots, D_k\}$ are of type 1. Then $s(D_2)$ is a proper ancestor of $s(C_i)$, since $D_2 < C_i$ and C_i is not of type 2. Moreover, $s(C_i)$ is a proper ancestor of $t(D_2)$, because otherwise



(a) 18.1 (b) 18.2 (c) 18.3

Figure 4: The three exceptions of Lemma 18. The black vertices in 18.1 and 18.3 can also be non-real.

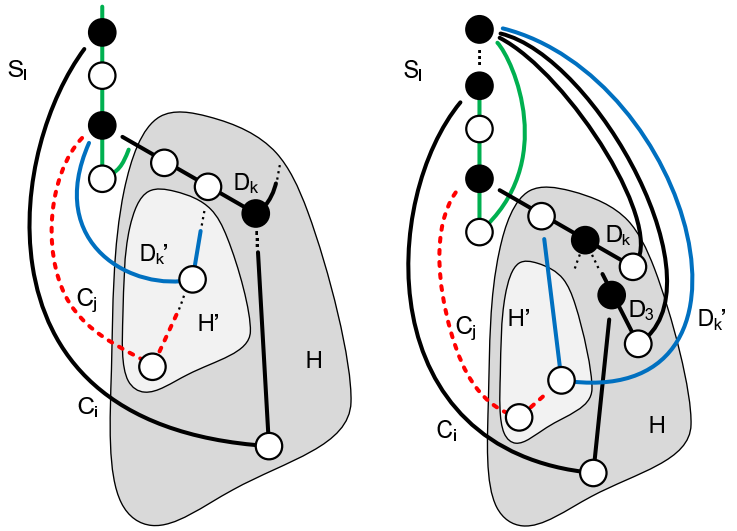
$H \cap S_l = \{s(D_2), t(D_2)\}$ is a separation pair of G due to the minimality of C_i. It follows that $s(C_i)$ is an inner vertex of $t(D_2) \to_T s(D_2)$. If $k > 2$, D_3 must contain $t(D_2) \to_T s(D_2)$, because D_2 is of type 1 and a child of D_3. Therefore, the edge e joining $s(C_i)$ with the parent of $s(C_i)$ in T is contained in D_3. But since S_l is upwards-closed, e is also contained in S_l, contradicting that $D_3 \not\subseteq S_l$. Thus, $k = 2$. If $t(D_2) \to_T s(D_2)$ contains an inner real vertex, D_2 and C_i can be subsequently added as BG-paths with Lemma 12, otherwise 18.1 is satisfied.

Let C_i be of type $3b$. Then all chains in $\{D_2, \dots, D_k\}$ that are of type $2b$ must be contained in L_i. Since every caterpillar L_j contains the parent of the chain C_j and since S_l contains no chain in L_i due to (R_1), $k > 1$ holds and D_2 is of type $2b$ with $D_2 \in L_i$. Let D_t with $1 < t \leq k$ be the minimal chain in L_i. If $t = k$ and L_i is good, all chains in L_i can be decomposed to BG-paths according to Lemma 16. If $t = k$ and L_i is bad, 18.2 is satisfied. Only the case $k > t$ remains. Then D_{t+1} is of type 1 and, using the same arguments as in the case for type $3a$, $s(C_i)$ is an inner vertex of $t(D_k) \to_T s(D_k)$ and $k = t + 1$. If $t(D_k) \to_T s(D_k)$ contains an inner real vertex, Lemmas 12 and 16 imply that D_k and L_i can be iteratively added as set of successive BG-paths, preserving (R_1) and (R_2). Otherwise, 18.3 is satisfied. □

We extend Lemma 18 to non-minimal chains of type 3.

Lemma 19. *Let the preconditions of Lemma 18 hold. If C_i is not contained in one of the exceptions 18.1-18.3 (as C_i), the chains of type 3 in H that start in S_l and their ancestors in H can be successively added as BG-paths, preserving (R_1) and (R_2).*

Proof. Using Lemma 18, we add the chains $C_i, D_2, \dots, D_k$ in H as BG-paths. This partitions H into new segments; let $H' \subseteq H \setminus \{C_i, D_2, \dots, D_k\}$ be such a



(a) C_j cannot be contained in exceptions 18.1 and 18.3.

(b) C_j cannot be contained in exception 18.2.

