Handling Proper Minor-Closed Graph Classes in Linear Time: Shortest Paths and 2-Approximate Steiner Trees

Matthias Müller-Hannemann, Siamak Tazari

Technical Report 2007/5
Handling Proper Minor-Closed Graph Classes in Linear Time: Shortest Paths and 2-Approximate Steiner Trees

Matthias Müller-Hannemann, Siamak Tazari

Handling Proper Minor-Closed Graph Classes in Linear Time: Shortest Paths and 2-Approximate Steiner Trees

Matthias Müller-Hannemann, Siamak Tazari

Abstract
We generalize the linear-time shortest-paths algorithm for planar graphs with nonnegative edge-weights of Henzinger et al. (1994) to work for any proper class of minor-closed graphs. We show by a counter-example that their algorithm can not be adapted straightforwardly to all proper minor-closed classes. By using recent deep results in graph minor theory, we show how to construct an appropriate recursive division in linear time for any graph excluding a fixed minor and how to transform the graph and its division afterwards, so that it has maximum degree three. Based on such a division, the original framework of Henzinger et al. can be applied. Afterwards, we show that using this algorithm, one can implement Mehlhorn’s (1988) 2-approximation algorithm for the Steiner tree problem in linear time on these graph classes.

Keywords: Minor-Closed, Shortest Path, Steiner Tree, Planar Graph

1 Introduction
The single-source shortest-paths problem with nonnegative edge-weights is one of the most-studied problems in computer science, because of both its theoretical and practical importance. Dijkstra’s classical algorithm [7] has ever since its discovery been one of the best choices in practice. Also from a theoretical point of view, until very recently, it had the best running time in the addition-comparison model of computation, namely $O(m + n \log n)$ using Fibonacci heaps [10] (we use $n$ to denote the number of vertices of a graph and $m$ for its number of edges). Pettie and Ramachandran [21] improved the theoretical running time in undirected graphs for the case when the ratio $r$ between the largest and smallest edge-weight is not too large. They achieve a running time of $O(m \alpha(m, n) + \min\{n \log n, n \log \log r\})$, where $\alpha(m, n)$ is the very slowly growing inverse-Ackermann function. Goldberg [12] proposed an algorithm that runs on average in linear time. For the case of integer edge-weights, Thorup [26] presented a linear-time algorithm in the word RAM model of computation, where the bit-manipulation of words in the processor is allowed. Hagerup [13] extended and simplified Thorup’s ideas to work for directed graphs in nearly linear time. But the question whether the standard addition-comparison model allows shortest-paths computation in worst-case linear-time is still open. For a fairly recent survey about shortest-paths algorithms, see Zwick [27].

For planar graphs, Henzinger et al. [14] presented the first linear-time algorithm to calculate shortest-paths with nonnegative edge-weights. Their algorithm works on directed graphs. It is based on Frederickson’s [8, 9] work who gave an $O(n \sqrt{\log n})$-time algorithm for this case and
whose idea was in turn based on planar separators [16] to decompose the graph. Henzinger et al. first decompose the graph into a recursive division and then use this division to relax the edges in a certain order that guarantees linear running time. They claim that their algorithm can be adapted to work for any proper minor-closed family of graphs where small separators can be found in linear time. Reed and Wood [23] showed that this is the case for all proper minor-closed graph classes and so, we should be done. However, both Frederickson’s algorithm and Henzinger et al.’s algorithm assume that the graph has maximum degree 3 and while this property can be achieved easily for planar graphs, we show that the standard “trick” cannot be arbitrarily applied to all proper minor-closed classes (in particular, it cannot be applied to apex graphs, i.e. planar graphs augmented by a “super-source”; these graphs have frequent application in the literature).

We show how to build an appropriate recursive division of a graph from a proper minor-closed family in linear time by a non-trivial extension of the algorithm in [14]. Our algorithm works for graphs with arbitrary degrees. But even after having the recursive division, the shortest paths algorithm in [14] depends on the assumption that the graph has bounded degree (and contains only a single source labeled initially with distance zero, cf. apex graphs). Using the recursive division, we show how to transform the graph and its division to have maximum degree 3, so that Henzinger et al.’s shortest-paths algorithm can be applied. Our modifications lead to the first shortest-paths algorithm for all proper minor-closed classes of graphs that runs in linear time in the addition-comparison model of computation.

We also consider the Steiner tree problem, namely finding the shortest tree that connects a given set of terminals in an undirected graph. The Steiner tree problem is also one of the most fundamental problems in computer science and of the first problems shown to be \(\mathcal{NP}\)-complete by Karp [15]. Bern and Plassmann [1] showed that it is even \(\mathcal{APX}\)-hard and the best-known non-approximability result is due to Chlebík and Chlebíková [5] who showed a bound of \(96/95 \approx 1.01053\). Robins and Zelikovsky [25] presented an algorithm with approximation guarantee \(1 + \frac{\ln 3}{2} + \epsilon \approx 1.55 + \epsilon\) which is the best approximation algorithm for this problem known so far. There is a well-known 2-approximation algorithm for this problem [6, 22] that is based on finding the minimum spanning tree of the complete distance network of the set of terminals. Mehlhorn [19] improved the running time of this algorithm to \(O(m + n \log n)\).

The Steiner tree problem in planar graphs is also \(\mathcal{NP}\)-hard [11] but very recently a polynomial time approximation scheme (PTAS) has been found by Borradaile et al. [2, 3] for this case. The running time of the PTAS is \(O(n \log n)\) with a constant factor that is exponential in the inverse of
the desired accuracy. As an application of our shortest-paths algorithm, we show how to implement Mehlhorn’s [19] 2-approximation algorithm in linear time on proper minor-closed graph classes. No better time bound than Mehlhorn’s own implementation of $O(m + n \log n)$ has previously been known even for planar graphs. An important observation that we made to improve this running time is that Mehlhorn’s distance network is a minor of the given graph and thus, its minimum spanning tree can be calculated in linear time with the algorithm of Mares [18].

In Section 2, we review some needed concepts and previous work; in Section 3, we present our main result about shortest paths and in Section 4, the application to Steiner tree approximation.

2 Preliminaries

In this section, we review some concepts and some previous results that are needed in this work. These include graph minors, vertex partitioning, graph decomposition, and an overview of Henzinger et al.’s [14] single-source shortest-paths algorithm.

