Understanding Small Separators in Road Networks

Master's Thesis of

Samuel Born

At the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	T.T.-Prof. Dr. Thomas Bläsius
Second reviewer:	Dr. Torsten Ueckerdt
Advisors:	Adrian Feilhauer
	Michael Zündorf

January 1, 2025 – July 1, 2025

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself. I have not used any other than the aids that I have mentioned. I have marked all parts of the thesis that I have included from referenced literature, either in their original wording or paraphrasing their contents. I have followed the by-laws to implement scientific integrity at KIT.

Karlsruhe, July 1, 2025

. . . . . .

(Samuel Born)

Abstract

Zusammenfassung

Contents

1 Introduction

fluff text

1.1 Motivation

In graph theory, a separator is a subset of vertices whose removal divides the graph into disconnected components of roughly equal size. The size of these separators significantly affects the performance of numerous algorithms, particularly those utilizing divide-and-conquer approaches. For instance, Tarjan's and Lipton's foundational work on planar graphs [LT77] demonstrates their utility in optimizing algorithms for many problems like e.g. maximum flow.

Empirical studies suggest road networks have balanced separators on the order of $\mathcal{O}(n^{1/3})$ [DSW16], which is significantly smaller than the $\mathcal{O}(n^{1/2})$ worst-case bound for planar graphs [LT79]. This is noteworthy given that road networks are can be treated as nearly planar due to their geographic nature (few crossings from overpasses/tunnels).

In road networks, these separators enable the creation of effective node orders, which are critical for the performance of search queries in advanced routing algorithms like CCH. This thesis seeks to uncover the properties responsible for the presence of such small separators in road networks. We aim to determine whether these separators stem from inherent graph characteristics, such as limited vertex degrees or sparsity, or from physical real-world features, such as borders, rivers, or a hierarchical structure. Gaining insight into these properties promises to advance our theoretical understanding and offers practical benefits, such as identifying new applications or generating comparable synthetic graphs.

Road networks represent an intriguing subject for the study of graph separators. Classical results within graph theory primarily establish balanced separators characterized by asymptotic sizes such as $\Theta(1)$, common in structures like trees, or $\Omega(\sqrt{n})$, e.g. planar graphs and unit disk graphs. To the best of our knowledge, established graph classes consistently exhibiting separator sizes strictly between these $\Theta(1)$ and $\Omega(\sqrt{n})$ bounds are not prominently featured in the graph theory literature. This finding positions road networks within this sparsely populated intermediate complexity range, thereby highlighting the compelling nature of investigating their structural properties.

1.2 Contribution

todo

1.3 Outline

todo

2 Preliminaries

fluff text

2.1 Graph Theory

Road networks can be modeled as graphs. A graph G is formally defined as a pair (V, E), where V represents a finite set of vertices (or nodes) and E represents a set of edges connecting pairs of vertices. In many applications, particularly route planning, graphs are augmented with a weight function $w : E \to \mathbb{R}^+$, assigning a positive real value such as distance or travel time to each edge. However, for the purpose of this thesis, the topological strucuture of the graph is of primary interest, and we will not focus on edge weights. We will also only consider simple graphs, meaning graphs without multiple edges between the same pair of vertices and without edges connecting a vertex to itself (loops). Furthermore, as the concept of separators primarily applies to connectivity, we will consider undirected graphs, where edges represent symmetric relationships. An edge connecting vertices u and v in an undirected graph is denoted as the set $\{u, v\}$. The neighborhood of a vertex v is defined as the set of vertices adjacent to v, denoted as $N : V \to \mathcal{P}(V)$.

A graph *embedding* assigns each vertex $v \in V$ of a graph $G = (V, E)$ to a unique point p in a specific geometric space, such as the Euclidean plane $\mathbb{R}^2$ or the surface of a sphere. We then consider edges as straight line segments connecting the points corresponding to their incident vertices.

Throughout this thesis, the term *graph size* refers specifically to the number of vertices, $|V|$. We frequently adopt the notation n for the number of vertices, $|V|$, and m for the number of edges, $|E|$. It is worth noting that for many graph classes discussed herein, particularly sparse graphs like road networks, the number of edges m is asymptotically linear in the number of vertices n. For planar graphs, a specific bound guarantees this linear growth. Euler's formula for connected planar graphs states that $n - m + f = 2$, where f is the number of faces (regions) defined by the graph embedding. By observing that each face is bounded by at least three edges (for $n \geq 3$) and each edge separates at most two faces, we derive the inequality $2m \geq 3f$. Substituting $f = 2 - n + m$ from Euler's formula into this inequality yields $m \leq 3n - 6$. This confirms the linear relationship between the number of edges and vertices for planar graphs. Therefore, the choice of n as the measure of size generally does not impact asymptotic complexity results for the graphs under consideration.

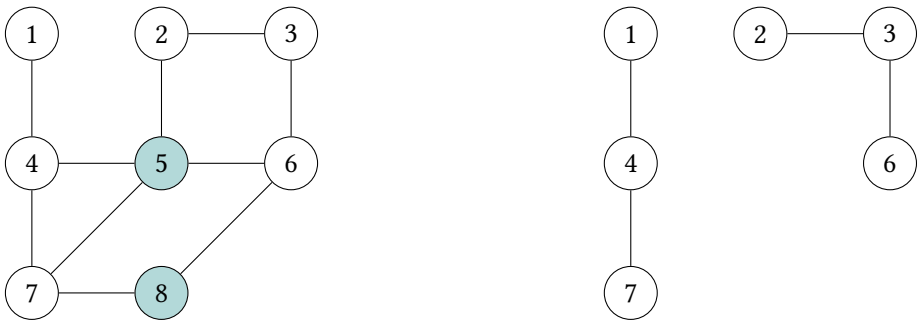
2.2 Graph Separators

A vertex separator (or simply separator) of a graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ whose removal disconnects the graph into two or more components. More formally, the subgraph induced by $V \setminus S$, denoted $G[V \setminus S]$, is disconnected. For algorithmic applications, particularly divide-and-conquer strategies, balanced separators are crucial.

Let $V_1, \ldots, V_k$ be the vertex sets corresponding to the connected components of the subgraph $G[V \setminus S]$. Most often, the removal of such a separator yields exactly two components ($k = 2$), as partitioning the graph into a larger number of components generally demands a larger separator. For a given constant $\alpha \in (0, 1)$, a separator S is termed α-balanced if the size of every resulting component V_i is bounded. Specifically, the condition $|V_i| \leq \alpha|V|$ must hold for all $i \in \{1, \ldots, k\}$. A simple illustration of a balanced separator is shown in Figure 2.1. A common requirement is 2/3-balancedness, meaning each component contains at most 2/3 of the original graph's vertices. Balancedness ensures that recursive applications of the separator lead to subproblems of substantially smaller size, which is essential for the efficiency of algorithms based on this technique.

Furthermore, minimizing the size of the separator S itself is critical for algorithmic performance. The size of the separator is typically evaluated asymptotically as a function of the number of vertices $n = |V|$ e.g. n^β for $\beta \in (0, 1)$.

The concept of *recursive* α-balanced separators extends this idea by ensuring that the property of finding small, balanced separators persists in the resulting subgraphs. Specifically, after removing an α-balanced separator S from G, each induced subgraph $G[V_i]$ (for $i = 1, 2, \ldots, k$) can itself be partitioned using another α-balanced separator of small size.



(a) G with separator $S = \{5, 8\}$ **(b)** $G[V \setminus S]$

Figure 2.1: Example of a well balanced separator in a graph. The vertices 5 and 8 form a balanced separator that disconnects the graph into two components.

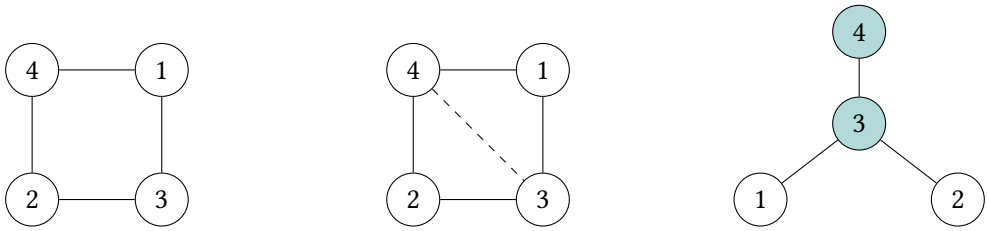
To compute separators, various algorithms can be employed. In this thesis, we primarily utilize InertialFlowCutter [GHUW19]. This algorithm leverages geometric embeddings, often available for road networks, to compute high-quality node orderings efficiently. These node orderings serve as the basis for extracting separators from the graph using the method described below [Blä+25]. Employing this combined approach provides a practical pathway to generate separators for our analysis, adapting the use of InertialFlowCutter's output to suit the specific requirements of this work. However, InertialFlowCutter requires such a geometric embedding as input. For graphs without an embedding, we utilize the KaHIP (Karlsruhe High Quality Partitioning) framework [SS13].

When using InertialFlowCutter, the resulting node ordering is interpreted as an elimination order for the vertices of the graph $G = (V, E)$. Based on this order, a chordal supergraph $G_C = (V, E \cup F)$ is constructed, where F represents the fill-in edges. The chordal supergraph is constructed by processing vertices v according to their rank. Fill-in edges are added such that for each vertex v, all its neighbors with a rank greater than $rank(v)$ form a clique. An efficient implementation connects the neighbor u_{min} with the minimum rank among those

where $rank(u_{min}) > rank(v)$ to all other neighbors w also satisfying $rank(w) > rank(v)$. This suffices because the responsibility for adding edges between the remaining pairs of these higher-ranked neighbors w is effectively delegated to u_{min}.

Afterwards, a tree structure T can be constructed. Each node $v \in V$ selects its parent in the tree as the neighbor u that appears earliest in the elimination order among all neighbors w in G_C with rank$(w) >$ rank(v). If a node has no neighbors later in the order, it becomes the root.

Separators in the original graph G can be derived from this tree structure using a traversal algorithm. The fundamental idea is to identify paths representing non-branching segments of the tree. Starting from a node r (representing the current subgraph), the traversal follows a path $P = (v_1 = r, v_2, \ldots, v_k)$ downwards, where each node v_i $(1 \leq i < k)$ has exactly one child v_{i+1} in the tree. The path ends at node v_k, which is the first node encountered that does not have exactly one child (i.e., it has zero or multiple children). The set of vertices on this path, $S = \{v_1, v_2, \ldots, v_k\}$, forms a separator. Its size is k, the number of nodes on the path. The traversal algorithm continues recursively into these subtrees. An overview of this process is illustrated in Figure 2.2.



(a) Original graph G

(b) Chordal supergraph based on the order

(c) Elimination tree T, Separator {4,3} highlighted

Figure 2.2: Example Process of deriving a separator from a node order. Node labels in indicate their rank in the node order.

To ensure separators yield balanced partitions, the extraction process is refined using a 'significance threshold' based on relative subgraph size. Child nodes are only considered 'significant' if the subgraph they represent exceeds this threshold (e.g., contains at least 5% of the nodes in the parent's subgraph). When tracing a potential separator path $P = (v_1, \ldots, v_k)$ downwards, the path extends from v_i to v_{i+1} only if v_{i+1} is the *single* significant child of v_i. The path $S = \{v_1, \ldots, v_k\}$ is finalized as the separator upon reaching the first node v_k that possesses *two or more* significant children. This ensures separators correspond to meaningful branches in the tree structure concerning substantial graph parts. Figure 2.3 illustrates an example of this process.

2.3 Customizable Contraction Hierarchies

Efficiently computing shortest paths in large graphs, such as continental road networks, is a fundamental problem. While Dijkstra's algorithm provides exact solutions for single-source shortest paths, its performance can be insufficient for real-time applications on large datasets. For instance, executing Dijkstra's algorithm on a graph representing the European road network can take over a second. To accelerate query performance, many algorithms employ

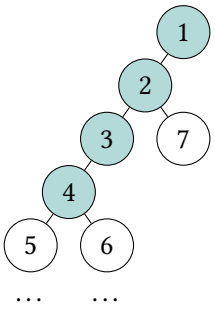


Figure 2.3: Separator identification with a significance threshold. The path extends downwards from node 1. At node 2, child 7 represents an insignificant subgraph (below the threshold), so the traversal continues via the single significant child path towards node 3. Node 4 is the first node encountered with two children (5 and 6) that both represent significant subgraphs. Therefore, the process stops here, and the identified separator is S = {1, 2, 3, 4}.