Figure 5: After C_i was added, the next minimal chain C_j is in no exception.

new segment. If H' does not contain chains of type 3 that start in S_l, the claim follows for such chains in H'. Otherwise, let C_j be the minimal chain of type 3 in H' that starts in S_l and let $C_j > D'_2 > \ldots > D'_k$ be its ancestors in H'. We show that C_j is not contained in one of the exceptions 18.1-18.3 and can therefore be added as BG-path with Lemma 18, along with its proper ancestors in H'. First, assume to the contrary that C_j is contained in exception 18.1 or 18.3 (see Figure 5(a)). Because D'_k is a proper descendant of D_k and D'_k is of type 1, $s(C_j) \in V(S_l)$ cannot be an inner vertex of $t(D'_k) \rightarrow_T s(D'_k)$, contradicting the assumption. Now assume to the contrary that C_j is contained in exception 18.2 (see Figure 5(b)). Then C_j is of type $3b$ and part of a bad caterpillar L_j, whose parent D is not contained in H'. Because L_j contains only chains in H', D must be a descendant of D_k and is therefore contained in $H \setminus H'$. Since L_j is bad, $s(C_j)$ is contained in $S_l \cap D$ and it follows with $s(C_j) \neq s(D)$ that D must end in S_l at the vertex $s(C_j)$. As D_k is the only chain in H that ends in S_l, $D = D_k$ must hold. But this contradicts L_j being bad, as D contains the inner real vertex $s(D_{k-1})$. Thus, C_j and its ancestors in H' can be added, partitioning H' into smaller segments. Iterating the same argument for these segments establishes the claim for all chains of type 3 in H that start in S_l. □

Each of the exceptions 18.1-18.3 in Lemma 18 contains a certain path without inner real vertices. We refer to this path in the following way.

Definition 20. Let a chain C_i that is of type 1 or $2a$ and has parent C_k be *dependent* on the path $s(C_i) \rightarrow_{C_k} t(C_i)$. Let every chain C_i of type $2b$ or $3b$ that is contained in a caterpillar L_i that parent C_k be *dependent* on the path $s(C_i) \rightarrow_T t(C_k) \cup t(C_k) \rightarrow_{C_k} s(C_k)$ (chains of type $3a$ are not dependent on anything).

The following lemma shows that the only chains of type 1 that cannot be added are either backedges or are contained as D_k in exceptions 18.1-18.3.

Lemma 21. *Let S_l be a subdivision constructed under (R_1) and (R_2), let C_j be a chain in S_l and let D_k be a child of C_j that is of type 1 and not in S_l. If D_k is not a backedge, there is a chain of type 3 in the segment containing D_k that starts in $t(D_k) \to_T s(D_k) \subset C_j$. If D_k is neither a backedge nor contained (as D_k) in the exceptions 18.1 and 18.3, D_k can be added as BG-path.*

Proof. Let H be the segment of D_k and assume that D_k is not a backedge. We first show that H contains a chain of type 3 that starts in $t(D_k) \to_T s(D_k)$. We can assume that D_k is not contained in the exceptions 18.1 and 18.3, as then H would contain such a chain by definition. Let x be the last but one vertex in D_k. Since G is 3-connected, there is a minimal chain C_i entering $T(x)$ such that $s(C_i)$ is an inner vertex of $t(D_k) \to_T s(D_k)$, as otherwise the inner vertices of D_k would be separated by $\{s(D_k), t(D_k)\}$. By definition of the chain decomposition, C_i must be of type $3a$ or $3b$. Because D_k is not contained in exceptions 18.1 and 18.3 and H cannot contain exception 18.2, Lemma 18 can be applied on C_i, obtaining the last claim. □

The next lemma ensures for every subdivision S_l the existence of a chain or caterpillar that obtains a BG-path for S_l, preserving (R_1) and (R_2).