2.1 Graph Minors

A minor of a graph $G$ is a graph that is obtained from a subgraph of $G$ by contracting a number of edges. A class of graphs is minor-closed if it is closed under building minors. It is called a proper class if it is neither empty nor the class of all graphs. Examples of proper minor-closed graph families are planar graphs and bounded-genus graphs. A seminal theorem of Robertson and Seymour [24] states that any proper minor-closed class of graphs can be characterized by a finite set of excluded minors. This is a very broad generalization of Kuratowski’s theorem about planar graphs. Note that for a proper minor-closed class of graph, we can always consider the number of vertices $\ell$ of the smallest excluded minor and conclude that the complete graph $K_\ell$ is an excluded minor of the class.

It follows from a theorem of Mader [17] that proper minor-closed classes of graphs have constant average degree, for some constant depending on the smallest excluded minor of the class. This, in turn, implies that these classes of graphs are sparse, i.e. we have $m = O(n)$. For planar graphs, we know by Euler’s formula that the number of edges is at most only $3n - 6$.

2.2 Vertex Partitioning

In [8], Frederickson presented a simple algorithm called FindClusters, based on depth-first search, that given a parameter $z$ and a graph with maximum degree 3, partitions its vertices into connected components each having at least $z$ and at most $3z$ vertices. Note that since the algorithm gives us
connected components, we can contract each one of them and get a minor of the input graph with at most \( n/z \) vertices. He used this algorithm to derive fast algorithms for the minimum spanning tree and shortest-paths \([9]\) problems. If a weighted graph does not have maximum degree 3, one can apply the following transformation: replace a vertex \( v \) of degree \( d(v) \) with a zero-weight cycle of length \( d(v) \), such that each edge incident to \( v \) is now incident to exactly one vertex of the cycle (a similar transformation can be applied to directed graphs, too). If the given graph is planar, one can consider a planar embedding and order the edges around a cycle in the same way they were ordered around the corresponding vertex in the given embedding. This way, the transformed graph will also be planar. The same is true for bounded genus graphs. However, for an arbitrary minor-closed class of graphs (e.g. apex graphs), it might not always be possible to remain in the class after transforming the graph this way, see Subsection 3.1. But Frederickson’s \texttt{FindClusters} depends on the graph having bounded degree. Any constant bound would suffice for our purposes but in general such a bound does not exist for arbitrary minor-closed graph families.

Reed and Wood \([23]\) introduced an alternative partitioning concept that can be applied to a graph \( G = (V, E) \) with arbitrary degrees excluding a fixed minor. Consider some partitioning \( P = \{P_1, \ldots, P_t\} \) of the vertex set \( V \). Let \( H = (V_H, E_H) \) be the graph obtained by collapsing every part \( P_i \) of \( G \) into a single vertex \( v_i \in V_H \) (\( 1 \leq i \leq t \)) and removing loops and parallel edges. This way, there is an edge between two vertices \( v_i \) and \( v_j \) of \( H \) if and only if there is an edge between a vertex of \( P_i \) and a vertex of \( P_j \) in \( G \) (\( 1 \leq i < j \leq t \)). We say \( P \) is a connected \( H \)-partition of \( G \) if \( v_iv_j \in E_H \) if and only if there is an edge of \( G \) between every connected component of \( P_i \) and every connected component of \( P_j \). Reed and Wood proved the following lemma\(^1\):

\begin{lem} \textnormal{([23])}. There is a linear-time algorithm that given a constant \( z \) and a graph \( G \) excluding a fixed minor, outputs a connected \( H \)-partition \( P = \{P_1, \ldots, P_t\} \) of \( G \) such that \( t \leq n/z \), and \( |P_i| < c_0 \cdot z \) for all \( 1 \leq i \leq t \), where \( c_0 \) is a constant depending on the size of the excluded minor. \end{lem}

Note that by contracting each connected component of each \( P_i \) in \( G \) to a single vertex, one gets a graph that contains an isomorphic copy of \( H \) as a subgraph and so, \( H \) is a minor of \( G \) and in particular, excludes the same fixed minor as \( G \). Hence, when dealing with graphs with no bounded degree, Lemma 2.1 can be used instead of \texttt{FindClusters} to partition the graph and reduce its size while keeping it free of some fixed minor.

\(^1\)In their lemma, we substitute \( c_0 := 2^{r^2+1} \) and \( z := 2k/c_0 \), where \( K \) is the smallest excluded minor of \( G \).
2.3 Graph Decomposition

A balanced node-separation of a graph $G = (V, E)$ is given by two sets $A$ and $B$, such that $A \cup B = V$, there is no edge between $A \setminus B$ and $B \setminus A$, and each one of $A$ and $B$ contains at most an $\alpha$-fraction of the nodes (for some $1/2 \leq \alpha < 1$). The size of the separation is $|A \cap B|$. For a function $f$, a subgraph-closed class of graphs is said to be $f$-separable if every $n$-node graph in the class has an $O(f(n))$-size separator. Reed and Wood [23] showed that all proper minor-closed classes of graphs are $f$-separable in linear time for $f(n) = O(n^{2/3})$. For planar graphs, one can use the original planar separator theorem of Lipton and Tarjan [16] that delivers an $O(\sqrt{n})$-separator in linear time.

An $(r, s)$-division of an $n$-node graph is a partition of the edges of the graph into $O(n/r)$ regions, each containing $r^{O(1)}$ nodes and each having at most $s$ boundary nodes (i.e. nodes that occur in more than one region). For a nondecreasing positive integer function $f$ and a positive integer sequence $\tau = (r_0, r_1, \ldots, r_k)$, an $(\tau, f)$-recursive division of an $n$-node graph is defined as follows: it contains one region $R_G$ consisting of all of $G$. If $G$ has more than one edge and $\tau$ is not empty, then the recursive division also contains an $(r_k, f(r_k))$-division of $G$ and an $(\tau', f)$-recursive division of each of its regions, where $\tau' = (r_0, r_1, \ldots, r_{k-1})$. A recursive division can be represented compactly by a recursive division tree, a rooted tree whose root represents the whole graph and whose leaves represent the edges of the graph. Every internal node represents a region, namely, the region induced by all the leaves in its subtree. The children of a node of the tree are its immediate subregions in the recursive division.