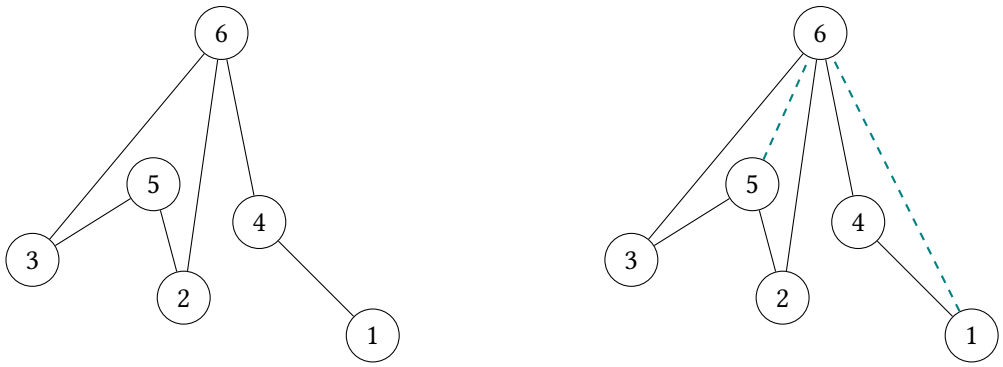
a two-phase approach: an initial precomputation phase followed by a query phase. This precomputation step processes the graph structure and edge weights to generate auxiliary data structures that enable faster subsequent queries.

However, edge weights in real-world networks, particularly road networks, are often dynamic due to factors like traffic congestion. Standard two-phase approaches typically require re-running the entire, often time-consuming, precomputation phase whenever edge weights change. To address this limitation, three-phase approaches have been developed [DGPW11], separating the process into precomputation, customization, and query phases. The initial precomputation relies only on the graph's topology (nodes and edges), which is assumed to be relatively static. The second phase, customization, quickly incorporates the current edge weights into the precomputed structures. Finally, the query phase uses the customized data structures to answer shortest path requests rapidly.

Customizable Contraction Hierarchies (CCH) represent a prominent and effective three-phase route planning technique [DSW16]. CCH enables fast customization, allowing adaptation to frequently changing edge weights, making it suitable for dynamic scenarios. The core idea underpinning CCH involves strategically inserting shortcut edges into the graph, analogous to the concept used in the original Contraction Hierarchies (CH) algorithm [GSSD08]. These shortcuts bypass sequences of original edges, effectively contracting the graph and speeding up queries. The efficiency of the CCH precomputation, particularly the node ordering it employs, can leverage the existence of small separators. We will now give a quick overview of the CCH algorithm.

Precomputation The CCH precomputation phase introduces shortcut edges based on a given vertex order. These shortcuts effectively bypass sections of the graph, allowing algorithms to skip over entire subgraphs, unless the target node resides within such a subgraph. Furthermore, the specific process of inserting shortcuts based on the contraction order guarantees that any shortest path in the original graph corresponds to an 'up-down' path in the hierarchy defined by the vertex ranks [GSSD08]. An 'up-down' path consists of a sequence of edges leading to vertices with increasing ranks (the 'up' segment), followed by a sequence of edges leading to vertices with decreasing ranks (the 'down' segment). This property enables efficient bidirectional search by restricting exploration to higher-ranked neighbors.

This order is defined by a bijection $\pi : \{1, \ldots, n\} \to V$, where $n = |V|$. We will call the inverse of this order rank : $V \to \{1, \ldots, n\}$, which assigns each vertex its position in the order. The core process involves iteratively contracting vertices in order of their rank, starting from rank 1 up to n. Contracting a vertex v_i involves removing it and its incident edges from the current graph representation. For every pair of (higher ranked) neighbors $u, w \in N(v_i)$, a shortcut edge (u, w) is introduced. Resulting multi-edges are simplified. We call the resulting graph $G_C = (V, E_C)$, where $E_C = E \cup F$ and F represents the set of shortcut edges. The contraction process is illustrated in Figure 2.4.



(a) Input graph. Already converted to be undirected and simple.

(b) Graph after precomputation, new shortcut edges are shown in teal.

Figure 2.4: Example of the CCH precomputation step. Nodes are named and positioned based on their rank.

A primary objective when selecting the vertex order is to minimize the number of shortcut edges introduced during the contraction process. Minimizing shortcuts is beneficial for both storage and query efficiency [DSW16]. However, solely minimizing the number of added shortcuts may not be sufficient in all cases. Different heuristics for selecting the contraction order exist.

Nested Dissection One method for computing good vertex orders are Nested Dissections. The process begins by identifying a small, balanced separator in the graph. Nodes within this separator are conceptually removed, partitioning the graph into smaller components. These separator nodes are designated as high-rank nodes in the hierarchy and are consequently placed towards the end of the final node ordering. This procedure is then applied recursively to the remaining components. Figure 2.5 provides a visual representation of this recursive partitioning strategy.

Customization Customization assigns the current metric's weights to the original edges within the CCH supergraph G_C and initializes shortcut edge weights to infinity. Following this initialization, edge weights are systematically updated to ensure the triangle inequality holds throughout G_C.

To achieve this, the concept of a lower triangle is employed. Given an edge $\{x, y\} \in E_C$, a lower triangle is formed by the vertices $\{x, y, z\}$ if the edges $\{z, x\}$ and $\{z, y\}$ also exist, and $rank(z) < \min\{rank(x), rank(y)\}$. The customization algorithm iterates through the vertices of the graph in ascending order of their precomputed rank. For each vertex x, it considers all upward edges $\{x, y\}$ in the graph, where y is a neighbor of x and $rank(y) >$

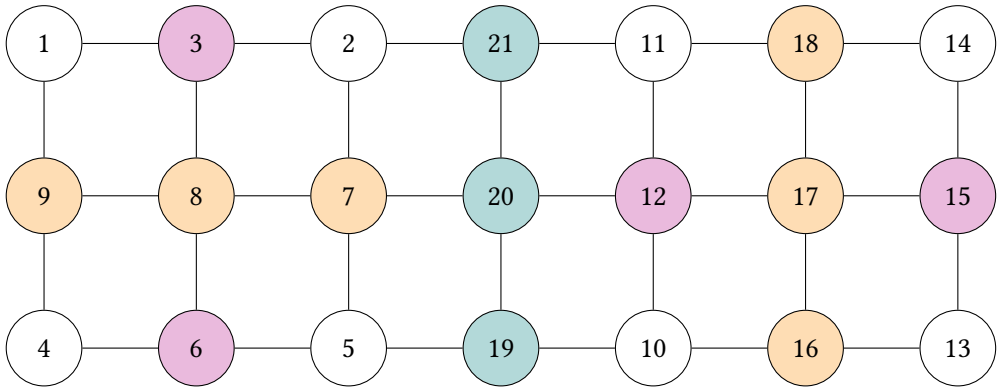


Figure 2.5: Example of a Nested Dissection. The top level separator is shown in teal, the second level in orange and the third level in purple. The nodes are named according to their rank in the resulting order.

$rank(x)$. For every such edge $\{x, y\}$ we determine all lower triangles $\{x, y, z\}$. If the path through z offers a shorter connection, the weight of the edge $\{x, y\}$ is updated to this smaller value: $w(x, y) \leftarrow \min\{w(x, y), w(x, z) + w(z, y)\}$. The detailed procedure is outlined in the pseudocode presented in Algorithm 2.1. An illustration of the customization process is provided in Figure 2.4.

Note that the outlined algorithm only considers undirected edge weights, the algorithm can be extendend to directed edge weights. Details can be found in [DSW16].

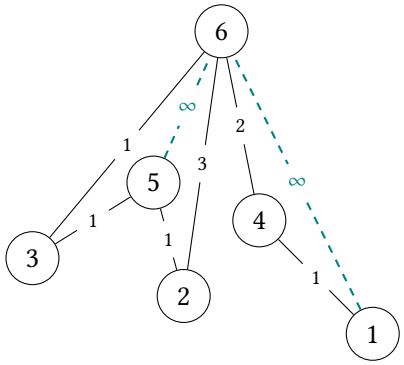
Algorithm 2.1: CCH Customization

Input: $G_C = (V, E_C)$, node ordering π, edge weights w
Output: Customized CCH graph

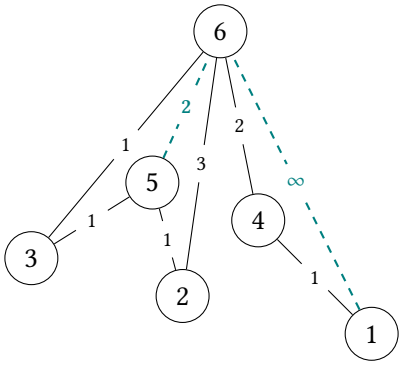
```
1 forall x in V in ascending order of rank do
2     forall upward edges {x, y} in E_C do
3         forall lower triangles {x, y, z} associated with {x, y} do
4             w(x, y) ⟵ min{w(x, y), w(x, z) + w(z, y)}
```

Query To answer a shortest path query between a source node s and a target node t, the algorithm utilizes an structure known as the elimination tree. The elimination tree is defined on the nodes of G_C. The parent of a node v in the elimination tree is the neighbor p of v in the CCH graph that has the lowest rank among all neighbors with a rank strictly greater than the rank of v. Figure 2.7 illustrates the elimination tree for the example graph shown in Figure 2.4. The query algorithm performs a bidirectional search upwards in this elimination tree, starting from s and t.

The core query process operates iteratively. Let u_s and u_t be the current nodes in the upward search originating from s and t, respectively; initially, $u_s = s$ and $u_t = t$. The algorithm proceeds until the root of the elimination tree is reached. In each step, the ranks of the current nodes u_s and u_t are compared. If u_s has a smaller rank than u_t, the algorithm relaxes all outgoing edges $\{u_s, v_i\}$ present in G_C. Subsequently, u_s is updated to become its parent node in the elimination tree. Otherwise (if u_t has a rank less than or equal to that of



(a) Graph after precomputation. Weights are added to the edges. Shortcuts get weight ∞.



(b) Graph after customization. The shortcut edge $\{5, 6\}$ is updated to weight 2.

Figure 2.6: Example of the CCH customization step.

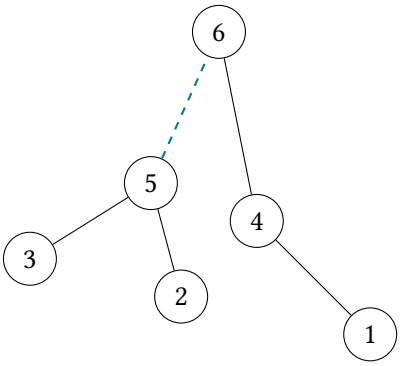


Figure 2.7: Elimination tree for the example graph in Figure 2.4.

u_s), the algorithm relaxes all outgoing edges $\{u_t, v_i\}$ existing in the CCH graph. Following the relaxation step, u_t is updated to its parent in the elimination tree. This process continues, effectively exploring paths upwards towards higher-ranked nodes. The correctness of this query algorithm for computing shortest path distances has been established; a detailed proof, which is beyond the scope of this thesis, can be found in [DSW16].

