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

Empirical observations suggest that road networks possess small graph separators, scaling approximately as $\mathcal{O}(n^{1/3})$. This scaling is considerably smaller than the worst-case bounds established for major graph classes, such as planar graphs, which have separators scaling as $\mathcal{O}(n^{1/2})$. However, the underlying structural properties responsible for this phenomenon remain poorly understood. This thesis systematically investigates which network characteristics explain the presence of these small separators.

Our analysis of real-world network data indicates a slightly larger separator scaling of approximately $\mathcal{O}(n^{0.37})$. We evaluate the impact of several graph properties by attempting to replicate the scaling behavior with synthetic graph models. The analysis reveals that simpler properties, such as sparsity or the existence of an embedding, are insufficient on their own to explain the separator sizes observed in road networks. Models based on such characteristics consistently yield separators that scale as $\mathcal{O}(n^{1/2})$ or worse.

Instead, our results indicate that small separators are an emergent property of a hierarchical structure. This conclusion is substantiated by our generative models. We find that two conceptually different approaches, one based on explicit hierarchical construction and another simulating physical barriers with multi-scale noise, both produce graphs whose separators scale as $\mathcal{O}(n^{0.37})$, closely matching our empirical findings. The shared success of diverse models that enforce a hierarchical organization suggests that this is a critical property responsible for the small separators in road networks.

Zusammenfassung

Empirische Beobachtungen legen nahe, dass reale Straßennetzwerke kleine Graphseparatoren aufweisen, deren Größe etwa in der Größenordnung von $\mathcal{O}(n^{1/3})$, wächst. Dieses Wachstum ist signifikant langsamer als die Worst-Case-Schranken bekannter Graphklassen, wie die planarer Graphen mit einer Separatorgröße von $\mathcal{O}(n^{1/2})$. Die diesem Phänomen zugrunde liegenden strukturellen Eigenschaften sind jedoch bisher nur unzureichend verstanden. Diese Arbeit untersucht daher systematisch, welche Netzwerkeigenschaften das Vorhandensein kleiner Separatoren erklären.

Unsere Analyse von realen Netzwerkdaten deutet auf ein Wachstum der Separatorgröße von circa $\mathcal{O}(n^{0.37})$ hin. Wir bewerten den Einfluss verschiedener Grapheigenschaften, indem wir versuchen, dieses Wachstumsverhalten mithilfe synthetisch generierter Graphen zu replizieren. Die Untersuchung zeigt, dass simplere Eigenschaften wie geringe Dichte oder die Existenz einer Einbettung allein nicht ausreichen, um die in Straßennetzwerken beobachteten Separatorgrößen zu erklären. Modelle, die allein auf solchen Merkmalen basieren, führen typischerweise zu Separatoren, deren Größe mit $\mathcal{O}(n^{1/2})$ oder schlechter wächst.

Unsere Ergebnisse deuten stattdessen darauf hin, dass die kleinen Separatoren eine direkte Folge einer hierarchischen Struktur sind. Diese Schlussfolgerung wird durch unsere generativen Modelle gestützt. Wir stellen fest, dass zwei konzeptionell unterschiedliche Ansätze, einer basierend auf expliziter hierarchischer Konstruktion, der andere auf der Simulation physischer Barrieren mittels mehrskaligem Rauschen, Graphen mit einer Separatorgröße von $\mathcal{O}(n^{0.37})$ erzeugen. Dieses Ergebnis kommt den empirischen Beobachtungen in Straßengraphen sehr nahe. Der Erfolg dieser diversen Modelle, die eine hierarchische Organisation erzwingen, legt nahe, dass Hierarchie die entscheidende Eigenschaft ist, die für die kleinen Separatoren in Straßennetzwerken verantwortlich ist.

Contents

1 Introduction

Road networks are a fundamental component of modern infrastructure, presenting significant computational challenges for applications like large-scale navigation due to their immense size. To address these challenges, we model them as mathematical graphs, but must recognize that they are not arbitrary structures, they are imbued with unique topological properties by real-world geographic and economic constraints. The central motivation for this thesis is to investigate why road networks possess an inherent structure that allows for highly efficient partitioning through small, balanced separators. This chapter introduces this core topic, outlines our contributions, and provides a roadmap for the investigations that follow.