Using Frederickson’s partitioning [8] and division [9] methods, Henzinger et al. [14] present a linear-time algorithm to find certain recursive divisions in planar graphs: they determine a vector $\tau$ and an $(\tau, cf)$-recursive division of the graph for some constant $c$, such that the inequality

$$\frac{r_i}{f(r_i)} \geq 8^if(r_{i-1})\log r_{i+1}(\sum_{j=1}^{i+1} \log r_j) \tag{1}$$

is satisfied for all $r_i$’s exceeding a constant. The obtained recursive division tree has $O(n)$ nodes and its depth is roughly $O(\log^* n)$. They claim that for $\epsilon > 0$, their algorithm can be modified to work for any $O(n^{1-\epsilon})$-separable minor-closed class of graphs that can be separated in linear time. This is indeed the case when the graph has maximum degree 3. However, they do not explain how this bound on the degree can be achieved when the graph is not planar (see Subsections 2.2 and 3.1).
2.4 Single-Source Shortest-Paths on Planar Graphs

Henzinger et al.'s [14] single-source shortest-paths algorithm can be summarized as follows: assume that a directed graph \( G \) with in-/outdegree at most 2 is given, equipped with an \((7, cf)\)-recursive division tree for some constant \( c \). Maintain a distance label for every node in the graph. Initialize the distance labels to infinity except for the source, which is initialized to zero (it is important that there is only one source with outdegree 2). Create a priority queue for each internal node of the recursive division tree and insert all the children of that node into it; the key of each child is the label of the smallest vertex with an outgoing edge in the region the child represents. Then perform edge-relax operations in an (complex) order that is specified by the priority queues and the recursive division tree until the distance labels provably correspond to the shortest-path distances from the source. They prove the following theorem:

**Theorem 2.2 ([14]).** Let a graph \( G \) with maximum in-/outdegree 2 and, for some constant \( c \), an \((7, cf)\)-recursive division tree of \( G \) be given, such that inequality (1) is satisfied for all \( r_i \)'s exceeding a constant. Then, the single-source shortest-paths problem with nonnegative edge-weights can be solved on \( G \) in linear time.

To prove this theorem, they use a complicated charging scheme that also depends on the graph having a single source and bounded degree. Together with the result from the previous subsection, it follows that single-source shortest-paths with nonnegative edge-weights can be calculated in linear-time on planar graphs. But for arbitrary proper minor-closed classes of graphs, some non-trivial changes and extra work needs to be done, as is shown in the next section.

3 Single-Source Shortest Paths on Minor-Closed Graph Classes

In this section, we prove our main theorem about shortest paths:

**Theorem 3.1.** In every proper minor-closed class of graphs, single-source shortest-paths with nonnegative edge-weights can be calculated in linear time.

We proceed as follows: first, we show below that the transformation of Subsection 2.2, when done arbitrarily on some minor-closed family of graphs, might result in a \( K_\ell \)-minor for any \( \ell \in \mathbb{N} \). We propose a modification of the recursive division algorithm of [14] that replaces Frederickson's algorithm with the \( H \)-partitioning algorithm of Reed and Wood [23] and that can be applied to any graph \( G \) with arbitrary degrees excluding some fixed minor. Then we show how the graph and
its recursive division can be modified to have maximum in-/outdegree 2 (the underlying undirected graph will have maximum degree 3), so that Theorem 2.2 can be applied.

3.1 A Counterexample

When applying the transformation from Subsection 2.2 to a planar graph to obtain a graph with maximum degree 3, we make use of the ordering of the edges in a planar embedding of the graph to make the transformed graph planar. But for an arbitrary class of minor-closed graphs, such an ordering is not given. We investigated the question about what happens if we allow any ordering of the edges: if the given class of graphs excludes some fixed minor, will the transformed graphs also exclude some fixed minor? Unfortunately, this is not the case, as is illustrated in Fig. 1. In part (a), we see a graph excluding $K_{4,2}$. But if we transform the graph and order the edges as in part (b), contracting the thick edges will result in a $K_6$ minor (c). By taking longer chains of parallel edges, one can produce any $K_\ell$ minor in this way. Even if we do not allow parallel edges, one can produce transformed graphs that include arbitrary minors: the planar graph given in Fig. 1(d) can be used to this end. These examples also demonstrate that if we assign an arbitrary ordering of edges around each vertex and use it to embed the graph on some surface (see [20]), the resulting surface might become arbitrarily complex, i.e. its smallest excluded minor might be arbitrarily larger than the smallest excluded minor of the given graph. This shows that we may not assume without loss of generality that the given graph for the shortest-paths computation has bounded degree$^3$. But exactly this property is assumed by Henzinger et al. when generalizing their algorithm to broader classes of graphs.

---

$^2$since the shown graph has a subgraph homomorphic to $K_4$, we choose $K_4$ as the smallest excluded minor.

$^3$unless there is some other more complicated trick to achieve this property.
Algorithm 3.1: Generalized Recursive Division Algorithm

**Input**: An undirected graph $G = (V, E)$ excluding a $K_r$-minor.

**Output**: A recursive division tree $T$ for $G$ satisfying inequality (1) for all $r_i$ exceeding a constant.

\begin{algorithm}
\begin{algorithmic}
\STATE begin
\STATE \hspace{1em} // partition and contract the graph recursively
\STATE \hspace{1em} let $G_0 := G$, $z_0 := 2$, $i := 0$;
\STATE \hspace{1em} while the number of nodes in $G_i > \frac{n}{\log n}$ do
\STATE \hspace{2em} let $G_{i+1} := H$-Partition($G_i, z_i, \ell$);
\STATE \hspace{2em} let $z_{i+1} := 14z_i^i$, $i := i + 1$;
\STATE \hspace{1em} let $I := i - 1$;
\STATE \hspace{1em} // divide the graphs and build recursive division tree
\STATE \hspace{1em} let $v_G$ be the root of $T$;
\STATE \hspace{1em} let $D_{i+1}$ be the trivial division of $G_{i+1}$ consisting of a single region;
\STATE \hspace{1em} for $i := I$ downto 0 do
\STATE \hspace{2em} for each region $R$ of $D_{i+1}$ do
\STATE \hspace{3em} let $S_R$ be the boundary-nodes of $R$ in the division $D_{i+1}$;
\STATE \hspace{3em} let $D_R := \text{Divide}(R, S_R, z_i, \ell)$;
\STATE \hspace{3em} for each region $R'$ of $D_R$ do
\STATE \hspace{4em} expand $R'$ into a region $R''$ of $G_i$ by expanding every vertex;
\STATE \hspace{4em} assign each boundary edge to one of the regions it occurs in;
\STATE \hspace{4em} create a child $v_{R''}$ of $v_R$ in $T$;
\STATE \hspace{2em} let $D_i$ be the decomposition of $G_i$ consisting of the regions $R''$ above;
\STATE \hspace{1em} // add the leaves
\STATE \hspace{1em} for each edge $e$ of each region $R$ of $D_0$ do
\STATE \hspace{2em} create a child $v_e$ of $v_R$ in $T$;
\STATE \hspace{1em} return $T$;
\STATE end
\end{algorithmic}
\end{algorithm}