Complexity The size of the separators found significantly impacts the efficiency of CCH queries. CCH queries restrict exploration to edges leading towards higher-ranked nodes (upward edges). Consider the separator identified at the highest level of the recursion, which contains approximately n^β nodes. When a query initiates within a component defined by this separator, nodes located in other components cannot be reached without traversing downwards through a separator node, violating the upward search constraint. This containment effect applies recursively within the sub-components generated during the nested dissection. Let α denote the balance factor. The sub-components at recursion level i consequently have size at most $\alpha^i \cdot n$. Analyzing the total bound of the search space involves summing these separator sizes across the finite levels i of the recursion. This sum can be bounded by approximating it with the corresponding infinite geometric series [BCRW16]:

$$
\begin{aligned}
&\sum_{i=0}^{\infty} (\alpha^i \cdot n)^\beta \\
&= n^\beta \cdot \sum_{i=0}^{\infty} \alpha^{i \cdot \beta} \\
&= n^\beta \cdot \frac{1}{1 - \alpha^\beta} && \text{Geometric series, since } \alpha \in (0, 1) \\
&\in \mathcal{O}\left(n^\beta\right)
\end{aligned}
$$

This analysis demonstrates that the total search space size explored during a CCH query is bounded by $\mathcal{O}(n^\beta)$, under the assumption that small separators can be found recursively. Note that we do not have worst case guarantees for the size of the separators in real road networks, but we can expect them to be small in practice. Thus, the performance of the CCH algorithm is directly linked to the ability to find small separators.

3 Approach

Add a short introduction to the chapter.

3.1 Empirical Analysis of Separator Scaling in Road Networks

To empirically investigate the relationship between graph size and separator size in road networks, we analyze data derived from a road network graph of Europe [PTV09]. A crucial first step in our research, addressed by this analysis, is to empirically validate whether road networks exhibit separators scaling near $\mathcal{O}(n^{1/3})$, as suggested by prior work [DSW16], as opposed to an alternative, like $\mathcal{O}(n^{1/2})$, potentially appearing smaller due to a low constant factor. Figure Figure 3.1 plots the size of separators against the size of the corresponding subgraphs from which they are computed. Each data point (x, y) in this figure signifies that a subgraph containing x nodes possesses a separator of size y. The dataset contains separators of subgraphs generated by a nested dissection, computing separators first for the original graph and then for the subgraphs induced at each level.

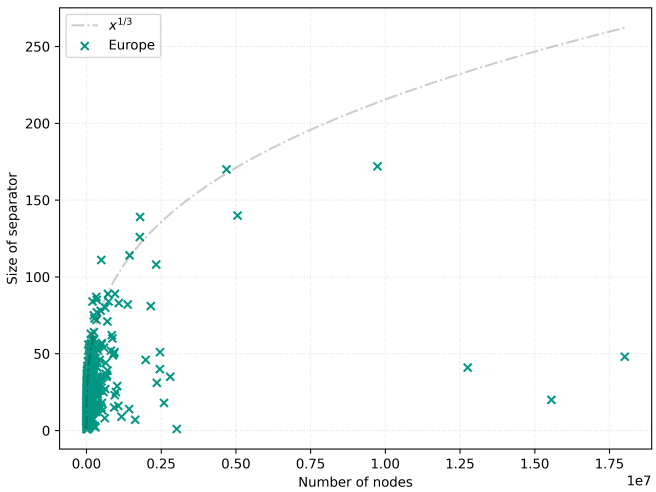


Figure 3.1: Empirical separator size versus subgraph size for the Europe road network. Each point represents a subgraph and its corresponding separator size.

Initial observations reveal outliers, particularly for very large subgraphs corresponding to continental or country scales. Specifically, analysis of the top-level separator structure for the Europe graph shows that the Scandinavian peninsula can be disconnected via separators significantly smaller than the general trend would suggest, due to specific geographic bottlenecks. This can be seen in Figure 3.2. Such outliers at the largest scales appear to be heavily influenced by macroscopic geographic features rather than intrinsic network structure

representative of typical road networks. Consequently, these data points may not accurately reflect the general separator properties inherent in the finer structure of the road network graph. To mitigate the influence of these large-scale geographical artifacts and focus on more representative structural properties, our analysis primarily considers subgraphs with fewer than 10,000,000 nodes.

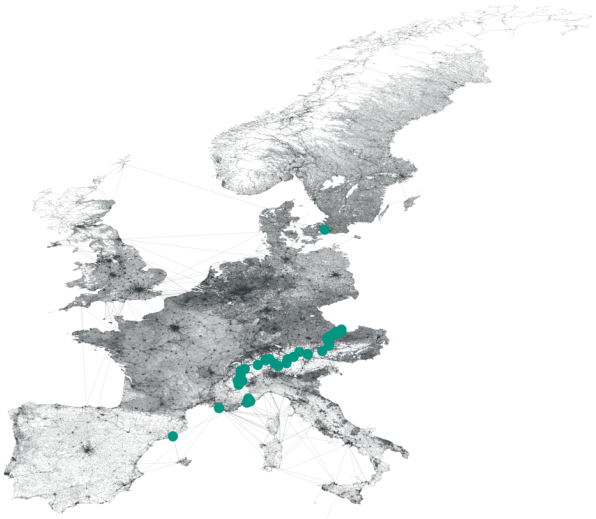
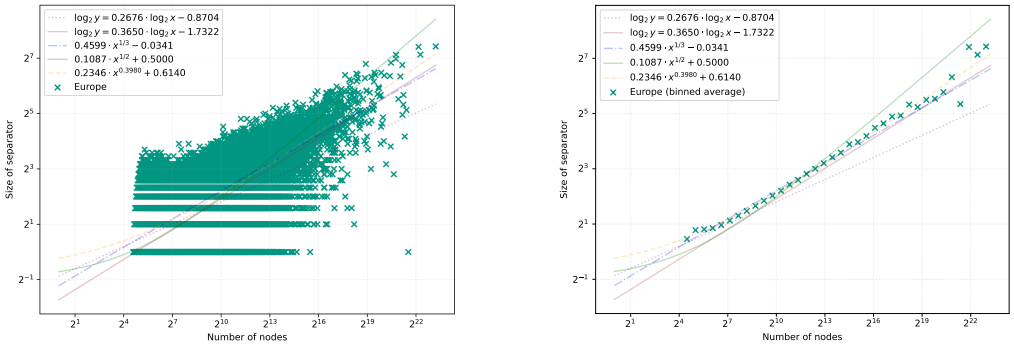


Figure 3.2: Illustration of a geographically influenced outlier at the continental scale: removal of a few nodes disconnects the whole Scandinavian peninsula.

For enhanced visibility, particularly concerning the numerous data points corresponding to smaller subgraphs, and to avoid overrepresentation of larger subgraphs, we will also present data a log-log scale. This logarithmic scaling offers the additional advantage that a polynomial relationship between separator size y and subgraph size x, such as $y \propto x^c$, manifests as a linear trend in the log-log plot, facilitating the identification of potential power-law dependencies. Furthermore, to improve the interpretability of the visualization and emphasize the underlying trend over individual fluctuations or outliers present at various scales, the data points are aggregated into bins. Let b be the number of bins chosen for the aggregation. Let $x_{\max}$ denote the maximum observed subgraph size, assuming $x_{\max} > 0$. A data point (x, y) is assigned to the bin with index $\lfloor \frac{x \cdot b}{x_{\max}} \rfloor$. After assigning all points to their respective bins, a single representative point is computed for each non-empty bin. This representative point $(\overline{x}_i, \overline{y}_i)$ for bin i is determined by calculating the arithmetic mean of the x coordinates and the arithmetic mean of the y coordinates of all data points (x, y) assigned to bin i. A primary consideration for using the arithmetic mean was the nature of subsequent analysis steps, such as curve fitting. While methods like box plots offer detailed distributional insights, they do not provide the single-point representation required for these analyses. Furthermore, the choice of the mean over the median, which is known for its robustness to outliers, was deliberate in this context. Outliers within bins are not necessarily disregarded as noise but are considered potentially informative data points reflecting relevant variations. Supporting this decision are the mean's straightforward calculation, computational efficiency, and clear interpretation as the centroid or center of mass of the points within the bin.

When constructing the log-log plot, this binning procedure is applied to the logarithmically transformed data. Figure 3.3b illustrates the binned data points on logarithmic axes.



(a) Non-binned separator sizes.

(b) Binned average separator sizes.

Figure 3.3: Separator sizes of Europe on logarithmic axes (subgraphs < 10M nodes).

To quantify the relationship between separator size y and subgraph size x, we performed statistical curve fitting on the empirical data. A preliminary linear regression applied to the logarithmically transformed data across all points yields a fitted line $\log y \approx 0.2676 \log x - 0.8704$. This initial fit suggests a scaling behavior close to $\mathcal{O}(x^{0.2676})$, which is seemingly better than the hypothesized $\mathcal{O}(x^{1/3})$. However, this result might be skewed by the large number of small subgraphs, particularly those with trivial separators, which may not fully represent the scaling trend at larger sizes. To test this hypothesis, we performed a second regression exclusively on data points corresponding to subgraphs with more than $2^8 = 256$ vertices. This analysis yielded the relationship $\log y \approx 0.3650 \log x - 1.7322$.

We report these coefficients to four decimal places as is conventional in scientific reporting. However, it is important to acknowledge that this level of precision might overstate the certainty of the fit, given the inherent noise in empirical measurements derived from complex graph algorithms. Minor variations in experimental conditions, such as changing the random seed used in the nested dissection algorithm, can cause slight fluctuations in these fitted parameters. Therefore, the reported values should be interpreted as indicative of the general trend rather than exact constants.

We also explore direct non-linear curve fitting to the original data points using several functional forms. Fitting $y = a \cdot x^{1/3} + b$ results in $y \approx 0.4599 \cdot x^{1/3} - 0.0341$ with an R^2 value of 0.4724. Fitting $y = a \cdot x^{1/2} + b$ yields $y \approx 0.1087 \cdot x^{1/2} + 0.5000$, but with a very low R^2 value of 0.0873, suggesting that a square-root dependency is unlikely. A more general power-law fit $y = a \cdot x^c + b$ results in $y \approx 0.2346 \cdot x^{0.3980} + 0.6140$ with an R^2 value of 0.4832. While these direct fits indicate a scaling exponent potentially closer to 0.4 than $1/3$ or $1/2$, the overall R^2 scores remain relatively low, indicating only a moderate goodness of fit. The above mentioned curves are visualized in Figure 3.3. The fitted lines displayed in Figure 3.3b are those computed from the non-binned data and shown previously in Figure 3.3. They are reproduced here to facilitate comparison with the binned averages, although visual alignment may differ due to the underlying averaging represented by the bins.

Observations from the binned log-log plot in Figure 3.3b suggest that the relationship between $\log y$ and $\log x$ is most consistent within a specific range of subgraph sizes. The data appears particularly stable for subgraph sizes x between approximately 2^7 and 2^{18} nodes. For

subgraphs smaller than 2^7 nodes, a slight upward trend is discernible, potentially reflecting behavior closer to grid-like structures at very small scales. For subgraphs larger than 2^{18} nodes, the increased scatter might be attributed to the limited number of data points available for aggregation in these higher size ranges.

To obtain a more robust estimate of the scaling exponent, we focus our analysis on the data within this more stable range ($2^7 \leq x \leq 2^{18}$). Furthermore, performing the linear regression on the binned data points, as plotted in Figure 3.3b, offers two key advantages. Firstly, binning averages out the influence of individual outliers within each bin. Secondly, it gives more equal weight to different orders of magnitude in subgraph size, mitigating the numerical dominance of the numerous small subgraphs in the dataset.

Applying linear regression to the mean coordinates of the bins within the selected range on the log-log scale yields the fitted line:

$$\log y \approx 0.3702 \log x - 1.5512 \qquad \Longleftrightarrow \qquad y \approx 0.3411 \cdot x^{0.3702}$$

This fit demonstrates an exceptionally high coefficient of determination ($R^2 \approx 0.9994$), indicating that the linear model explains almost all the variance in the binned log-transformed data. The statistical significance of the slope is confirmed by an extremely low p-value ($p \approx 1.64 \times 10^{-20}$).

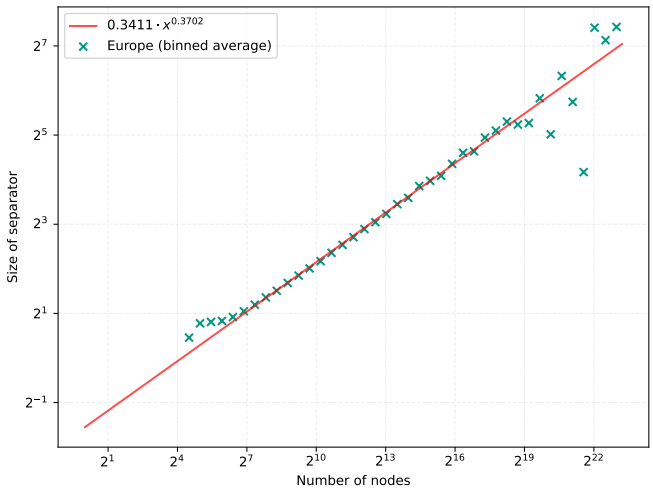


Figure 3.4: Linear regression fit to the binned data of separator size versus subgraph size, plotted on logarithmic axes. The regression considers only bins corresponding to subgraph sizes (x) in the range $2^7 \leq x \leq 2^{18}$.