1.1 Motivation

In graph theory, a balanced separator is a small subset of vertices whose removal partitions a graph into disconnected components of roughly equal size. The size of the smallest such separator is a fundamental graph property, as it governs the performance of numerous divide-and-conquer algorithms. The foundational work by Lipton and Tarjan on planar separators, for example, demonstrates how the existence of small separators can be leveraged to create efficient algorithms for numerous problems on planar graphs [LT77 | LT79].

Work in the context of advanced routing algorithms suggests that road networks have balanced separators with sizes scaling on the order of $\mathcal{O}(n^{1/3})$, as observed in experiments by Dibbelt et al. [DSW16]. This is a remarkable finding, as it is significantly smaller than the $\mathcal{O}(n^{1/2})$ worst-case bound guaranteed for planar graphs, even though road networks can be treated as nearly planar structures due to their geographic embedding [LT77 | LT79].

The existence of such small graph separators has significant practical implications. In road networks, small separators enable the creation of highly effective node orders, which are critical for the performance of state-of-the-art routing algorithms like Customizable Contraction Hierarchies (CCH). This thesis therefore seeks to uncover the properties responsible for the presence of these small separators. We aim to determine whether they stem from intrinsic graph characteristics, such as limited vertex degrees or sparsity, or from real-world physical features, such as geographic borders, rivers, or a hierarchical road system. Gaining insight into these properties promises not only to advance our theoretical understanding of this graph class but also to offer practical benefits, such as generating more realistic synthetic benchmarks for algorithm evaluation.

From a theoretical standpoint, road networks represent a particularly intriguing subject. Classical results in graph theory establish separator sizes at distinct scales, such as $\mathcal{O}(1)$ for graph classes with bounded treewidth and $\Theta(n^{1/2})$ for many other classes, e.g., planar graphs. To the best of our knowledge, established graph classes that consistently exhibit separator sizes strictly between these well-known bounds are not prominently featured in the literature. This finding positions road networks within a sparsely populated intermediate complexity range, thereby highlighting the compelling nature of investigating their unique structural properties.

1.2 Contribution

The primary contributions of this thesis are both empirical and generative. We first conduct a rigorous empirical analysis of separator scaling in large-scale, real-world road networks, establishing a power-law relationship of approximately $\mathcal{O}(n^{0.37})$. We then systematically evaluate a wide range of synthetic graph models, demonstrating that simple, single-property models based on degree distribution, basic locality, or standard planarity are mostly insufficient to reproduce this observed scaling. To address this gap, we develop and analyze two novel, more complex generative models: a hierarchical Delaunay generator that can replicate the target scaling through parameter tuning, and a multi-scale Perlin noise model that simulates physical barriers. A key finding is that this noise-based model naturally produces graphs with the desired $\mathcal{O}(n^{0.37})$ separator scaling without requiring extensive fine-tuning, suggesting it captures a fundamental principle of road network formation. Based on these findings, we propose that the small separators in road networks are an emergent property of their multi-scale structure, resulting directly from the network's adaptation to a complex physical and hierarchical environment.

1.3 Outline

The remainder of this thesis is structured as follows. Chapter 2 provides the necessary background by formally defining fundamental concepts from graph theory, including graph separators and planarity. Chapter 3 then examines Customizable Contraction Hierarchies as a state-of-the-art application that effectively leverages small separators, providing crucial real-world context for why this graph property is significant. Chapter 4 presents a detailed empirical analysis of real-world road networks, establishing their key structural characteristics concerning separator sizes, degree distribution, diameter, and other relevant metrics. Chapter 5 details our investigation into synthetic graph generation, systematically exploring a range of models from simple, property-based generators to more complex hierarchical and noise-based approaches. Finally, Chapter 6 summarizes the key findings of this work, provides an answer to the central research question, and discusses the implications, limitations, and potential directions for future research.