Theorem 22. *Let S_l be constructed under (R_1) and (R_2) and let C_i be a chain such that, for every proper ancestor C_j of C_i, every child of C_j and every chain of type 3 starting in $V(C_j)$ is contained in S_l. Let X' be the set of children of C_i that are not contained in S_l and let Y' be the set of chains of type 3 that start in C_i and are not contained in S_l. Then the chains in $X' \dot\cup Y'$ can be successively added as BG-paths (possibly being part of caterpillars) such that (R_1) and (R_2) is preserved and every chain in Y' is followed by its proper ancestors in its segment.*

Proof. By assumption, C_i is contained in S_l. Let $D \not\subseteq S_l$ be a child of C_i. If $C_i = C_0$, D must be of type 1. Let $C_i \neq C_0$. Then D cannot be of type $3b$, as otherwise it would be contained in S_l due to (R_1) and $C_i \subset S_l$. It is neither of type $3a$, since in that case $s(D)$ is contained in a proper ancestor of C_i, implying $D \subset S_l$ by assumption. We conclude that D is of type 1 or 2 and focus on the cases where D can not be added. Let P be the path on which D depends on. If D is of type 1, P does not contain an inner real vertex, as otherwise D can be added as BG-path due to Lemma 12. With Lemma 21, D must be either a backedge or be contained as the minimal chain in exception 18.1 or 18.3. If D is of type $2a$, $s(C_i)$ is real and neither $t(D)$ nor an inner vertex in P can be real, since otherwise D can be added as BG-path, preserving (R_1) and (R_2). If D is of type $2b$, D is the minimal chain of a caterpillar L_a with parent C_i. According to Lemma 16, L_a is bad and, thus, corresponds to exception 18.2. The following is a list of the possible cases for which a child D of C_i is not added.

1. D is of type 1 without an inner real vertex in P and either a backedge or the minimal chain in exception 18.1 or 18.3
2. $C_i \neq C_0$ and D is of type $2a$ without a real vertex in $P \setminus \{s(D)\}$

3. $C_i \neq C_0$ and D is of type $2b$ without an inner real vertex in P (D is the minimal chain in exception 18.2)

We iteratively add all chains D in $X' \dot\cup Y'$ that do not satisfy one of the above three cases 22.1-22.3 for $D \in X'$ and whose segments do not contain one of the exceptions 18.1-18.3 for $D \in Y'$ (the latter followed by adding the proper ancestors in the segment of D according to Lemma 19). Let X be the set of remaining chains in X' and let Y be the set of remaining chains in Y'. If $X = \emptyset$, $Y = \emptyset$ holds as well, as otherwise the minimal chain in the segment containing one of the exceptions 18.1-18.3 is a child of C_i, contradicting $X = \emptyset$. This implies the claim for $X = \emptyset$.

We prove the theorem by showing that $X = \emptyset$ must hold. Assume to the contrary that $X \neq \emptyset$ and let S_t be the current subdivision (all segments will be dependent on S_t). Then C_i must contain a link L of length at least two, because the dependent path P in each of the cases 22.1-22.3 is in C_i and contains a non-real vertex due to the 3-connectivity and simpleness of G. According to Lemma 4, L contains an inner vertex v on which a BG-path B starts (not necessarily being a chain and not necessarily preserving (R_1) or (R_2)). Let e be the first edge of B. Then e is not contained in the segment of any $x \in X$, as otherwise B would not have property 1.2, because v is non-real and all start vertices of the chains in the segment of x that are in S_t are contained in L. Thus, $C(e)$ cannot be a child of C_i and it follows that $s(C(e)) = v$. In particular, $C(e)$ is not of type 1.

The segment of e cannot contain a chain of type 3 that starts in C_i, as it otherwise contains a chain $x \in X$ of type 1 or $2b$ due to exceptions 18.1-18.3, contradicting the previous argument. In particular, $C(e)$ is not of type 3 and the only remaining case is that $C(e)$ is of type 2.

Let C_k be the maximal ancestor of $C(e)$ that is not of type 2. Then $s(C_k) = v$ holds by construction of the chain decomposition and C_k must be contained in the segment of $C(e)$ due to (R_1), (R_2) and v being non-real. Since the segment of e cannot contain a chain of type 3 that starts in C_i, C_k must be of type 1. But then, as v is an inner vertex of L, C_k must be a child of C_i, contradicting that e is not contained in the segment of any $x \in X$. This is a contradiction to the existence of B and it follows that $X = \emptyset$, which implies the claim. □

Corollary 23. Let G be a 3-connected graph with a chain decomposition $C = \{C_0, \ldots, C_{m-n+1}\}$. Then $C_3, \ldots, C_{m-n+1}$ can be added to $S_3 = C_0 \cup C_1 \cup C_2$ such that every chain satisfies (R_1) and (R_2).

Remark. From Theorem 22, one can possibly derive the characterization of 3-connected graphs of Vo in [7, 8].