Visually, this line provides an excellent fit to the binned data points within the considered range, as illustrated in Figure 3.4.

The slope of 0.3702 in the log-log regression corresponds to the exponent in the power-law relationship $y \propto x^c$. Based on the high quality of the fit ($R^2 \approx 0.9994$) and the statistical significance of the result, we can state with relatively high confidence that the observed scaling behavior is close to, yet slightly above, $\mathcal{O}(n^{1/3})$.

3.2 Planarity

Road networks can be modeled as nearly planar graphs, meaning they permit embedding in the plane with a limited number of edge crossings. Empirical evidence suggests that the number of such crossings in real-world road networks is typically in $\Theta(n^{1/2})$, where n represents the number of vertices [EG08]. It is a well-known result in graph theory that planar graphs admit $\frac{2}{3}$-balanced separators of size $\mathcal{O}(n^{1/2})$ [LT79].

A relevant inquiry is whether the near-planarity of road networks is a critical feature that influences their structural properties, or if the occasional non-planar elements are merely incidental and do not substantially affect the network's overall characteristics. This prompts the question of how the separator sizes of road networks are affected when they are transformed into strictly planar graphs, for instance, by altering edges to eliminate crossings.

To obtain a planar representation, we begin with the existing graph structure where vertices possess associated geometric coordinates. Each edge in this graph is interpreted as the straight line segment connecting the coordinates of its incident vertices. The algorithm then identifies all geometric intersection points occurring between these line segments. A new vertex is introduced into the graph at the precise coordinates of each detected intersection point, provided this point does not coincide with the coordinates of an existing endpoint of the intersecting segments. Subsequently, any original edge that contains one or more such intersection points along its segment is removed. It is replaced by a sequence of new, shorter edges that connect the original endpoints and the newly created intersection vertices in their linear order along the original segment. This process transforms the initial graph into a planar graph embedding by explicitly representing edge crossings as vertices. For efficient execution, we utilize a spatial index that stores the bounding boxes of all edges. Under the assumption of short edges, this structure enables rapid identification of potential intersections by querying overlapping bounding boxes, followed by verification of actual crossings. While other algorithms exist, such as the Bentley-Ottmann algorithm [BO79], which is designed for general segment crossings, or a linear-time algorithm (e.g., as described in [EGS10]), which is tailored for graph structures with a sublinear number of edge crossings, we choose this spatial index-based approach due to its simplicity and ease of implementation, especially since performance is not a critical concern in this context. Given that a single edge may intersect multiple other edges, we sort the intersection points along each edge and introduce new edges accordingly. Pseudo-code for this planarization algorithm is provided in Algorithm 3.1.

We apply this planarization method to real-world road networks. The Karlsruhe network, with approximately 120,000 nodes, has around 2,500 intersections. The Germany network, comprising about 6 million nodes, has approximately 100,000 intersections. Finally the Europe network, with around 18 million nodes, contains about 300,000 intersections. These numbers exceed the $\mathcal{O}(n^{1/2})$ intersection counts reported in prior studies but remain within a similar magnitude [EG08]. The differences could be explained by the unoptimal linear assumption of edges and might be mitigated by using a more modeled road network like OpenStreetMap.

Analysis of separator sizes shows minimal variation post-planarization. We identify $\frac{2}{3}$-balanced separators of size approximately $\mathcal{O}(n^{1/3})$, aligning with the values from non planar graphs. A comparison of the separator sizes in the planar and non-planar versions of the Germany network is depicted in Figure 3.5.

Our findings indicate that separators in non-planar road networks closely resemble those in their planarized versions. Just treating the separator of a non-planar road network graph as a separator in the planarized graph is not always sufficient. We therefor traverse a single arm of the nested dissection and checked if the separators of the subgraphs would also hold

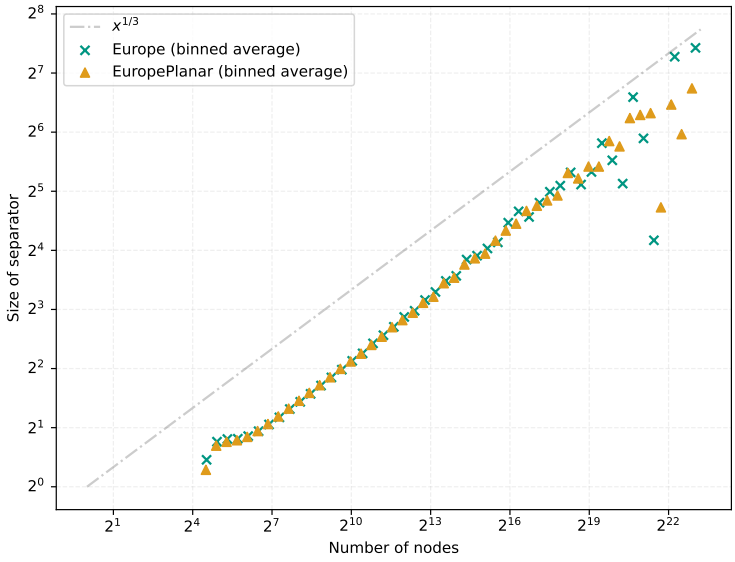


Figure 3.5: Comparison of separator sizes in the European road network: planar vs. non-planar.

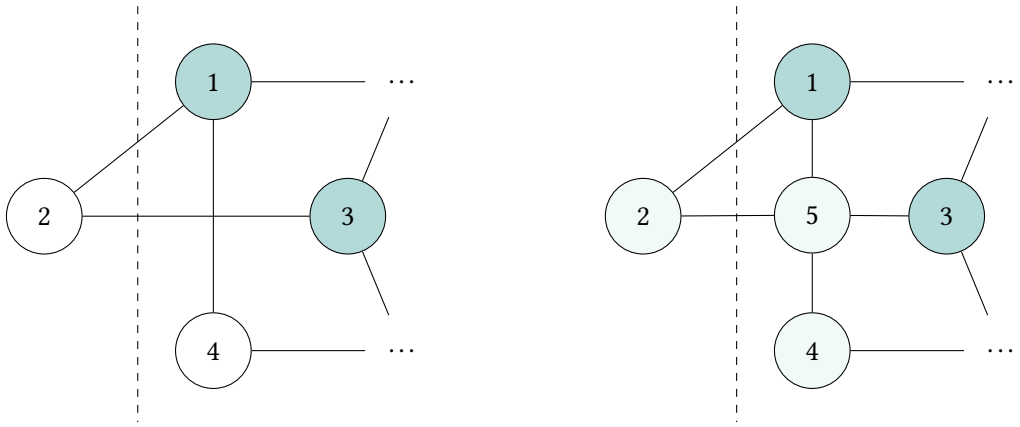
in the planarized graph. Around one third of the separators could be applied directly to the planarized graph. With it being more probable for smaller graphs to be valid separators in the planarized graph. We assume this is due to the fact that smaller graphs are less likely to contain a shape like shown in Figure 3.7. And just a single new edge crossing can invalidate the separator. In cases where the separator of the non-planar graph is not a valid separator in the planarized graph, we can often add just a few nodes to the separator to make it valid. This can be seen in Figure 3.8, which depicts a non-planar separator extended to be a separator in the planarized Karlsruhe network. Note that this paragraph does not make claims if we can in general find better separators in the planarized graph.

(a) Separator in non-planar graph

(b) Better separator in planarized graph

Figure 3.6: Example of a separator, where a better separator can be found in the planarized graph.

These findings highlight that the near-planar structure of road networks has minimal impact on separator size, suggesting that such networks can typically be analyzed as planar graphs.



(a) Separator in non-planar graph

(b) The separator of the original graph is not a separator in the planarized graph.

Figure 3.7: Example where the separator of the original graph is not a separator in the planarized graph.

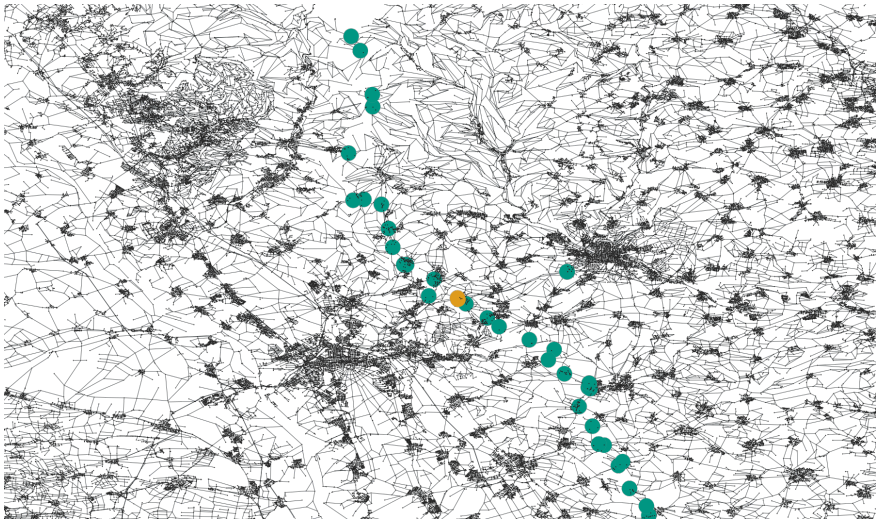


Figure 3.8: Visualization of a potential top-level separator for the road network of Karlsruhe. Vertices colored teal represent the separator nodes identified within the original, non-planar graph representation. Orange vertices (two almost overlapping ones) indicate the additional nodes required to establish a valid separator for the planarized version of the Karlsruhe road network. The single teal vertex located on the right side of the figure corresponds to a highway intersection. Although seemingly isolated in this view, its inclusion in the separator is necessary.

Algorithm 3.1: Simple planarization algorithm

Input: Non-planar graph $G = (V, E, pos)$.
Output: Planarized version of G.

```
1  spatial_index ⟵ load(bounding_boxes(E))
2  crossings ⟵ {}
3  forall e in E do
4      forall candidates c in spatial_index.query(e) do
5          if c intersects e then
6              crossings[e].append(c)
7              crossings[c].append(e)
8  forall (e, crossed) in crossings do
9      G.remove(e)
10     vertices ⟵ get_intersection_vertices(e, crossed)
11     sort vertices along e
12     add_new_edges(e, vertices)
```

4 Synthetic Graph Generation for Feature Isolation

To better understand the underlying reasons for the small separators observed in road networks, we investigate the influence of specific graph properties in isolation. This chapter details our approach to generating synthetic graph classes that exhibit selected features characteristic of road networks, such as low average degree, specific degree distributions, or properties related to planarity and locality. By analyzing the separator sizes within these synthetic graphs, we aim to determine which properties, or combinations thereof, are crucial for enabling small separators.

4.1 Degree Distribution

Our initial focus is on isolating the effect of the degree distribution. Road networks are known to be sparse graphs. The specific PTV Europe road network dataset [PTV09] utilized in our experiments exhibits an average vertex degree of approximately 2.5. A detailed visualization of the degree distribution for this network is provided in Figure 4.1.

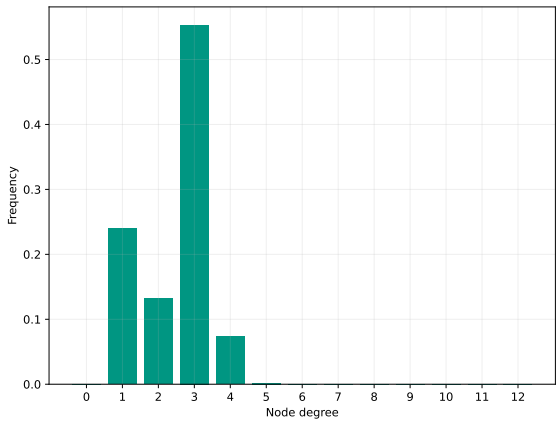


Figure 4.1: Degree distribution of the PTV Europe road network. The x-axis represents the vertex degree, and the y-axis shows the fraction of vertices with that degree. A single node with degree 12 has the highest degree.