2 Preliminaries

This chapter establishes the theoretical preliminaries that form the basis of our investigation. We begin by covering fundamental concepts and terminology from graph theory, focusing on the properties relevant to large networks like roads. Subsequently, we provide a detailed exposition of graph separators, the central concept of this thesis, including their formal definition, the importance of balanced partitions, and the specific algorithms employed in our work to compute them.

2.1 Graph Theory

Road networks can be modeled as graphs. A graph G is formally defined as a tuple (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 structure 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 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 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.

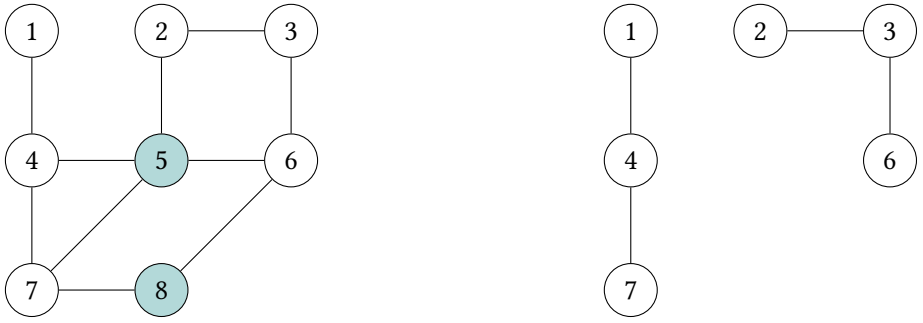
2.2 Graph Separators

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 $G[V_i]$ ($i \in \{1, 2, \ldots, k\}$) 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.

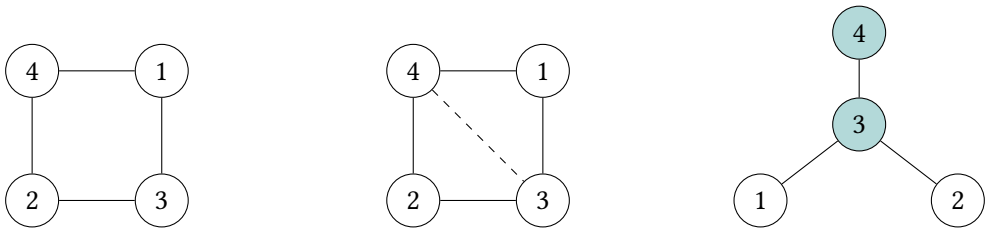
To compute separators, various algorithms can be employed. In this thesis, we primarily utilize InertialFlowCutter [GHUW19]. This algorithm leverages geometric embeddings, often available for road networks, to compute high-quality node orderings efficiently. These node orderings serve as the basis for 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. Since InertialFlowCutter is specifically designed for the structure of road networks, its performance on the more arbitrary synthetic graphs investigated in this thesis is not guaranteed to be optimal. To address this potential bias and validate our results, we cross-check our findings using two other established partitioning frameworks: KaHIP [SS13] and FlowCutter [HS18]. Despite its specialization, we observe that

for most graph instances examined in this thesis, InertialFlowCutter consistently produces separators of smaller size compared to the other two methods. A key constraint of InertialFlowCutter, however, is its requirement of a geometric embedding as input. Consequently, for synthetic graphs generated without an explicit embedding, we rely exclusively on KaHIP and FlowCutter for separator computation.

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 [Blä+25]. 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 $\text{rank}(w) > \text{rank}(v)$. If a node does not have 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 v_1 (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.

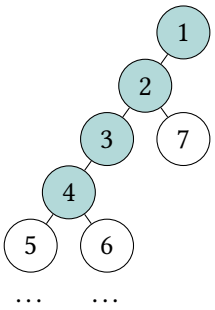


Figure 2.3: Separator identification with a significance threshold. The path extends downwards from node 1. At node 2, child 7 represents an insignificant subgraph (below the threshold), so the traversal continues via the single significant child path towards node 3. Node 4 is the first node encountered 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}.