4 A Linear-Time Algorithm

With Lemma 8, a chain decomposition $C_0, \ldots, C_{m-n+1}$ can be computed by essentially one DFS-traversal in $O(n+m)$. According to Theorem 22, we can find a construction sequence that satisfies (R_1) and (R_2) by the following method: Iteratively for each chain C_i, $3 \leq i \leq m-n$, we add the children of C_i and the chains of type 3 that start in C_i as BG-paths or, if of type $3b$, as part of

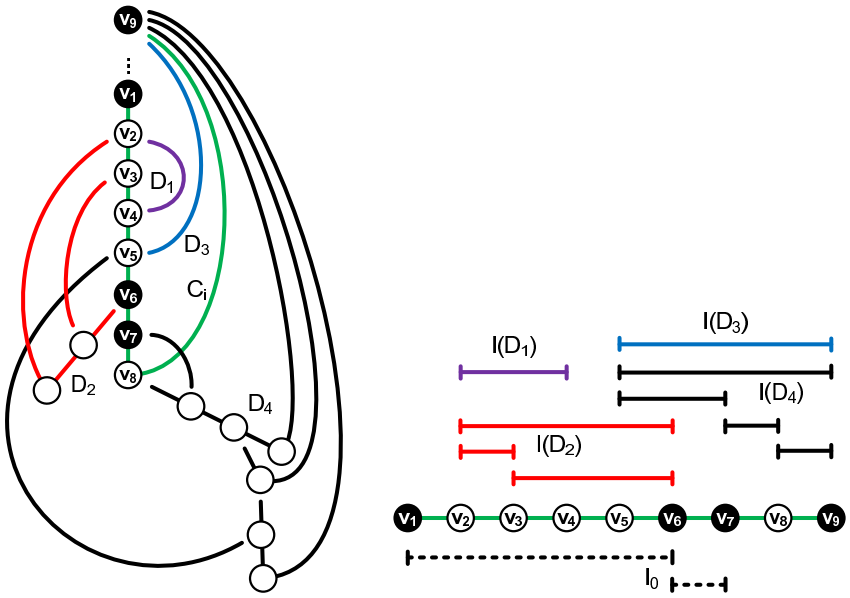
caterpillars, which can be decomposed into BG-paths. This obtains the desired construction sequence of type 2.(3), which can be transformed easily to the other sequences 2.(2), 2.(4) and 2.(5). However, Theorem 22 does not specify in which order the children of C_i and the chains of type 3 that start in C_i have to be added. Moreover, this order depends strongly on the input graph. We show how the computation of this order can be integrated in the chain decomposition while preserving the overall running time of $O(n+m)$.

We maintain the following information during the chain decomposition: On each vertex $v \notin \{s(C_0), t(C_0)\}$, we store a pointer to the unique chain that contains v as an inner vertex. On each chain C_j, we store pointers to $s(C_j)$ and $t(C_j)$, a list $\mathit{Children}_{12}(C_j)$ of the children of C_j that are of type 1 or 2 and the information whether C_j was already added. For each chain $C_i \neq C_0$, we save a list $\mathit{Type}_3(C_i)$ of pointers to the chains of type 3 that start in $C_i \setminus \{s(C_i), t(C_i)\}$. For C_0, we save the list $\mathit{Type}_3(C_0)$ of pointers to the chains of type 3 that start in C_0. For each caterpillar L_k, we maintain the list of chains that are contained in L_k and store a pointer to this list on C_k and on the minimal chain in L_k.

Let C_i be the current chain, i.e., all chains C_j with $j < i$ satisfy the preconditions of Theorem 22. Let S_l be the current subdivision. Then $\mathit{Type}_3(C_i)$ contains exactly the chains of type 3 that start in C_i and are not contained in S_l, because the chains $C_i \neq C_0$ that start with $s(C_i)$ or $t(C_i)$ cannot be contained in S_l. Let X be the children of C_i that are not contained in S_l. It follows from the first argument in the proof of Theorem 22 that $X \subseteq \mathit{Children}_{12}(C_i)$. In order to ensure $X = \mathit{Children}_{12}(C_i)$, all chains in $\mathit{Children}_{12}(C_i)$ that are contained in S_l are deleted in time $O(|E(C_i)|)$.