To examine whether this low average degree alone is sufficient to yield small separators, we generate connected random graphs matching this average degree. The generation process involves two main steps. First, a random spanning tree is created for a given set of n vertices using the algorithm described by Broder [Bro89]. This algorithm performs a random walk starting from an arbitrary vertex on the complete graph of n vertices. An edge becomes

part of the spanning tree the first time a vertex is discovered via that edge during the walk. The process continues until all vertices are visited, resulting in a uniformly sampled random spanning tree in expected time $\mathcal{O}(n \log n)$.

It is noteworthy that random trees generated in this manner exhibit properties distinct from those of road networks. For instance, the diameter of such random trees is known to be in $\mathcal{O}(n^{1/2})$ [CK87]. This differs from empirical observations on road networks, such as subgraphs from the nested dissection of the Karlsruhe graph, where the diameter appears smaller, estimated empirically as $\mathcal{O}(n^{0.3683})$. This observed diameter growth is visualized in Figure 4.2.

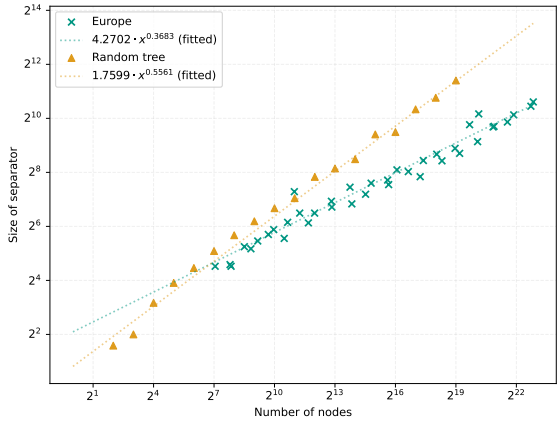


Figure 4.2: Empirical diameter growth observed in the Karlsruhe road network subgraph. The plot shows the diameter (y-axis) as a function of the number of nodes n (x-axis, log scale).

Following the generation of the initial random spanning tree, we proceed to the second step: adding edges randomly between pairs of non-adjacent vertices. This edge addition continues until the target average degree of 2.5 is reached for the entire graph. The resulting graphs, by construction, lack the inherent locality often present in road networks. A consequence of adding these random edges is a notable decrease in the graph diameter. For example, graphs generated with one million nodes using this method exhibit diameters of approximately 40.

These synthetic graphs do not replicate the small separator sizes observed in real-world road networks. Experiments yielded large top-level separators, whose sizes scaled approximately linearly with the graph size n, i.e., as $\mathcal{O}(n)$. However, separators found during subsequent recursive partitioning behaved differently: their sizes were significantly smaller than this linear trend would predict for the corresponding subgraph sizes. We attribute this deviation to structural changes in the subgraphs induced by the partitioning process. Specifically, the large top-level separators often remove a significant fraction of the non-tree edges that were originally added to achieve the target average degree of approximately 2.5 in the parent graph. Consequently, the resulting subgraphs become sparser—our observations showed a decrease in average degree to approximately 2.2—and increasingly dominated by their underlying tree structure. This shift means these subgraphs effectively belong to a different, more tree-like graph class than initially generated, which naturally leads to smaller relative separator sizes. The separator scaling is illustrated in Figure 4.3.

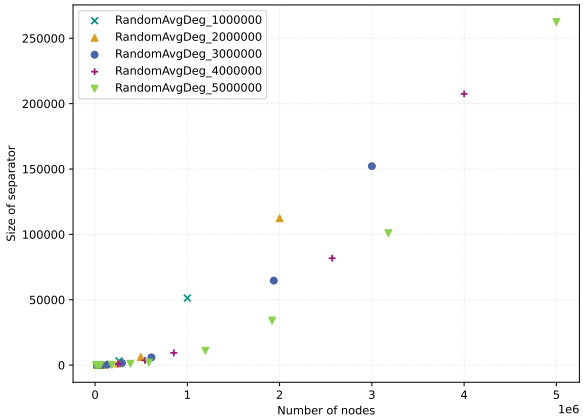


Figure 4.3: Separator sizes observed in random graphs generated with an average degree matching the PTV Europe road network (≈ 2.5).

We also conduct experiments generating graphs that match not just the average degree, but the specific degree distribution of the reference road network. This involves a similar process of starting with a random tree and adding edges randomly. However, edge addition is constrained: an edge (u, v) is added only if adding it does not cause either vertex u or v to exceed the count allocated for their respective degrees according to the target distribution sampled from the road network. The results obtained from this refined approach are largely similar to those using only the average degree constraint. It is important to note, however, that precisely replicating the target degree distribution using this generation process presents challenges. While the maximum degree observed in the generated graphs generally approximately aligns with that of the reference road network, discrepancies arise in the distribution concerning nodes with higher degrees. Specifically, the underlying structure of the initial random spanning tree tends to produce more vertices with high degrees compared to the road network. Approximately 1.8% of vertices in the random tree had a degree of 5 or greater, substantially more than the 0.2% observed in the actual Europe dataset [PTV09]. Therefore, the process achieves an approximation of the target distribution rather than an exact match. We contend, however, that this approximation is sufficiently close for the purposes of this investigation, and argue that the relatively small fraction of vertices whose degrees exceed the target constraints are unlikely to fundamentally alter the observed separator scaling behavior for this class of graphs.

The generated graphs continue to exhibit large separators, reinforcing the conclusion drawn from the average-degree matching experiments. These findings suggest that the degree distribution alone is likely insufficient to explain the empirically observed small separators. Nevertheless, the low average degree remains an important factor. Intuitively, sparser graphs, possessing fewer edges overall, should generally be easier to partition compared to denser graphs.

4.2 Locality

Building upon the previous experiments focused solely on degree distribution, we maintain the core methodology of first generating a random spanning tree and then augmenting it with additional edges to achieve a desired average degree. The crucial difference in the approach detailed in this section lies in the edge augmentation phase: we introduce constraints intended to promote locality. This modification reflects the inherent spatial structure of road networks, where connections predominantly link geographically proximate locations. Two distinct strategies for introducing locality during edge addition are explored.

Our first approach attempts to simulate locality based on distances within the initial random spanning tree structure. As before, we begin by generating a random spanning tree using Broder's algorithm [Bro89]. Subsequently, edges are added between non-adjacent vertices until a target average degree, chosen as 2.5 to approximate that of road networks, is achieved. To incorporate locality, the probability of adding an edge between two vertices u and v is made dependent on their distance $\text{dist}_T(u, v)$ within the initial spanning tree T. Specifically, we select a random vertex x and choose a second vertex y with a probability related to $f(\text{dist}_T(x, y))$, where f is a decreasing function.

The computation of tree distances $\text{dist}_T(x, y)$ for numerous candidate pairs (x, y) is a critical step in this edge addition phase. Initially, for each randomly selected vertex x and potential neighbor y, we performed a Breadth-First Search (BFS) within the tree T to find $\text{dist}_T(x, y)$. While BFS on a tree is linear in the number of vertices n, repeated invocations for many pairs proved computationally intensive for larger graphs. To accelerate this, we adopted an approach based on Lowest Common Ancestor (LCA) queries. This involved an initial precomputation of an LCA data structure for the tree T, which can be achieved in $\mathcal{O}(n \log n)$ time (e.g., using an Euler tour and sparse table for Range Minimum Queries, enabling $\mathcal{O}(1)$ query time). Once the LCA of two nodes u and v is known, their tree distance is given by $\text{dist}_T(u, v) = \text{depth}(u) + \text{depth}(v) - 2 \cdot \text{depth}(lca(u, v))$. Although a single BFS is also linear, this LCA-based method yielded a significant speedup, due to better parallelization. Nevertheless, even with this optimization, the approach of potentially considering many y candidates for each x (full sampling) remained too slow for large graphs. A possible further refinement, not explored in this work, could involve capping the search for y (e.g., limiting BFS depth or number of hops from x), under the assumption that for rapidly decaying probability functions f, the likelihood of adding edges to very distant nodes becomes negligible.

Experiments using functions such as $f(\text{dist}) = 1/\text{dist}$ result in graphs that still exhibit large separators scaling as $\mathcal{O}(n)$. For subgraphs of the nested dissection we observe again the super-linear decrease in separator size, but the top-level separators remain large. Using moderately decaying functions like $f(\text{dist}) = 1/\text{dist}^2$ leads to separators that scale sub-linearly but they are still significantly larger than $\mathcal{O}(n^{1/3})$, especially the constant factor, where we are one to two orders of magnitude larger than the expected size, despite having the same average degree. Conversely, using rapidly decaying functions like $f(\text{dist}) = 2^{-\text{dist}}$ leads to separators of almost constant size, likely due to the graph structure remaining very close to the initial tree.

While the initial experiments using simple decay functions $f(\text{dist}_T)$ did not yield the desired separator scaling, the concept of incorporating locality based on distances within the existing network structure (represented here by the initial tree) remains potentially insightful. Such a metric might capture aspects beyond mere Euclidean distance, possibly relating to factors like travel time reduction incentives for new connections within the existing network topology. However, our preliminary results indicated a high sensitivity to the choice of the decay function

f, with simple functional forms leading to either overly large or near-constant separator sizes. Identifying a function capable of consistently producing $\mathcal{O}(n^{1/3})$ scaling appeared non-trivial and potentially specific to this particular generative model, lacking immediate theoretical guidance or clear real-world correspondence. Given these initial findings and the decision to prioritize the investigation of other structural properties such as geometric locality, planarity, and explicit hierarchy, we opted not to pursue an exhaustive empirical search for an optimal tree-distance decay function within the scope of this work.

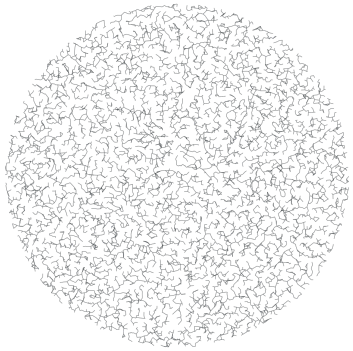
Our second approach incorporates geometric locality directly. The core idea is to construct a graph starting with a basic connected structure and then augment it by adding only edges that connect geometrically close vertices, until a target average degree (e.g., 2.5) is achieved. To establish a meaningful threshold for 'closeness', we relate it to the natural scale derived from the spatial distribution of points. Specifically, we first sample n points uniformly within a defined spatial domain (e.g., a circle). We then compute the Minimum Spanning Tree (MST) of these points using Euclidean distances, via Kruskal's algorithm [Kru56]. The maximum edge length found within this MST, denoted $\ell_{\max}$, serves as our threshold distance, capturing a characteristic length scale of the initial sparse connection of the points. The graph generation then proceeds by initially taking the edges of the MST. Subsequently, additional edges (u, v) between non-adjacent vertices are iteratively added, but crucially, only if their Euclidean distance is less than or equal to the threshold $\ell_{\max}$. This edge addition process continues until the overall graph reaches the target average degree of 2.5. For efficient implementation of the edge addition step, we utilize spatial queries: for a randomly selected vertex u, we query for potential neighbors v within the radius $\ell_{\max}$ and randomly select one to connect to, avoiding multi-edges and self-loops.

Despite incorporating this explicit geometric constraint, the resulting synthetic graphs fail to exhibit the desired small separator sizes characteristic of road networks. To analyze the separator properties of these graphs, we computed recursive separators. Implementations from both InertialFlowCutter and KaHIP were utilized for this purpose. Our experiments using these methods indicate that separators in these geometrically generated graphs scale approximately as $\mathcal{O}(n^{1/2})$. This outcome is visualized in Figure 4.4.