3 CCH: A State-of-the-Art Application

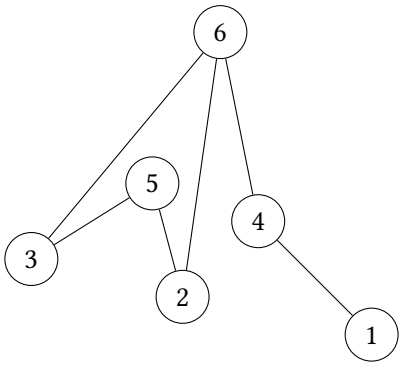
As established in the introduction, real-world road networks exhibit small separators. This observation, stemming from the experimental results of the Customizable Contraction Hierarchies (CCH) paper [DSW16], is a key inspiration for this thesis. This chapter details how the state-of-the-art Customizable Contraction Hierarchies (CCH) algorithm leverages small separators to enable fast queries, particularly in dynamic scenarios with changing edge weights. Understanding this application provides further context for the significance of the small separator phenomenon that this thesis investigates.

Customizable Contraction Hierarchies address the challenge of dynamic edge weights by employing a three-phase approach: an initial, topology-dependent precomputation; a subsequent, fast customization phase that incorporates current edge weights; and finally, an efficient query phase [DGPW11 | DSW16]. 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. This section provides an overview of the CCH algorithm, focusing on how its components benefit from the small separators found in road networks.

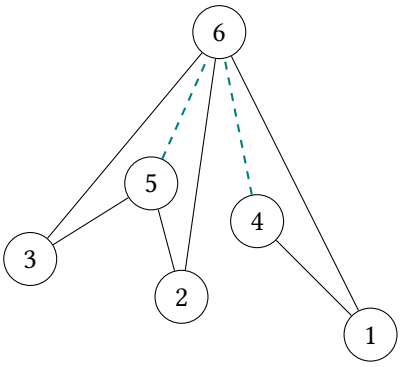
Precomputation The CCH precomputation phase introduces shortcut edges based on a given contraction order. These shortcuts effectively bypass sections of the graph, allowing algorithms to skip over entire subgraphs, unless the target node resides within such a subgraph. Furthermore, the specific process 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 an efficient bidirectional search by restricting exploration to higher-ranked neighbors.

The 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 according to their rank, 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. Any 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 3.1.

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.



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



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

Figure 3.1: Example of the CCH precomputation step. Nodes are named and positioned based on their rank.

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 3.2 provides a visual representation of this recursive partitioning strategy.

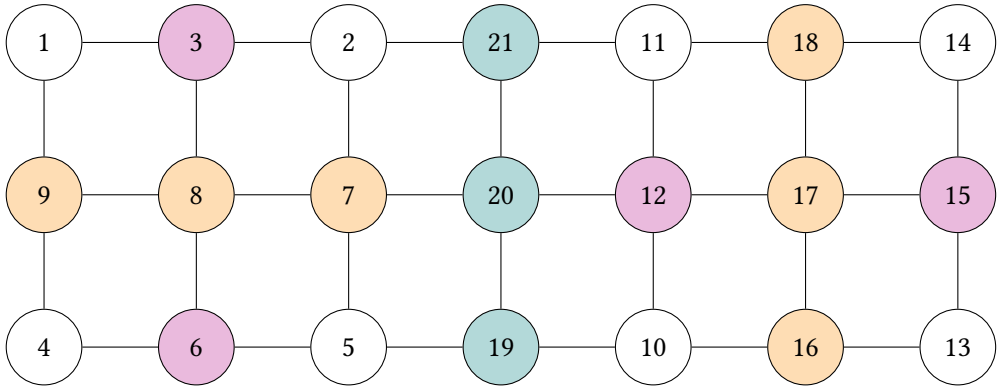


Figure 3.2: Example of a nested dissection. The top level separator is shown in teal, the second level in orange and the third level in purple. The nodes are named according to their rank in the resulting order.