3.2 Our Generalized Recursive Division Algorithm

Henzinger et al.’s [14] recursive division algorithm works roughly as follows: it partitions and reduces the size of the graph using a procedure called Contract. This procedure, takes a parameter $z$ and calls Frederickson’s FindClusters (see Subsection 2.2) to partition the graph into connected components of size between $z$ and $3z$ and then contracts each part. This is done recursively with the values of $z$ defined as $z_0 = 2$, $z_{i+1} = 7z_i^i$. They let $G_0 := G$ and define $G_i$ to be the graph obtained by applying Contract to $G_{i-1}$ with parameter $z_{i-1}$ for $i > 0$. This is done until the size of some $G_i$ drops below $\frac{n}{\log n}$. They let $I$ be this last value of $i$ minus 1. Then they start considering the $G_i$ backwards from $I$ downto 0 and divide each $G_i$ into regions using the division obtained from $G_{i+1}$ and building the recursive division tree at the same time. The values of $z_i$ specified above are chosen so that the sizes of the $G_i$ become small enough to guarantee an overall runtime of $O(n)$. We now describe what has to be modified in this algorithm in order to prove the following central lemma:
Lemma 3.2. There is a linear-time algorithm that given a graph \( G \) excluding some fixed minor, finds an \((r,f)\)-recursive division of \( G \) that satisfies inequality (1) for all \( r \), exceeding a constant and whose recursive division tree has \( O(n) \) nodes.

Proof. The modified algorithm is given in Algorithm 3.1. First of all, we have replaced the \( \text{Contract}(G,z) \) procedure with the connected \( H\text{-Partition}(G,z,\ell) \) procedure of [23]. The connected \( H\text{-Partition} \) algorithm produces a minor \( H \) of \( G \) with at most \( n/z \) vertices, each representing at most \( c_0 z \) vertices of \( G \), in linear time (see Lemma 2.1). It allows graphs with arbitrary degrees, whereas the \text{Contract} procedure uses Frederickson’s \text{FindClusters} and thus requires that the graph has maximum degree 3 (but as we pointed out in the previous subsection, it might not be possible to establish the degree requirement while keeping the resulting graph free of the fixed minor).

The \text{Divide} procedure of Henzinger et al. — taken from Frederickson [9] — is based on an \( O(\sqrt{n}) \)-separation of the graph, such as Lipton and Tarjan’s [16] planar separator. We replace the planar separator in [9] with Reed and Wood’s linear-time \( O(n^{4/3}) \)-separation algorithm for proper minor-closed graph classes. The division procedure takes parameters \( G, S \), and \( r \) and will now have the following properties (see the appendix for a full proof): it divides an \( n \)-node graph \( G \) and a node-subset \( S \) into at most \( c_2(|S|/r^{4/3} + \frac{n}{r}) \) regions, each one having at most \( r \) nodes and at most \( c_1 r^{4/3} \) boundary nodes, where the nodes in \( S \) are counted as boundary nodes, too, and \( c_1 \) and \( c_2 \) are constants. We call the modified procedure \text{Divide}(G,S,r,\ell) to indicate its dependence on the excluded \( K_\ell \)-minor. If the input graph \( G \) has \( n \) nodes, the \text{Divide} procedure takes time \( O(n \log n) \).

Finally, the definition of the sequence \( z_i \) also has to change. For the proof of inequality (1) to work, we need to reduce the exponent in the recursive definition of the \( z_i \) to \( \frac{1}{7} \), i.e. define \( z_{i+1} = 14^{\frac{i}{7}} \). Note that the choice of 7 and 14 as the base of the exponentiations above is not arbitrary; these are the smallest values that ensure that the defined sequences grow (very rapidly) towards infinity.

With the changes given above, the proof of the correctness of the algorithm and all the calculations therein can be carried out in a similar way as is done in [14]; a number of subtle details have to be adapted, as is shown in the appendix. Here, we prove the correctness of inequality (1).

Let \( n_i \) be the number of vertices of \( G_i \). First, note that combining the inequalities \( n_{i+1} \leq n_i/z_i \), we obtain \( n_i \leq n/\prod_{j<i} z_j \). Note moreover that each node of \( G_i \) expands to at most \( \prod_{j<i} c_0 z_j \) nodes of \( G \). Consider the division \( D_i \) of \( G_i \), and the division it induces on \( G \). The division \( D_i \)
consists of \( O(n_1/z_i^2) \) regions (see the appendix for a proof), each having \( O(z_i^2) \) vertices and \( O(z_i^{-2}) \) boundary vertices. This induces \( O(n_1/z_i^2) \) regions in \( G \), each consisting of \( O(z_i^2 \prod_{j<i} c_0 z_j) \) vertices and \( O(z_i^{-2} \prod_{j<i} c_0 z_j) \) boundary vertices. Let \( r_i = z_i^2 \prod_{j<i} c_0 z_j \) and define \( f(r_i) = z_i^{-2} \prod_{j<i} c_0 z_j \). Then, the induced division of \( G \) has \( O(n/r_i) \) regions with each \( O(r_i c_0) \) vertices and \( O(f(r_i)) \) boundary vertices. Since \( c_0 = O(\prod_{j<i} z_j) \), we get that the number of vertices per region is \( O(r_i^2) \).

We have \( f(r_i) = \frac{z_i^2}{z_i^2 c_0^{-1}} = \frac{z_i^{-2}}{c_0} \). Using the definition of \( z_i \), one can verify that \( z_{i-1} = \theta(\log^7 z_i) \) and \( \prod_{j<i} z_j = O(\log^8 z_i) \). Hence \( f(r_{i-1}) = c_0^{i-2} z_{i-1}^{35} \prod_{j<i-1} z_j = c_0^{i-2} O(\log^{35} z_i \log^8 \log z_i) \). We also have \( \log r_{i+1} = \log(\log(z_{i+1}^{21} \prod_{j \leq i} z_j)) = O(\log(z_{i+1}^{38} z_{i+1})) = O(\log z_{i+1}) = O(z_i^{1/4}) \) and consequently \( \sum_{j=1}^{i+1} \log r_j = O(z_i^{1/4}) \). For a sufficiently large constant \( i_0 \), we have for all \( i \geq i_0 \),

\[
8^i f(r_{i-1}) \log r_{i+1} (\sum_{j=1}^{i+1} \log r_j) \leq 8^i c_0^{i-2} O(\log^{35} z_i \log^8 z_i) O(z_i^{1/4}) O(z_i^{1/4}) \\
= 8^i c_0^{i-2} O(z_i^{1/2} \log^20 z_i) \leq \frac{z_i^{-2}}{c_0} \leq \frac{r_i}{f(r_i)},
\]

since the \( z_i \) grow much faster than any exponential function having a constant in the base; specifically, a simple calculation shows that \( z_i^{1/4} \geq g_0 \log^{20} z_i \) for any constant \( g_0 \geq 0 \) if \( i \) is larger than a constant. So, inequality \((1)\) is fulfilled for all \( r_i \) exceeding the constant \( r_{i_0} \).