We partition $\mathit{Type}_3(C_i)$ into the classes of chains that are contained in the same segment of S_l. This is done by storing a pointer for each $C_j \in \mathit{Type}_3(C_i)$ to the minimal chain D_k of the segment H that contains C_j. If D_k is a backedge, H consists only of one chain $D_k = C_j$, which can be added immediately due to Lemma 18. For every chain $C_j \in \mathit{Type}_3(C_i)$ in a segment H containing at least two chains, we traverse and mark the path $(t(C_j) \to_T u) \setminus \{u\}$, where u is the first ancestor of $t(C_j)$ in T that is contained in S_l or marked. Let P be the first traversed path in H. Then the last but one vertex of P must be an inner vertex of the minimal chain D_k in H and we use D_k as marker for P, for all paths that end at P and for all further paths that are contained in H. This way C_k can be computed by traversing only the chains that we can add by Theorem 22; the total running time amortizes therefore to $O(n+m)$.

Let now H be the segment of a chain $C_j \in \mathit{Type}_3(C_i)$. If the minimal chain D in H is not of type 1 or $2b$, we can immediately add C_j along with its ancestors in H due to Lemma 19. Otherwise, D is contained in $\mathit{Children}_{12}(C_i)$ and the previous computation provides the set $H \cap \mathit{Type}_3(C_i)$. If D is of type $2a$ and $t(D)$ is real, we add D. Now every remaining chain D in $\mathit{Children}_{12}(C_i)$ is the chain D of one of the cases 22.1-22.3 with the only exception that there may be inner real vertices on its dependent path $P \subseteq C_i$. If P contains an inner real vertex, each chain in $H \cap \mathit{Type}_3(C_i)$ can be added along with its ancestors in H. If P does not contain an inner real vertex, no chain in H can be added under (R_1) and (R_2). It is therefore possible to restrict ourselves to find an ordering on the chains $\mathit{Children}_{12}(C_i)$, because the chains in the segments of every such chain can be added immediately afterwards. However, we lack an efficient way of choosing subsequently chains in $\mathit{Children}_{12}(C_i)$ that contain an inner real vertex in their dependent path, although Theorem 22 guarantees the



(a) The chain C_i with the chains in $\mathit{Type}_3(C_i) \cup \mathit{Children}_{12}(C_i)$.

(b) The dashed intervals in I_0 are constructed from the real vertices in $C_i \setminus s(C_i)$.

Figure 6: Mapping the dependent paths in the chain C_i to intervals. Different shades depict different segments.

existence of such chains. We deal with this problem by the following approach.

Let H be the segment that contains a chain $D \in \mathit{Children}_{12}(C_i)$. We map the dependent path P of D to a set of integer intervals $I(D)$ on the path $t(C_i) \rightarrow_{C_i} s(C_i)$, where vertices depict integers. If D is a backedge, $H = \{D\}$ holds and we set $I(D) := \{[s(P), t(P)]\}$ (see Figure 6). Otherwise, D must be in case 22.1 or 22.3 and H contains a chain of type 3 that starts at an inner vertex of P by Lemma 21. Then the precomputed set $H \cap \mathit{Type}_3(C_i)$ is non-empty and we can efficiently extract the set $\{v_1, \ldots, v_k\}$ of vertices in $H \cap P$, $k \geq 3$, ordered by increasing distance to $r \in V(T)$; note that every vertex in $H \cap P$ occurs only once and that $t(D)$ is also contained in $H \cap P$ if D is of type 2*b*. We set $I(D) := \{[v_1, v_k], [v_1, v_2], [v_2, v_3], \ldots, [v_{k-1}, v_k]\}$. Additionally, we add for the ordered set $\{t(C_i) = w_1, w_2, \ldots, w_u = s(C_i)\}$ of real vertices in C_i, $u \geq 3$, the set of intervals $I_0 := \{[w_1, w_2], [w_2, w_3], \ldots, [w_{u-2}, w_{u-1}]\}$. Let I be the set of all generated intervals. By construction, $|I|$ is at most $\frac{3}{2} * |\mathit{Children}_{12}(C_i)| + |\mathit{Type}_3(C_i)| + |\{x | x \text{ is inner real node in } C_i\}|$, where the first two terms are linearly dependent on the length of chains that will be added and the last term sums up to at most n in total.