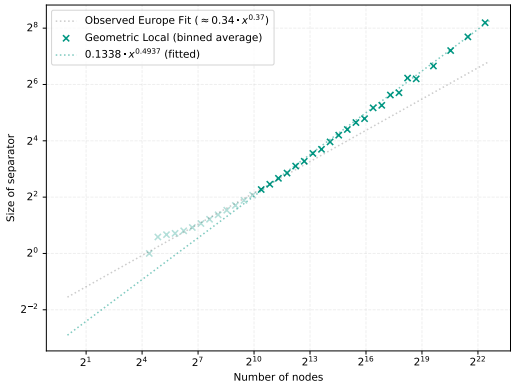
The results from the geometric locality approach suggest that merely combining low average degree with this specific form of locality is insufficient to reproduce the cubic root separator sizes of road networks.

Real-World Implications It is worth considering the practical implications of different asymptotic growth rates for separator sizes within the typical scale of road networks. While $\mathcal{O}(c_1 \cdot n^{1/3})$ and $\mathcal{O}(c_2 \cdot n^{1/2})$, with $c_2 \ll c1$, diverge for large n, the actual separator sizes for graphs up to around 20 million vertices are similar. The performance of algorithms leveraging graph separators in real-world applications may be more influenced by the absolute sizes of separators achievable in practice, compared to a specific scaling model. Performance gains, for example for CCH [DSW16], could potentially be realized even if the separators behave like $c \cdot n^{1/2}$ for a sufficiently small c, as the absolute separator sizes remain manageable for networks of practical relevance.

Initial Deviations in Separator Scaling An interesting characteristic is observable for graphs with small node counts in empirical studies of road networks. As a careful reader may notice in Figure 3.3a, real road networks often exhibit a deviation in separator scaling for graphs containing approximately 2^6 nodes. Figure 4.5 presents histogram plots to better



(a) Visualization of a graph with 10K nodes generated using the geometric locality method.



(b) Separator size scaling for synthetic graphs generated with geometric locality. The fit shown excludes subgraphs with ≤ 1000 vertices due to their disproportionately large separatos.

Figure 4.4: Synthetic graph generation using geometric locality and analysis of separator sizes.

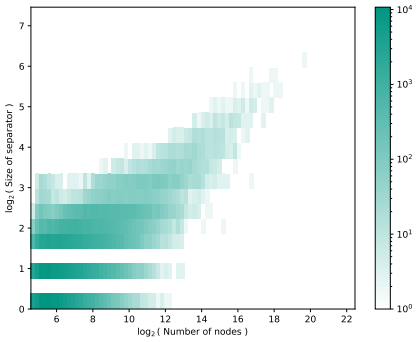
visualize the prevalence of these instances for both real-world data and our synthetic model. Specifically, Figure 4.5a shows this phenomenon for European road network samples. These plots reveal an initial peak in relative separator size for small graphs, followed by a slight decrease, before the data align with the more dominant scaling trend observed for larger graphs.

Our synthetic graphs generated using the geometric locality approach exhibit a similar initial pattern, as can be discerned from the histogram in Figure 4.5b. This initial behavior in the synthetic model appears qualitatively comparable, although slightly more pronounced than in the real network data. This correspondence is noteworthy. It suggests that the geometric locality model, despite failing to replicate the overall asymptotic separator scaling of $\mathcal{O}(n^{1/3})$, may capture certain structural properties relevant at smaller graph scales that are present in actual road networks.

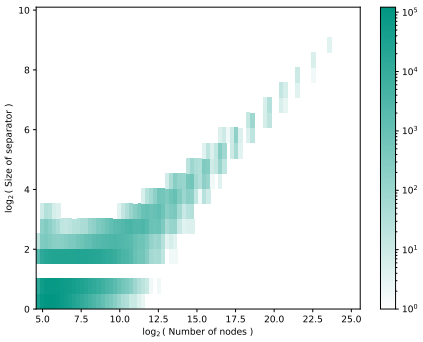
4.3 Planarity

We examine separator properties in two classes of planar graphs: grids and Delaunay triangulations. These serve as fundamental theoretical models for planar structures relevant to the study of road networks due to their near-planarity. In our study, these graph classes are sparsified to achieve a low average degree (approximately 2.5), reflecting the sparsity observed in road networks.

Our initial investigation focuses on grid graphs, a fundamental class known to possess separators scaling as $\mathcal{O}(n^{1/2})$. To align their sparsity with road networks, we generate modified grids with an average degree of approximately 2.5. The generation process starts with a square two-dimensional grid graph. Edges are then removed uniformly at random until the target average degree is reached over the entire graph. Subsequently, we identify and utilize the



(a) Histogram of separator sizes for the European road network.



(b) Histogram of separator sizes for synthetic graphs using geometric locality.

Figure 4.5: Histograms illustrating the distribution of separator sizes, comparing real road networks (left) and the geometric locality model (right). A slight increase in separator size is observed for graphs with $\approx 2^6$ nodes.

largest connected component for analysis. We observe that even without explicit mechanisms to prevent disconnection during edge removal, the largest connected component typically encompasses a large fraction of the initial vertices.

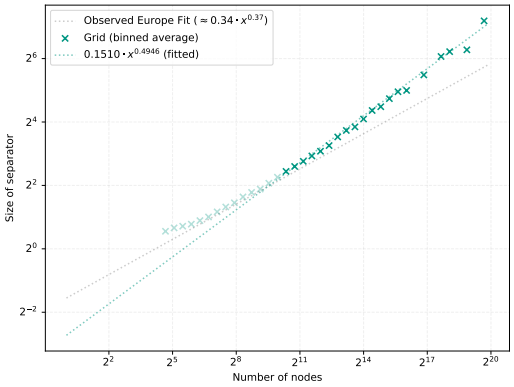
Analysis of these sparse grid graphs reveals separator sizes consistent with the $\mathcal{O}(n^{1/2})$ asymptotic behavior of complete grids. However, the constant factor associated with this scaling appears to be relatively small. Consequently, although the asymptotic limit behavior differs from the $\mathcal{O}(n^{1/3})$ scaling empirically observed for road networks, the absolute separator sizes in these sparse grids are numerically similar to those of road networks for graphs up to typical sizes (e.g., around 20 million nodes). This finding highlights that sparsity, even within a simple planar structure like a grid, can lead to separators that are small in absolute terms for practical graph dimensions. Figure 4.6 illustrates a sample sparse grid and the observed separator scaling.

We extend this investigation to Delaunay triangulations, which can be viewed as a generalization of grids derived from point sets. Our process involves sampling points uniformly at random in a two-dimensional space and computing their Delaunay triangulation. Similar to the grid experiments, we then randomly delete edges from the triangulation until the average degree reaches the target value of 2.5, again analyzing the largest connected component.

As might be expected, the results obtained from these sparse Delaunay graphs are largely analogous to those from sparse grids. The separators exhibit scaling consistent with $\mathcal{O}(n^{1/2})$, characteristic of planar graphs, yet the associated constant factors are small. This again leads to absolute separator sizes that are numerically comparable to those measured in road networks for graphs of relevant sizes. These findings are depicted in Figure 4.7.

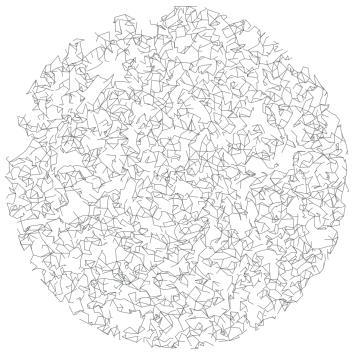
The experiments with sparse grids and Delaunay triangulations suggest that combining geometry with low average degree results in graphs whose practical separator sizes are close to those of road networks, despite differing asymptotic scaling.

(a) Visualization of the largest connected component of a grid graph with ≈ 10K nodes after random edge removal to achieve an average degree of 2.5.

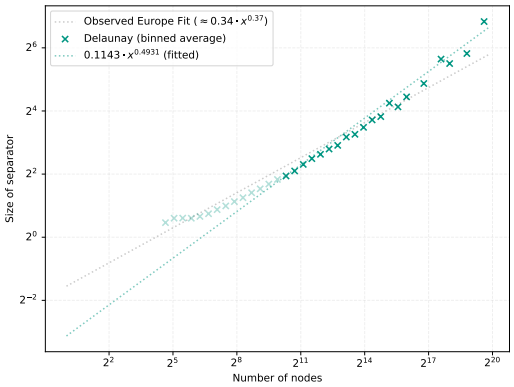


(b) Separator size scaling for sparse grid graphs (average degree 2.5). Separator sizes scale approximately as $\mathcal{O}(n^{1/2})$, but with a small constant factor leading to absolute sizes comparable to road networks at practical scales. The fit shown excludes subgraphs with ≤ 1000 vertices due to their disproportionately large separators.

Figure 4.6: Analysis of sparse grid graphs with average degree 2.5.



(a) Visualization of a sparse Delaunay triangulation with 5K nodes after random edge removal to achieve an average degree of 2.5.



(b) Separator size scaling for sparse Delaunay graphs (average degree 2.5). Separator sizes scale approximately as $\mathcal{O}(n^{1/2})$ with small constant factors, yielding absolute sizes similar to road networks at practical scales. The fit shown excludes subgraphs with ≤ 1000 vertices due to their disproportionately large separatos.

Figure 4.7: Analysis of sparse Delaunay graphs with average degree 2.5.

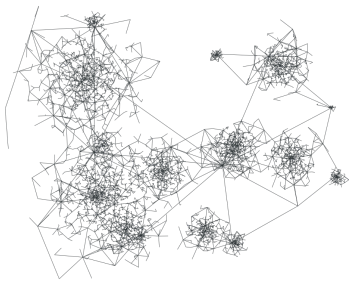
4.4 Highway Dimension

Given that the simpler graph models explored in previous sections do not reproduce the observed $\mathcal{O}(n^{1/3})$ separator scaling of road networks, we turn our attention to more complex generative processes designed to capture other relevant structural properties. One such property is highway dimension, introduced by Abraham et al. [AFGW10]. Intuitively, a graph possesses a small highway dimension if, for every radius $r > 0$, there exists a sparse set of vertices S_r such that every shortest path longer than r intersects S_r. A set is considered sparse if every ball of radius $\mathcal{O}(r)$ contains only a small number of vertices from S_r [AFGW10]. The significance of this property stems from the finding that low highway dimension provides provable performance guarantees for several important route planning algorithms, including REACH [GKW06], Contraction Hierarchies [GSSD08], Highway Hierarchies [SS05], Transit Node Routing [BFSS07], and SHARC [BD10].

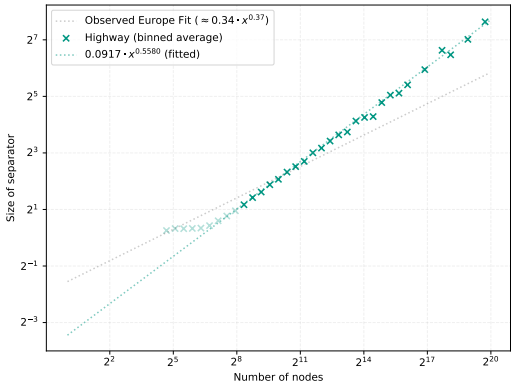
The work introducing highway dimension also proposes a synthetic graph generator (henceforth ABR generator) intended to produce graphs exhibiting this property [AFGW10]. A good explanation of the generation process is provided in [BKMW10], which we summarize here. The generation process operates iteratively. It begins with an empty graph $G = (V, E) = (\emptyset, \emptyset)$ and progressively adds new vertices v_t to V, whose locations in the metric space are chosen randomly. Throughout this process, the generator maintains a series of 2^i-covers, denoted C_i, for each level i where $1 \leq i \leq \log D$, and D represents the diameter of the metric space. A set $C_i \subseteq V$ is a 2^i-cover if any two vertices $u, v \in C_i$ satisfy $d(u, v) \geq 2^i$, and every vertex $u \in V$ is within distance 2^i of some vertex in C_i. When a new vertex v_t is added, the generator identifies the smallest index i such that there exists a vertex $w \in C_i$ with $d(v_t, w) \leq 2^i$. The new vertex v_t is then added to all covers C_j for which $0 \leq j < i$. If no such index i exists, v_t is added to all cover sets C_j. Edges are subsequently added based on these covers and a tuning parameter k. For each cover C_j containing v_t (where $0 \leq j < i$), and for each existing vertex $w \in C_j$, an edge (w, v_t) is added if their distance satisfies $d(w, v_t) \leq k \cdot 2^j$. Furthermore, for each C_j containing v_t where $j < \log D$ and v_t is also present in C_{j+1}, an edge is added connecting v_t to its nearest neighbor within the set C_{j+1}. For our experiments, we adopt parameter settings similar to those used by [BKMW10], setting the diameter $D = 2^{25}$ and the connection parameter $k = \sqrt{2}$.