### 3.3 Establishing The Degree Requirement

Given a directed graph \( G \) with nonnegative edge-weights, excluding some fixed minor, we first consider the underlying undirected graph and apply Lemma 3.2 to get a recursive division of \( G \) with the given properties. In order to be able to find shortest paths in linear time using Theorem 2.2, we have to modify \( G \) so that every vertex has in-/outdegree at most 2. We apply the standard trick of Subsection 2.2 to do so. But now, we define the ordering of the edges around the cycles according to the given recursive division of \( G \). Specifically, we do the transformation as is described in the next lemma:

**Lemma 3.3.** Let \( G \) be an edge-weighted directed graph excluding a fixed minor and let \( T \) be a recursive division tree representing an \((\tau, f)\)-recursive division of \( G \). Then one can replace every vertex of \( G \) with a zero-weight cycle to obtain a graph \( G' \) and at the same time modify \( T \) into a tree \( T' \), so that \( G' \) has in-/outdegree at most 2 and \( T' \) represents an \((\tau, f)\)-recursive division of \( G' \). This modification takes linear time.

**Proof.** Recall that the leaves of \( T \) represent the edges of \( G \) and that internal nodes of \( T \) correspond to regions of \( G \), namely, the region induced by all the leaves in the subtree of that node. We modify
G and T at the same time. First, for every vertex v of G with degree \(d(v)\) (the sum of the indegree and outdegree), we add new vertices \(v_1, \ldots, v_{d(v)}\) to \(G\). We do an in-order traversal of \(T\) and for every leaf of \(T\) representing an edge \(e = vw\) of \(G\), we do the following: let \(e\) be the \(i\)th edge of \(v\) and the \(j\)th edge of \(w\) that we encounter. We change the endpoints of \(e\) to be the vertices \(v_i\) and \(w_j\) and add two new zero-weight edges \(v_i v_{i+1}\) and \(w_j w_{j+1}\) as siblings of \(e\) to \(T\) (if \(i = d(v)\), we use \(v_d(v) v_1\) instead; same for \(w\)). This way, every vertex \(v\) of \(G\) is replaced by a zero-weight cycle \((v_1; \ldots; v_{d(v)})\) (see Fig. 2). The original vertices of \(G\) will become isolated and can be removed. We call the resulting graph \(G'\) and the modified recursive division tree \(T'\). Note that since \(T\) has size \(O(n)\), this procedure takes only linear time. Also note that we only added new leaves to \(T\) and thus, the internal nodes of \(T\) and \(T'\) correspond one-to-one to each other.

Now consider an internal node \(q'\) of \(T'\). It represents a region \(R'\) of \(G'\) and corresponds to a node \(q\) of \(T\), representing a region \(R\) of \(G\). \(R\) has \(r^{O(1)}\) vertices (and edges) and \(O(f(r))\) boundary-nodes. The number of edges of \(R'\) is at most 3-times as much as in \(R\) and the number of vertices is proportional to the number of edges of \(R\). But \(R\) is a subgraph of \(G\), excludes the same fixed minor and thus, the number of its edges is linear in the number of its vertices. Hence, \(R'\) still has \(r^{O(1)}\) vertices and edges. Also, since \(R\) is represented by the subtree rooted at \(q\), its edges were traversed in order while building \(T'\) and \(G'\). So, every vertex \(v\) in \(R\) will be replaced by a path \(v_i, v_{i+1}, \ldots, v_j\) with \(1 \leq i \leq j \leq d(v)\) in \(R'\). Thus, if \(v\) is a boundary node of \(R\), then instead, we have \(v_i\) and \(v_j\) as boundary nodes of \(R'\). So \(R'\) has at most twice as many boundary nodes as \(R\), i.e. still \(O(f(r))\). So, \(T'\) represents an \((r, f)\)-recursive division of \(G'\).

Proof of Theorem 3.1. Note that up to the choice of the start- and endvertex inside the zero-weight cycles of \(G'\), shortest paths in \(G\) and \(G'\) correspond one-to-one to each other. \(G'\) fulfills all the requirements of Theorem 2.2 and combining this with Lemma 3.2, and Lemma 3.3, we obtain our main theorem, namely, Theorem 3.1. 
4 Steiner Tree Approximation

We show how to implement Mehlhorn’s 2-approximation algorithm for the Steiner tree problem [19] in linear time on proper minor-closed graph classes using the result above and the observation that Mehlhorn’s distance network is a minor of the input graph. First, we briefly review Mehlhorn’s algorithm and then we present our implementation.

4.1 Overview of Mehlhorn’s Algorithm

Given an $n$-node graph $G = (V, E)$ with nonnegative edge-weights and a node-subset $K$ of terminals, one can determine a Steiner tree of $K$ in $G$ as follows: first, build Mehlhorn’s distance network $N^*_D = (K, E^*_D)$, a special graph defined on the set of terminals, in which every edge corresponds to a path in $G$. To calculate $N^*_D$, we first have to partition the graph into Voronoi regions with respect to the set of terminals $K$. Every vertex of the graph belongs to the Voronoi region of its closest terminal (if a vertex happens to have the same distance to more than one terminal, it should belong to the Voronoi region of the terminal with the smallest index). Voronoi regions in graphs can be calculated easily using a shortest-paths computation: add a super-source $s_0$ to the graph and connect it to every terminal with a directed zero-weight edge; find the shortest paths from $s_0$ to every vertex and then remove $s_0$ from the resulting shortest-paths tree. The tree falls apart into $|K|$ connected components, each having a terminal as their root. These components correspond exactly to the Voronoi regions of the terminals. Using Dijkstra’s algorithm, one obtains a running time of $O(n \log n + m)$ for general graphs.

In the distance network $N^*_D$, there exists an edge between two terminals $u$ and $v$ if and only if there exists an edge between two vertices $x$ and $y$ in $G$, so that $x$ belongs to the Voronoi region of $u$ and $y$ belongs to the Voronoi region of $v$. The weight of such an edge is the length of the shortest such paths connecting $u$ and $v$. Once the Voronoi regions of $G$ with respect to $K$ are determined, $N^*_D$ can be constructed in linear time using bucket sort.