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

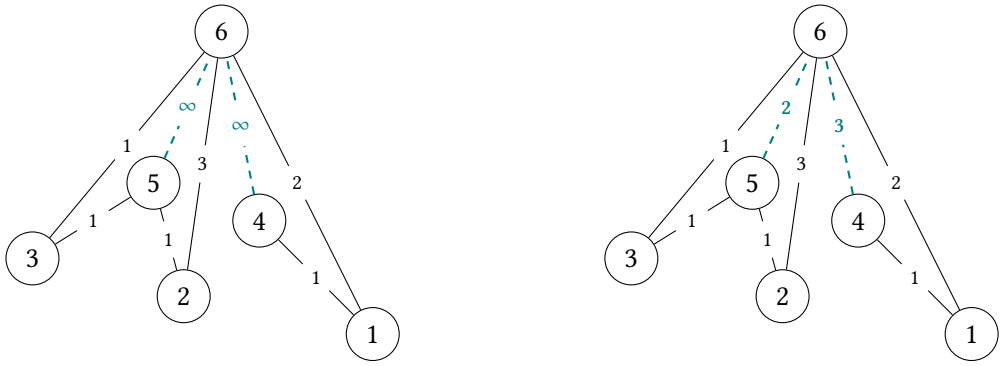
vertices of the graph in ascending order of their precomputed rank. For each vertex x, it considers all upward edges $\{x, y\}$ in the graph, where y is a neighbor of x and $rank(y) > rank(x)$. For every such edge $\{x, y\}$, we determine all lower triangles $\{x, y, z\}$. If the path through z offers 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 3.1. An illustration of the customization process is provided in Figure 3.1.