Bauer et al. also describe an alternative node sampling strategy aimed at creating structures resembling city clusters [BKMW10]. We implement and test both the uniform random sampling and this cluster-based sampling approach. Our experiments indicate no significant difference in the resulting separator sizes between the two sampling methods using the ABR generator framework.

The analysis of graphs generated using the ABR method yields separators that scale approximately as $\mathcal{O}(n^{1/2})$. This result is noteworthy because graphs generated by this process are typically highly non-planar. It serves as a reminder that observing $\mathcal{O}(n^{1/2})$ separator scaling does not necessarily imply planarity. Figure 4.8 provides a visual example of an ABR-generated graph and illustrates the observed separator scaling.



(a) Visualization of a synthetic graph generated using the ABR low highway dimension algorithm with 10K nodes.



(b) Separator size scaling observed during recursive partitioning of ABR-generated graphs. Separator sizes scale approximately as $\mathcal{O}(n^{1/2})$. The fit shown excludes subgraphs with ≤ 256 vertices due to their disproportionately large separatos.

Figure 4.8: Analysis of synthetic graphs generated using the ABR algorithm [AFGW10].

4.5 Hierarchical Structure

Real-world road networks intrinsically possess a hierarchical structure. Different types of roads cater to different travel distances and volumes, forming levels within the network. Depending on the specific taxonomy used, road networks are categorized into various numbers of classes or levels. For instance, a common classification for Germany includes (using original German terms alongside translations):

- Federal Motorways (Bundesautobahnen)
- Federal Highways (Bundesstraßen)
- State Roads (Landesstraßen)
- Municipal Roads (Gemeindestraßen)

This inherent hierarchy is another potentially relevant feature influencing separator sizes, which we investigate using synthetic models.

4.5.1 Voronoi-Based Hierarchical Generator

One approach to generating synthetic networks with hierarchical features follows the method proposed by Bauer et al. [BKMW10]. This method utilizes Voronoi diagrams iteratively. A Voronoi diagram, given a set of points P, partitions the plane into convex regions, where each region contains the area closer to one point in P than to any other point in P. The generation process commences within a predefined initial polygon, which defines the spatial extent of the synthetic network. Points, designated as sites P, are then generated within this polygon according to a chosen sampling strategy. One strategy involves sampling a specified number of points uniformly at random throughout the polygon's interior. Alternatively, to better

emulate settlement patterns, a clustered sampling approach can be employed. This approach involves first sampling locations for a number of population centers within the polygon. Each center is then assigned characteristics, such as a population size or an influence radius, drawn from a chosen distribution. Subsequently, points are sampled in the vicinity of each city center, concentrated within its influence radius, with the number of points related to the city's size.

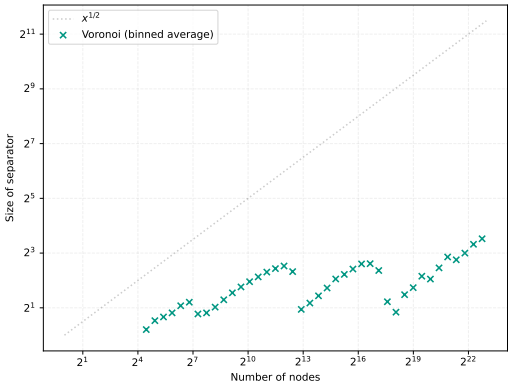
Regardless of the sampling method, the resulting set of points P serves as the input for computing the Voronoi diagram for this initial level. The Voronoi diagram partitions the polygon into regions based on nearest-neighbor relationships to the points in P. The edges of this Voronoi diagram (where adjacent regions meet) are then added to the synthetic graph. To introduce hierarchy, some of the resulting Voronoi regions (polygons) are selected, and the process is recursively applied within them: new points are sampled inside the selected region (using either uniform or clustered sampling), and a new Voronoi diagram is constructed within its boundaries, adding further edges to the graph.

Phase two of the generation process addresses the density of the graph G resulting from the Voronoi tessellation by constructing a sparse graph t-spanner H. A subgraph H is a t-spanner of G if it preserves all pairwise distances up to a multiplicative factor of t. The spanner is built using a greedy algorithm that iterates through the edges of G sorted by non-increasing length (longest first). An edge (u, v) with length $len(u, v)$ is added to the spanner H only if the current shortest path distance between u and v within H, denoted $dist_H(u, v)$, is greater than $t \cdot len(u, v)$. To compute $dist_H(u, v)$, Dijkstra's algorithm is employed. Additionally, a union-find data structure maintains the connected components of H, allowing for immediate addition of edges connecting previously disconnected vertices. Bauer et al. propose processing edges in non-increasing order (longest first) as a heuristic primarily intended to improve the computational efficiency of the spanner construction; the idea is to handle long edges while the intermediate spanner graph H is still sparse, potentially speeding up distance computations [BKMW10]. In our own experiments, however, we explore alternative edge processing orders. We observe that processing the edges in a random order, or even in increasing order of length (shortest first), appears to produce graph structures with greater visual resemblance to actual road networks compared to the longest-first approach described by Bauer et al., without substantially increasing computation times in our experiments.

This generation method introduces artifacts. For example, connections between major structures defined at higher levels (like large "cities") might be unrealistically sparse, potentially consisting of only a few edges. Another artifact is that higher-level Voronoi edges (e.g., "highways" separating major regions) act as hard boundaries that lower-level edges (within regions) often do not cross, creating unrealistically effective separators along these top-level edges.

Analyzing separators in the final sparse graphs generated by this Voronoi-based method reveals interesting characteristics related to the hierarchy. Plots of separator size versus subgraph size often exhibit noticeable local minima, which appear to correspond to the transitions between the hierarchical layers created during generation. This can lead to separators of approximately constant size when partitioning cuts primarily occur between these major structures at layer intersections. Within a single layer generated by the Voronoi tessellation (before sparsification), the separators tend to exhibit scaling closer to $\mathcal{O}(n^{1/2})$. Consequently, if the recursion depth is limited and the highest-level structures are allowed to grow large, their $\mathcal{O}(n^{1/2})$ separator behavior will dominate the separator size scaling. Figure 4.9 illustrates the structure and separator behavior of the final sparse graph.

(a) Visualization of a synthetic graph generated using the Voronoi method.

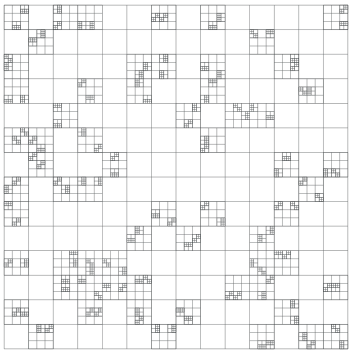


(b) Separator size scaling for the hierarchical Voronoi graph, showing local minima at layer transitions.

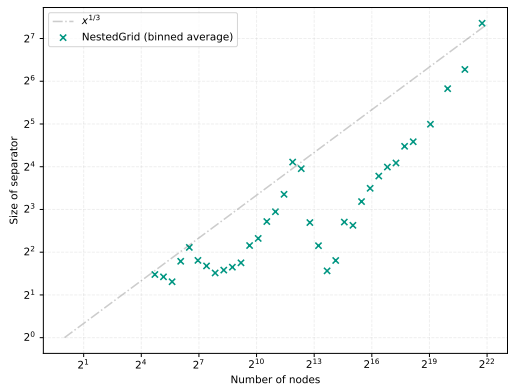
Figure 4.9: Analysis of synthetic graphs generated using the hierarchical Voronoi approach from Bauer et al. [BKMW10].

4.5.2 Nested Grids

To explore hierarchical effects in a more controlled manner, we also investigate a simpler but conceptually related approach using nested grids. This method aims to construct a hierarchical graph by recursively embedding grid graphs within the cells of a parent grid, potentially allowing us to deliberately engineer specific separator scaling properties. The fundamental idea involves placing copies of a subgraph (e.g., a square grid) into selected cells of a higher-level grid. This resulting composite graph can then itself be treated as a subgraph to be placed within cells of a grid at an even higher level, and so on. Figure 4.10 shows a small conceptual example of a nested grid with three layers.



(a) Conceptual illustration of a simple nested grid structure with three hierarchical levels.



(b) Separator size scaling for nested grids, showing pronounced local minima.

Figure 4.10: Analysis of synthetic graphs generated using the nested grid approach.

A nested grid construction with l layers is determined by l grid sizes (specifying the dimensions of the grid at each layer) and $l-1$ placement parameters (specifying how many subgraphs from layer i+1 are placed within layer i). This parameterization offers the possibility of attempting to enforce a target separator scaling, such as $\mathcal{O}(n^{1/3})$, by carefully choosing the parameters at each level transition. We assume that separating the nested structure primarily involves cutting through the top-level grid structure. If the number of embedded subgraphs is not excessively large, the separator size might be approximated by the width w of the top-level grid, similar to a standard grid separator. We can then estimate the total number of vertices $|V|$ at a given stage based on the grid width w of the current layer, the number k of subgraphs placed within its cells, and the size s of each subgraph:

$$|V| \approx w^2 + k \cdot s$$

This estimate is approximate as it may double-count vertices along the boundaries where subgraphs connect to the parent grid. To enforce $\mathcal{O}(n^{1/3})$ scaling specifically at this layer transition, we can equate the separator size (approximated by w) with $|V|^{1/3}$:

$$w \approx (|V|)^{1/3} \approx (w^2 + ks)^{1/3}$$

This leads to the cubic equation $w^3 - w^2 - ks = 0$. If we fix the subgraph count k and subgraph size s, we can solve for the grid width w required to satisfy this condition at the transition. One root of this cubic equation for w is given by:

$$w = \frac{(27ks + \sqrt{(27ks+2)^2 - 4} + 2)^{1/3}}{3 \cdot 2^{1/3}} + \frac{2^{1/3}}{3(27ks + \sqrt{(27ks+2)^2 - 4} + 2)^{1/3}} + \frac{1}{3}$$

By carefully selecting w based on k and s using this relationship at each hierarchical level, one can attempt to construct a graph where the separator size scales roughly as the cube root of the number of nodes at each layer transition. This controlled approach allows the local minima in separator sizes, already seen in the Voronoi method, to be made more pronounced and regular, as all subgraphs at a given level transition can be designed with the same size s. However, achieving a consistent overall $\mathcal{O}(n^{1/3})$ scaling for the entire graph across all sizes n remains a challenge.

The observed local minima in separator size occur systematically at the transitions between hierarchical levels. This is because the construction method inherently creates sparse connections between components defined at different levels. Figure 4.11 provides a simplified illustration of this principle, showing two example components from the same levels within the hierarchy, not the entire structure of a layer. In this specific illustration, the components (represented by nodes 1-4 and 5-8) are linked only by the edges (3)–(5) and (4)–(6). Consequently, removing just one endpoint from each connecting edge suffices to disconnect them. This demonstrates how a vertex separator of small, constant size (size 2) exists between levels, regardless of the internal size of the components themselves. Such constant-size separators at every level transition are the underlying cause of the pronounced local minima observed in the separator scaling plots for these nested structures.