After the distance network $N^*_D$ is determined, one can find its minimum spanning tree and replace every edge with the corresponding path in $G$. Mehlhorn shows that the resulting graph is indeed a tree and its weight is at most $(2 - \frac{2}{|K|})$ times the weight of the minimum Steiner tree of $K$ in $G$. The implementation he offers runs in time $O(n \log n + m)$ for general graphs.
4.2 A Linear-Time Implementation for Proper Minor-Closed Classes

**Theorem 4.1.** There is a linear-time algorithm that calculates a 2-approximation for the Steiner minimum tree problem in any proper minor-closed class of graphs.

We first show how to find the Voronoi regions in linear time. In graphs excluding a fixed minor $K_d$, we observe that the graph with an added super-source will exclude $K_{d+1}$; so, Theorem 3.1 applies and shortest paths can be calculated in linear time. Alternatively, using a similar method as in Subsection 3.3, one can first find a recursive division of $G$ and then add the super-source and its edges to $G$ and to the recursive division. This could result in much better constants in the running time of the algorithm, especially for planar graphs. We get

**Lemma 4.2.** For a graph $G$ excluding a fixed minor and having nonnegative edge-weights and a given set of terminals $K$ in $G$, the Voronoi regions of $G$ with respect to $K$ can be determined in linear time.

**Corollary 4.3.** In a proper minor-closed class of graphs, the distance network $N^*_D$ can be calculated in linear time for any given set of terminals in a given graph from the class.

The next step of Mehlhorn’s algorithm is to calculate the minimum spanning tree of $N^*_D$. But notice that $N^*_D$ is obtained by contracting the Voronoi regions of the graph (which are connected) and removing loops and parallel edges, i.e.

**Observation 4.4.** For a given graph $G$ and a set of terminals, the distance network $N^*_D$ is a minor of $G$.

Thus, $N^*_D$ belongs to the same proper class of minor-closed graphs as $G$ and one can apply the linear-time minimum spanning tree algorithm of Mares [18]. When we are dealing with planar graphs, the algorithm of Cheriton and Tarjan can be used [4]. As mentioned before, the last step of Mehlhorn’s algorithm is to replace the edges of $N^*_D$ with the corresponding paths from $G$ and this can clearly be done in linear time. Hence, Theorem 4.1 is proven.

**Acknowledgment**

Siamak Tazari was supported by the Deutsche Forschungsgemeinschaft (DFG), grant MU1482/3-1.
References


A Detailed Proof for Lemma 3.2

For the sake of completeness, we have included the detailed proof of Lemma 3.2 below. The proof of Lemma A.1 is based on the original proof of Frederickson [9] and the remaining lemmas and proofs follow the proof in [14]. A number of details and calculations have been filled in and replaced at several places. Lemma A.1 shows the correctness of the Divide procedure and Lemmas A.2 and A.3 serve the analysis of Algorithm 3.1.

Lemma A.1. Replacing the planar separator in Frederickson’s Divide procedure [9] with the separator algorithm of Reed and Wood [23] causes the Divide(G, S, r, ℓ) procedure to work as follows (where G is a graph with n vertices and excludes \( K_\ell \) as a minor and \( c_1 \) and \( c_2 \) are constants depending on \( \ell \)):

- it divides \( G \) into at most \( c_2(|S|/r^{3\ell} + \frac{n}{r}) \) regions;
- each region has at most \( r \) vertices;
- each region has at most \( c_1 r^{\frac{3\ell}{2}} \) boundary vertices, where the vertices in \( S \) also count as boundary;
- it takes time \( O(n \log n) \).

Proof. In the following, when we refer to boundary vertices, we mean vertices that belong to more than one region or vertices that belong to the set \( S \). The Divide procedure works as follows: assign weight \( \frac{1}{n} \) to each vertex of \( G \) and find a \( O(n^{3\ell}) \)-separator in \( G \); recursively apply the separation algorithm to each region with more than \( r \) vertices. Now each region has at most \( r \) vertices. While there is a region with more than \( c_1 r^{\frac{3\ell}{2}} \) boundary vertices, do the following: if such a region has \( n' \) boundary vertices, assign weight \( \frac{1}{n'} \) to each of them, assign weight zero to the other vertices of that region and apply the separator theorem. In the end, all regions will have the desired properties and the algorithm takes time \( O(n \log n) \). It remains to show the bound on the number of the regions.

Consider the division before the regions are further split to enforce the requirement on the number of boundary vertices (i.e. just when we have achieved that each region has size at most \( r \)). Let \( V_B \) be the set of vertices that are included in more than one region. For a vertex \( v \in V_B \), let \( b(v) \) be one less than the number of regions that contain \( v \) in the division. Let \( B(n, r) \) be the total of \( b(v) \) over all vertices \( v \in V_B \). Thus \( B(n, r) \) is the sum of the number of vertices \( v \in V_B \) weighted
by the count $b(v)$. From the separation theorem in [23], we have the following recurrence:

$$B(n, r) \leq d_0 n^{\frac{2}{3}} + B(\alpha n, r) + B((1 - \alpha)n, r) \quad \text{for } n > r,$$

$$B(n, r) = 0 \quad \text{for } n \leq r$$

(3)

where $d_0$ is a constant and $\frac{1}{2} \leq \alpha \leq \frac{2}{3}$. We claim that

$$B(n, r) \leq \frac{d_1 n}{r^{\frac{2}{3}}} - d_2 n^{\frac{2}{3}} \quad \text{for } n \geq \frac{r}{3},$$

(4)

with some constants $d_1$ and $d_2$. The claim can be shown by induction:

As the base of the induction, we consider the cases $\frac{r}{3} \leq n \leq r$. Note that since after splitting a region, each subregion still has at least one-third of the total vertices, it is sufficient to only consider graphs with at least $r/3$ vertices. By choosing $d_1 \geq 3^{\frac{4}{3}} d_2$, we have

$$\frac{d_1 n}{r^{\frac{2}{3}}} \geq \frac{3^{\frac{4}{3}} d_2 n^{\frac{2}{3}}}{3^{\frac{4}{3}} n^{\frac{2}{3}}} = d_2 n^{\frac{2}{3}} \Rightarrow \frac{d_1 n}{r^{\frac{2}{3}}} - d_2 n^{\frac{2}{3}} \geq 0 = B(n, r).$$

(5)

For the inductive step, i.e. for $n > r$, we have

$$B(n, r) \leq d_0 n^{\frac{2}{3}} + d_1 \frac{\alpha n}{r^{\frac{2}{3}}} - d_2(1 - \alpha) n^{\frac{2}{3}} + d_4 \frac{(1 - \alpha)n}{r^{\frac{2}{3}}} - d_2(1 - \alpha) n^{\frac{2}{3}}$$

$$= d_1 \frac{n}{r^{\frac{2}{3}}} - d_2 n^{\frac{2}{3}} + n^{\frac{2}{3}} (d_0 - d_2(1 - \alpha) n^{\frac{2}{3}})$$

$$\leq d_1 \frac{n}{r^{\frac{2}{3}}} - d_2 n^{\frac{2}{3}}$$

(6)

if we choose $d_2 \leq d_2\alpha n^{\frac{2}{3}} + d_2 (1 - \alpha) n^{\frac{2}{3}} - d_0$. This can be achieved by setting $d_2 = 5d_0 \geq \frac{d_0}{\alpha^{\frac{4}{3}} + (1 - \alpha)^{\frac{4}{3}} - 1}$.