Let two intervals $[a, b]$ and $[c, d]$ *overlap* if $a < c < b < d$ or $c < a < d < b$. We find the next chain having an inner real vertex on its dependent path by finding a next overlapping interval when starting with an interval in I_0. We show that these two methods are equivalent. Clearly, an overlap of two intervals, of which exactly one is contained in the preimage of a chain in S_l, induces an inner real vertex in the other interval. Conversely, let D_2 be a chain in $\mathit{Children}_{12}(C_i)$ with the dependent path P_2, let H be the segment containing D_2 and let D_1

be the chain that provides the first inner real vertex for P_2 (real vertices can only be provided by the remaining chains in $\mathit{Children}_{12}(C_i) \cup \mathit{Type}_3(C_i)$). Then $[s(P_1), t(P_1)] \in I(D_1)$ overlaps $[s(P_2), t(P_2)] \in I(D_2)$ or the start vertex v of a chain in $H \cap \mathit{Type}_3 C_i$ is an inner vertex of $[s(P_2), t(P_2)]$. In the latter case, v is by construction the end point of a chain in $I(D_1)$. It therefore suffices to find an ordering π of I that starts with an interval of I_0 such that for every two subsequent intervals I_1 and I_2 in π holds

- I_1 overlaps I_2,
- $I_2 \subseteq I_0$ or
- $I_2 \subset I(D)$ for a chain D such that $I(D)$ contains a smaller interval in the ordering.

For the set I of intervals, let the *overlap graph* O of I be the graph (I, E') with two intervals being in E' if and only if they overlap. If we drop the three latter conditions, there exists an sequential algorithm finding π in time $O(|I|)$ by computing the connected components of O [3]. Note that the connected components give us also the information whether there exists π in the case of non-3-connected graphs. The Lemmas 4.1 and 4.2 in [3] allow for a simplified variant of this algorithm, which first constructs by two simple sweep lines the forests F_1 and F_2 such that the connected components of $F_1 \cup F_2$ (having at most $2|I|$ edges) partitions I into the same vertices as the connected components of O. We extend this algorithm to deal with our additional three conditions by merging the intervals in I_0 and $I(D)$ for each $D \in \mathit{Children}_{12}(C_i)$ to the same connected component; the number of additional edges in $F_1 \cup F_2$ is at most $|I|$ and the asymptotic linear running time is preserved. Using Corollary 23, this gives the following result.

Corollary 24. The construction sequences *2.*(2), *2.*(3), *2.*(4) and *2.*(5) of a 3-connected graph can be computed in time $O(n + m)$.

Remark. The algorithm should easily extend to a certifying 3-connectivity test. The main argument is that, for a graph G that is not 3-connected, a subdivision must occur for which no BG-path exists by Theorem 2. But every path that contradicts property 1.2 or 1.3 leads to a cut vertex or separation pair. It would be interesting to show how to get implicitly all the separation pairs (and the unique SPQR-tree), as done in [2].

Remark. Note that C_i can already be processed when all chains with a starting vertex in $\mathit{low}(C_i) \to_T r$ for r being the root of T have been found. Therefore, the chain decomposition does not have to be finished for testing whether a proper ordering on the chains $\mathit{Children}_{12}(C_i) \cup \mathit{Type}_3(C_i)$ exists and these chains can immediately be added in the affirmative case. In that sense, the 3-connectivity is checked locally for each chain C_i; it might be interesting whether it is faster than the algorithm of Hopcroft and Tarjan [2] in practice.

5 Verifying the Construction Sequence

The certificate of the 3-connectivity of G, e.g., the construction sequences 2.2 and 2.3, can be stored in $O(n + m)$ space and easily verified in time $O(n + m)$ as described in [4].

References

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

[3] S. Olariu and A. Y. Zomaya. A time- and cost-optimal algorithm for interlocking sets – With applications. *IEEE Trans. Parallel Distrib. Syst.*, 7(10):1009–1025, 1996.

[4] J. M. Schmidt. Construction sequences and certifying 3-connectedness. In *27th International Symposium on Theoretical Aspects of Computer Science (STACS'10), Nancy, France*, pages 633–644, 2010.

[5] W. T. Tutte. A theory of 3-connected graphs. *Indag. Math.*, 23:441–455, 1961.

[6] W. T. Tutte. Connectivity in graphs. In *Mathematical Expositions*, volume 15. University of Toronto Press, 1966.

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

[8] K.-P. Vo. Segment graphs, depth-first cycle bases, 3-connectivity, and planarity of graphs. *Linear and Multilinear Algebra*, 13:119–141, 1983.