4.6 Hierarchical Delaunay Graph Generation

Our efforts to generate synthetic graphs with realistic road network properties, particularly small separators, led us to refine techniques inspired by prior work on synthetic road network generation by [BKMW10]. A significant artifact of Voronoi-based hierarchical models was that major transport arteries, analogous to motorways, could never be crossed by lower-level

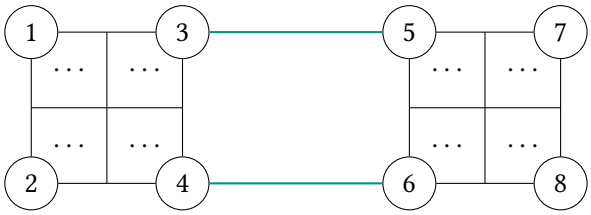


Figure 4.11: Simplified illustration of connectivity between components from adjacent hierarchical levels. The components (nodes 1-4 and 5-8) are connected only by the two green edges (3)–(5) and (4)–(6). Removing one endpoint from each edge (e.g., {3, 6}) forms a size-2 vertex separator, irrespective of component sizes.

roads. This arose because new expansion sites were generated strictly within the polygonal cells defined by the higher-level Voronoi tessellation, making top-level edges hard boundaries. Our revised core idea is to allow new centers of development to emerge from the existing network infrastructure itself; thus, new expansion sites are selected from the nodes of the current graph structure rather than from within areal polygons.

Our first refined concept based on this idea involved an iterative Delaunay approach. This process would begin with an initial global Delaunay triangulation of a point set. Then, in subsequent hierarchical levels, a subset of existing points would be selected as *expansion sites*. New points would be sampled in the vicinity of each of these sites. We then perform a new Delaunay triangulation, localized to the region around these newly added points. This iterative process aimed to simulate a "rich get richer" phenomenon: regions with a higher density of points would be more likely to have their constituent points selected as expansion sites for subsequent levels, leading to further concentrated development in those areas. This recursive generation would continue for a desired number of levels. As a final step, a planarization process, like described in Algorithm 3.1, is applied: edges interpreted as straight line segments are checked for intersections, and any intersections found are resolved by introducing new vertices at these points and subdividing the original edges. This method showed initial promise in creating structures that could yield separators with the same scaling as road networks.

Building upon insights from this approach, we developed a more streamlined and effective method, which is the primary focus of this section. Instead of performing Delaunay triangulations at each hierarchical level and planarizing in the end, we first generate the complete set of points across all desired levels. Only after all points have been generated is a single, global Delaunay triangulation performed on the entire final point set. The pseudocode for this hierarchical delaunay generation is detailed in Algorithm 4.1.

Pruning This global Delaunay triangulation typically results in a graph with an average vertex degree of approximately six, which is denser than typical road networks. To better emulate the characteristics of real road networks, e.g. an average degree around 2.5 and few nodes with high degree, we apply a pruning step. A pruning step was also part of the methodology suggested by Bauer et al. [BKMW10]; however, we adopt a different pruning strategy designed to complement the specific underlying structure of our generated graphs. Edges are considered in order of increasing Euclidean length (shortest first). An edge (u, v) is removed from the graph if the shortest path distance between u and v in the graph, after the hypothetical removal of (u, v), is no more than a factor α times the original direct length of (u, v), i.e., $\text{dist}_{G\setminus(u,v)}(u, v) \leq \alpha \cdot \text{length}(u, v)$. Through experimentation, we found that

Algorithm 4.1: Hierarchical Delaunay Graph Generation

Input: Number of hierarchical levels L,
Level expansion fractions f_i (where $f_1 = 1.0$),
Points to generate per expansion site k_i,
Expansion radii r_i,
$1 \leq i \leq L$

Output: Geometric graph $G = (V, E)$

```
P ⟵ {p_random}
forall i ∈ [1, L] do
    C_i ⟵ chooseRandom(P, ⌊f_i · |P|⌋)
    forall center ∈ C_i do
        P ⟵ P ∪ sampleRandomPointsInCircle(center, r_i, k_i)
G ⟵ delaunay(P)
pruneEdges(G)
return G
```

a pruning parameter of $\alpha = 2.5$ effectively reduced the average degree to the target range comparable to road networks. The pseudocode for this edge pruning process is given in Algorithm 4.2. To compare the graphs of pruned and non Figure 4.12.

Algorithm 4.2: Prune edges in G based on distance threshold.

Input: Graph $G = (V, E)$,
Pruning parameter t

Output: Pruned version of G

```
sortByLength(E)
forall (u, v) ∈ E do
    if dist(u, v) > t · len(u, v) then
        G ⟵ G \ (u, v)
```

(a) Graph before pruning.

(b) Graph after pruning.

Figure 4.12: Comparison of a pruned and unpruned hierarchical Delaunay graph.

The pruning step, while beneficial for achieving a target average degree and a more realistic degree distribution, does not fundamentally alter the asymptotic separator scaling of the generated Delaunay graphs. Figure 4.13 illustrates that the separator size scaling remains similar for both pruned and unpruned graphs.

(a) Separator scaling before pruning. **(b)** Separator scaling after pruning.

Figure 4.13: Comparison of separator size scaling in hierarchical Delaunay graphs before and after the pruning step.

Given that the pruning step can be computationally intensive due to the numerous shortest path computations (Dijkstra, as precomputation is not possible because we are altering the graph) required, we also explored an approximated parallel version. In this variant, edges are processed in chunks, typically matching the number of available processing threads. The shortest path computations for all candidate edges within a chunk are performed in parallel, based on the graph state before any edges in that chunk are removed. After all checks for the current chunk are complete, the identified edges are removed. This parallelization is only an approximation: an edge (u, v) might be removed because its check relied on a path through another edge (x, y) from the same chunk, which is also simultaneously identified for removal. If (x, y) is removed, the shortest path for (u, v) might have become longer than the threshold, but this would not be caught as the checks were concurrent. Despite this potential for over-pruning, our empirical results suggest that this approximation has a negligible impact on the final graph structure and its properties. This is likely because edges are sorted by length before being chunked, meaning edges within a single chunk are often spatially distributed across the graph rather than being locally interdependent, minimizing negative interference.

5 Conclusion

5.1 Future Work

Bibliography

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. "Highway Dimension, Shortest Paths, and Provably Efficient Algorithms". In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*. Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Jan. 17, 2010, pp. 782–793. ISBN: 978-0-89871-701-3 978-1-61197-307-5. DOI: *10.1137/1.9781611973075.64*.

[BCRW16] Reinhard Bauer, Tobias Columbus, Ignaz Rutter, and Dorothea Wagner. "Search-space size in contraction hierarchies". In: *Theoretical Computer Science* Volume 645 (Sept. 2016), pp. 112–127. ISSN: 03043975. DOI: *10.1016/j.tcs.2016.07.003*.

[BD10] Reinhard Bauer and Daniel Delling. "SHARC: Fast and robust unidirectional routing". In: *ACM J. Exp. Algorithmics* Volume 14 (Jan. 5, 2010), 4:2.4–4:2.29. ISSN: 1084-6654. DOI: *10.1145/1498698.1537599*.

[BFSS07] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. "Fast Routing in Road Networks with Transit Nodes". In: *Science* Volume 316 (Apr. 27, 2007), pp. 566–566. ISSN: 0036-8075, 1095-9203. DOI: *10.1126/science.1137521*.

[BKMW10] Reinhard Bauer, Marcus Krug, Sascha Meinert, and Dorothea Wagner. "Synthetic Road Networks". In: *Algorithmic Aspects in Information and Management*. Edited by Bo Chen. Vol. 6124. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 46–57. ISBN: 978-3-642-14354-0 978-3-642-14355-7. DOI: *10.1007/978-3-642-14355-7_6*.

[Blä+25] Thomas Bläsius, Valentin Buchhold, Dorothea Wagner, Tim Zeitz, and Michael Zündorf. *Customizable Contraction Hierarchies – A Survey*. Feb. 14, 2025. arXiv: *2502.10519[cs]*.

[BO79] Bentley and Ottmann. "Algorithms for Reporting and Counting Geometric Intersections". In: *IEEE Transactions on Computers* Volume C-28 (Sept. 1979), pp. 643–647. ISSN: 0018-9340. DOI: *10.1109/TC.1979.1675432*.

[Bro89] A. Broder. "Generating random spanning trees". In: *30th Annual Symposium on Foundations of Computer Science*. 30th Annual Symposium on Foundations of Computer Science. Research Triangle Park, NC, USA: IEEE, 1989, pp. 442–447. ISBN: 978-0-8186-1982-3. DOI: *10.1109/SFCS.1989.63516*.

[CK87] Chlamtac and Kutten. "Tree-Based Broadcasting in Multihop Radio Networks". In: *IEEE Transactions on Computers* Volume C-36 (Oct. 1987), pp. 1209–1223. ISSN: 0018-9340. DOI: *10.1109/TC.1987.1676861*.

[DGPW11] Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. "Customizable Route Planning". In: *Experimental Algorithms*. Edited by Panos M. Pardalos and Steffen Rebennack. Berlin, Heidelberg: Springer, 2011, pp. 376–387. ISBN: 978-3-642-20662-7. DOI: *10.1007/978-3-642-20662-7_32*.

[DSW16] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. "Customizable Contraction Hierarchies". In: *ACM Journal of Experimental Algorithmics* Volume 21 (Nov. 4, 2016), pp. 1–49. ISSN: 1084-6654, 1084-6654. DOI: *10.1145/2886843*.

[EG08] David Eppstein and Michael T. Goodrich. "Studying (non-planar) road networks through an algorithmic lens". In: *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. New York, NY, USA: Association for Computing Machinery, Nov. 5, 2008, pp. 1–10. ISBN: 978-1-60558-323-5. DOI: *10.1145/1463434.1463455*.

[EGS10] David Eppstein, Michael T. Goodrich, and Darren Strash. "Linear-Time Algorithms for Geometric Graphs with Sublinearly Many Edge Crossings". In: *SIAM Journal on Computing* Volume 39 (Jan. 2010), pp. 3814–3829. ISSN: 0097-5397, 1095-7111. DOI: *10.1137/090759112*.

[GHUW19] Lars Gottesbüren, Michael Hamann, Tim Niklas Uhl, and Dorothea Wagner. "Faster and Better Nested Dissection Orders for Customizable Contraction Hierarchies". In: *Algorithms* Volume 12 (Sept. 16, 2019), p. 196. ISSN: 1999-4893. arXiv: *1906.11811[cs]*.

[GKW06] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. "Reach for A*: Efficient Point-to-Point Shortest Path Algorithms". In: *2006 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, Jan. 21, 2006, pp. 129–143. DOI: *10.1137/1.9781611972863.13*.

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks". In: *Experimental Algorithms*. Edited by Catherine C. McGeoch. Berlin, Heidelberg: Springer, 2008, pp. 319–333. ISBN: 978-3-540-68552-4. DOI: *10.1007/978-3-540-68552-4_24*.

[Kru56] Joseph B. Kruskal. "On the shortest spanning subtree of a graph and the traveling salesman problem". In: *Proceedings of the American Mathematical Society* Volume 7 (Feb. 1956), pp. 48–50. ISSN: 0002-9939, 1088-6826. DOI: *10.1090/S0002-9939-1956-0078686-7*.

[LT77] Richard J. Lipton and Robert Endre Tarjan. "Applications of a planar separator theorem". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. 18th Annual Symposium on Foundations of Computer Science (sfcs 1977). Providence, RI, USA: IEEE, Sept. 1977, pp. 162–170. DOI: *10.1109/SFCS.1977.6*.

[LT79] Richard J. Lipton and Robert Endre Tarjan. "A Separator Theorem for Planar Graphs". In: *SIAM Journal on Applied Mathematics* Volume 36 (Apr. 1979), pp. 177–189. ISSN: 0036-1399, 1095-712X. DOI: *10.1137/0136016*.

[PTV09] PTV Group. *DIMACS-Europe Graph for the 9th DIMACS Implementation Challenge. Visit www.iti.kit.edu/resources/roadgraphs.php for details on how to acquire this graph.* 2009.

[SS05] Peter Sanders and Dominik Schultes. "Highway Hierarchies Hasten Exact Shortest Path Queries". In: *Algorithms – ESA 2005*. Edited by Gerth Stølting Brodal and Stefano Leonardi. Berlin, Heidelberg: Springer, 2005, pp. 568–579. ISBN: 978-3-540-31951-1. DOI: *10.1007/11561071_51*.

[SS13] Peter Sanders and Christian Schulz. “Think Locally, Act Globally: Highly Balanced Graph Partitioning”. In: *Experimental Algorithms*. Edited by Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela. Berlin, Heidelberg: Springer, 2013, pp. 164–175. ISBN: 978-3-642-38527-8. DOI: *10.1007/978-3-642-38527-8_16*.