In particular, we have shown so far that $B(n, r) = O(n/r^{\frac{2}{3}})$. The sum of the number of vertices in each region is $n + B(n, r) = n + O(n/r^{\frac{2}{3}})$ and each region has $\Theta(r)$ vertices, so the number of regions we have so far is $\Theta(n/r)$.

Let $t_i$ be the number of regions with $i$ boundary vertices (recall that in our definition, the set of boundary vertices is $V_B \cup S$). We have

$$\sum_i i t_i = \sum_{v \in V_B} (b(v) + 1) + |S \setminus V_B| < 2B(n, r) + |S| = O(n/r^{\frac{2}{3}} + |S|).$$

(7)

Let $s(i)$ be an upper bound on the number of splits that have to be applied to a graph with at most $r$ vertices and $i$ boundary vertices, until each of its regions has at most $c_1 r^{\frac{2}{3}}$ boundary vertices, for a constant $c_1$ to be determined. We have that

$$s(i) \leq s(i + d_0 r^{\frac{2}{3}}) + s((1 - \alpha)i + d_0 r^{\frac{2}{3}}) + 1 \quad \text{for } i > c_1 r^{\frac{2}{3}}$$

$$s(i) = 0 \quad \text{for } i \leq c_1 r^{\frac{2}{3}}$$

(8)

where $\frac{1}{2} \leq \alpha \leq \frac{2}{3}$. We claim that

$$s(i) \leq \frac{d_3 i}{c_1 r^{\frac{2}{3}}} - \frac{2d_0 d_3}{c_1} - 1 \quad \text{for } i \geq \frac{c_1 r^{\frac{2}{3}}}{3}$$

(9)
for some constant $d_3$. We prove our claim by induction. Like in the previous induction, for the base case we may assume $\frac{c_1 r^\frac{2}{3}}{3} \leq i \leq c_1 r^\frac{2}{3}$. By choosing $d_3 = 12$ and $c_1 = 8d_0$, we have
\[ \frac{d_3 i}{c_1 r^\frac{2}{3}} - \frac{2d_0 d_3}{c_1} - 1 \geq 12 \cdot \frac{1}{3} - \frac{24d_0}{8d_0} - 1 = 0 = s(i). \tag{10} \]
For the inductive step with $i > c_1 r^\frac{2}{3}$, note that $i + d_0 r^\frac{2}{3} \leq \frac{2}{3}i + \frac{d_3 i}{c_1} \leq (\frac{2}{3} + \frac{1}{3})i < i$. The same way, we have $(1 - \alpha)i + d_0 r^\frac{2}{3} < i$. So, we may apply the induction hypothesis to (8) and a straightforward calculation will prove our claim.

We have shown that for a region with $i$ boundary vertices, where $i > c_1 r^\frac{2}{3}$, at most $\frac{d_3 i}{c_1 r^\frac{2}{3}}$ splits need be done for some constants $c_1$ and $d_3$. This will result in at most $d_0 r^\frac{2}{3}$ new boundary vertices per split and a total of at most $d_3 i/(c_1 r^\frac{2}{3})$ new regions. Thus the total number of new boundary vertices is at most
\[ \sum_i (d_0 r^\frac{2}{3}) \frac{d_3 i}{(c_1 r^\frac{2}{3})} i = \frac{d_0 d_3}{c_1} \sum_i i \cdot t_i = O(n/r^\frac{1}{3} + |S|). \tag{11} \]
The number of new regions is at most
\[ \sum_i (d_3 i/(c_1 r^\frac{2}{3})) i = \frac{d_3}{c_1 r^\frac{2}{3}} O(n/r^\frac{1}{3} + |S|) = O(n/r + |S|/r^\frac{2}{3}). \tag{12} \]

By Lemma 2.1, the $H$-Partition($G, z, \ell$) procedure has the following properties on a graph $G$ with $n$ vertices excluding a $K_\ell$-minor ($c_0$ is a constant depending on $\ell$):

- it divides $G$ into at most $n/z$ regions;
- each region has at most $c_0 z$ vertices;
- it collapses each region into a single node, creating a new graph with at most $n/z$ vertices;
- it takes time $O(n)$.

Recall that we start with the graph $G_0 = G$ and repeatedly apply the $H$-Partition procedure to each $G_i$ to obtain $G_{i+1}$. For each $i$, let $n_i$ denote the number of vertices of $G_i$. Afterwards we work our way back from $G_{I+1}$ to $G_0$ and obtain a division $D_i$ on each $G_i$. Let $k_i$ denote the number of regions of $D_i$. Note that the recursive division tree $T$ has depth $I + 1$.

The following proof has four parts. First, we show that each region of the division $D_i$ has at most $O(z_i^\frac{2}{3})$ vertices and at most $O(z_i^\frac{2}{3})$ boundary vertices. Second, we show that the number $k_i$ of regions is $O(n_i/z_i^\frac{2}{3})$. Third, we show that Algorithm 3.1 takes linear time and finally, we show that the division fulfills inequality (1)\(^4\).

\(^4\)The last part is already shown in the main part of the paper.
For notational convenience, let $z_{i+1} = \sqrt{m_{i+1}}$, so the single region of the division $D_{i+1}$ of $G_{i+1}$ has $z_{i+1}^2$ vertices. Consider iteration $i \leq I$ in the second phase of the algorithm. By the correctness of Divide, the decomposition $D_R$ of a region of $D_{i+1}$ consists of regions $R'$ of size at most $z_i$. By the correctness of H-Partition, each node of $G_{i+1}$ expands to at most $c_0z_i$ nodes of $G_i$. Hence, each region $R''$ obtained from $R'$ by expanding its vertices has size at most $c_0z_i^2$. Similarly, each region $R'$ has at most $c_1z_i^\frac{5}{2}$ boundary vertices by the correctness of Divide, so the corresponding region $R''$ has at most $c_0c_1z_i^\frac{5}{2}$ boundary vertices.

**Lemma A.2.** The number $k_i$ of regions in the division $D_i$ is $O(n_i/z_i^2)$.

*Proof.* We show by reverse induction on $i$ that $k_i \leq c_3n_i/z_i^2$ for all $i \geq i_0$, where $i_0$ and $c_3$ are constants to be determined. For the basis, we have $k_{i+1} = 1$.

Consider iteration $i \leq I$ in the second phase, and suppose $i \geq i_0$. The regions of $D_i$ are obtained by subdividing the $k_{i+1}$ regions comprising the division of $G_{i+1}$. Since $n_{i+1} \leq n_i/z_i$ and $z_{i+1}^2 \geq z_i$, we have by the induction hypothesis that

$$k_{i+1} \leq c_3n_{i+1}/z_{i+1}^2 \leq c_3n_{i}/z_i^2.$$

(13)

Each region $R$ of the division of $G_{i+1}$ has $|S_R| \leq c_0c_1z_{i+1}^\frac{5}{2}$ boundary vertices. Summing over all regions $R$ in $D_{i+1}$, we obtain

$$\sum_R n_R = \sum_R (\text{# of non-boundary vertices} + \text{# of boundary vertices})$$

$$\leq n_{i+1} + \sum_R c_0c_1z_{i+1}^\frac{5}{2}$$

$$\leq n_{i+1} + c_0c_1k_{i+1}z_{i+1}^\frac{5}{2}. \tag{14}$$

For each region $R$, by correctness of Divide, the number of subregions into which $R$ is divided is at most $c_2(|S_R|/z_i^\frac{5}{2} + n_R/z_i)$, which is in turn at most $c_2(c_0c_1z_{i+1}^\frac{5}{2}/z_i^\frac{5}{2} + n_R/z_i)$. Summing over all such regions $R$ and using (14) and (13), we infer that the total number of subregions is at most

$$\sum_R c_2(c_0c_1z_{i+1}^\frac{5}{2}/z_i^\frac{5}{2} + n_R/z_i) = c_0c_1c_2k_{i+1}z_{i+1}^\frac{5}{2}/z_i^\frac{5}{2} + c_2 \sum_R n_R/z_i$$

$$\leq c_0c_1c_2k_{i+1}z_{i+1}^\frac{5}{2}/z_i^\frac{5}{2} + c_2(n_{i+1} + c_0c_1k_{i+1}z_{i+1}^\frac{5}{2})/z_i$$

$$\leq c_0c_1c_2(c_3n_{i+1}/z_{i+1}^2)z_{i+1}^\frac{5}{2}/z_i^\frac{5}{2} + c_2n_{i+1}/z_i + c_0c_1c_2(c_3n_{i+1}/z_{i+1}^2)z_{i+1}^\frac{5}{2}/z_i$$

$$\leq c_0c_1c_2c_3n_{i+1}/(z_i^\frac{5}{2}z_{i+1}^\frac{5}{2}) + c_2n_{i+1}/z_i + c_0c_1c_2c_3n_{i+1}/(z_i^\frac{5}{2}z_{i+1}^\frac{5}{2})$$

$$\leq c_0c_1c_2c_3n_i/(z_i^\frac{5}{2}z_{i+1}^\frac{5}{2}) + c_2n_i/z_i^2 + c_0c_1c_2c_3n_i/(z_i^\frac{5}{2}z_{i+1}^\frac{5}{2}). \tag{15}$$

where in the last line we use the fact that $n_{i+1} \leq n_i/z_i$. We have obtained an upper bound on the total number of subregions into which the regions of $D_{i+1}$ are divided. Each subregion becomes
M. Müller-Hannemann and S. Tazari: Shortest Paths on H-Minor-Free Graphs

a region of \( D_i \). Thus we have in fact bounded \( k_i \), the number of regions of \( D_i \). To complete the induction step, we show that each of the three terms in (15) is bounded by \( c_3 n_i / 3z_i^2 \).

The second term, \( c_2 n_i / z_i^2 \), is bounded by \( c_3 n_i / z_i^2 \) if we choose \( c_3 \geq 3c_2 \). The third term is smaller than the first term. As for the first term, recall that \( z_{i+1} = 14z_i^{4/7} \). For sufficiently large choice of \( i_0 \), we can ensure that \( i \geq i_0 \) implies \( z_{i+1} \geq 3c_0c_1c_2/z_i^{4/7} \). Thus the first term is also bounded as desired.

We conclude that \( k_i \leq c_3 n_i / z_i^2 \), completing the induction step. We have shown this inequality holds for all \( i \geq i_0 \). As for \( i < i_0 \), clearly \( k_i \leq (z_i^2)n_i / z_i^2 \leq (z_{i_0}^2)n_i / z_i^2 \). Thus by choosing \( c_3 \) to exceed the constant \( z_{i_0}^2 \), we obtain the lemma for every \( i \).

**Lemma A.3.** The algorithm runs in linear time.

**Proof.** The time required to form the graphs \( G_1, G_2, \ldots, G_{I+1} \) is \( O(\sum_i n_i/z_i) \), which is \( O(n) \). For \( i \leq I \), the time to apply \texttt{Divide} to a region \( R \) of \( G_{i+1} \) with \( n_R \) vertices is \( O(n_R \log n_R) \). Each such region has \( O(z_{i+1}^2) \) vertices, so the time is \( O(n_R \log z_{i+1}) \). Summed over all regions \( R \), we get \( \sum_R O(n_R \log z_{i+1}) = O(n_{i+1} \log z_{i+1}) \). The time to obtain the induced division of \( G_i \) is \( O(n_i) \). Thus the time to obtain divisions of all the \( G_i \)'s is \( \sum_i O(n_i \log z_{i+1}) \). Since \( n_i+1 \leq n_i / z_i \leq n / z_i \) and \( \log z_{i+1} = O(z_i^{4/7}) \), the sum is \( O(n) \). 

Information about the Author(s)

Matthias Müller-Hannemann  
Institut für Informatik  
Universität Halle  
06099 Halle, Germany  
E-Mail: matthias.mueller-hannemann@informatik.uni-halle.de  
WWW: http://www.informatik.uni-halle.de/muellerh

Siamak Tazari  
Fachbereich Informatik,  
Technische Universität Darmstadt,  
Hochschulstraße 10,  
64289 Darmstadt, Germany  
E-Mail: tazari@algo.informatik.tu-darmstadt.de  
WWW: http://www.algo.informatik.tu-darmstadt.de