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

Algorithm 3.1: 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)}
```



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

(b) Graph after customization.

Figure 3.3: Example of the CCH customization step.

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 3.4 illustrates the elimination tree for the example graph shown in Figure 3.1. 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

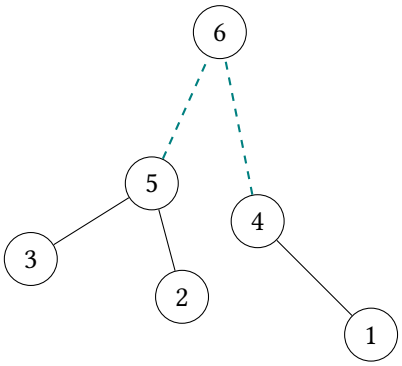


Figure 3.4: Elimination tree for the example graph in Figure 3.1.

parent node in the elimination tree. Otherwise (if u_t has a rank less than or equal to that of 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 induced by a 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 a size of 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 $\mathcal{O}\left(n^\beta\right)$, under the assumption that small separators can be found recursively. Note that while we do not have worst-case guarantees for the separator sizes in real road networks, they are expected to be small in practice. Thus, the performance of the CCH algorithm is directly linked to the ability to find small separators.

4 Road Network Properties

This chapter provides an empirical analysis of several fundamental properties of real-world road networks. We investigate key characteristics such as separator sizes, degree distribution, planarity, hierarchy, and diameter to establish a baseline understanding of these networks. These empirical findings serve as a foundation for the synthetic graph generation and analysis presented in subsequent chapters.

4.1 Separator Sizes

To empirically investigate the relationship between graph size and separator size in road networks, we analyze the DIMACS Europe dataset provided by PTV [PTV09]. We take the largest connected component of this graph and make it undirected by ignoring edge directions as we are primarily interested in the topological structure rather than the specific flow direction of traffic. A crucial first step in our research is to empirically validate the scaling of these separators. We seek to determine if they scale near $\mathcal{O}(n^{1/3})$, as suggested by prior work [DSW16], or if an alternative scaling, like $\mathcal{O}(n^{1/2})$, merely appears smaller due to a low constant factor. Figure 4.1 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 data points are generated by recursively applying nested dissection, computing separators first for the original graph and then for the subgraphs induced at each subsequent level.

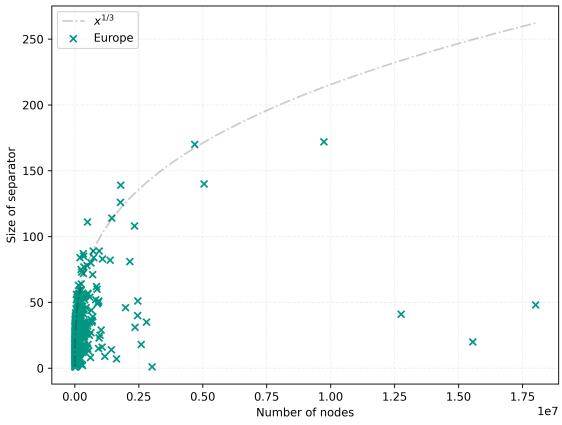


Figure 4.1: Empirical separator size versus subgraph size for the Europe road network. Each point represents a subgraph and its corresponding separator size. Separators were computed using InertialFlowCutter.

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. This is due to specific geographic bottlenecks, as illustrated in Figure 4.2. 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.

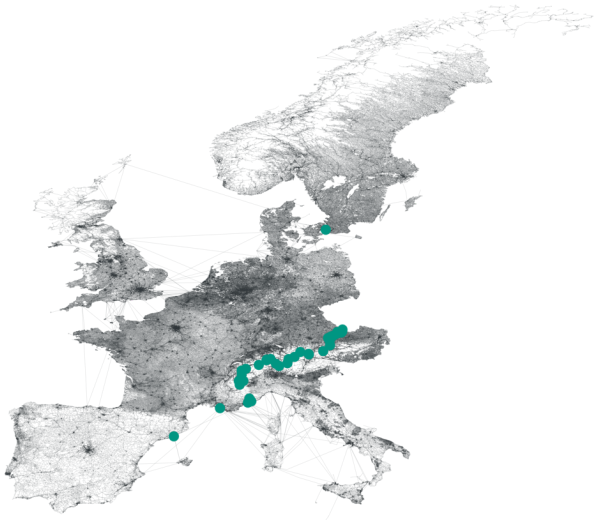
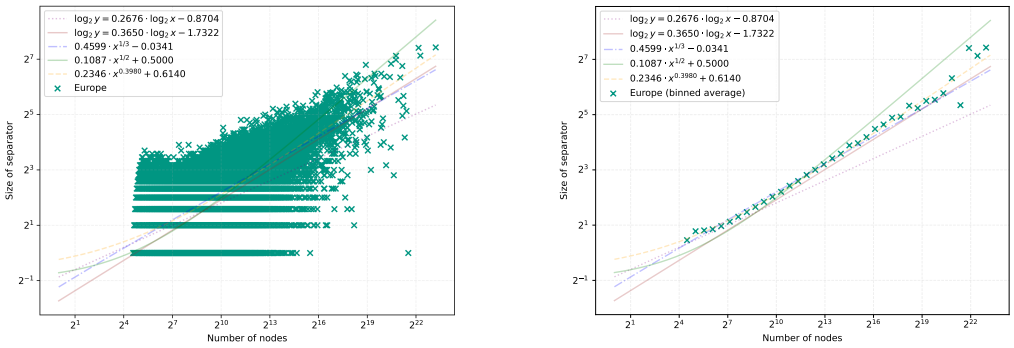


Figure 4.2: Illustration of a geographically influenced outlier at the continental scale: removal of a few nodes disconnects the entire Scandinavian peninsula.

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 on 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. 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.

When constructing the log-log plot, this binning procedure is applied to the logarithmically transformed data. Figure 4.3 illustrates the binned data points on logarithmic axes alongside the non-binned data.



(a) Non-binned separator sizes.

(b) Binned average separator sizes.

Figure 4.3: Separator sizes of Europe on logarithmic axes (subgraphs < 10M nodes). Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

To quantify the relationship between separator size y and subgraph size x, we perform 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 $\mathcal{O}(x^{0.2676})$, seemingly better than the hypothesized $\mathcal{O}(x^{1/3})$. However, this result is likely skewed by the large number of small subgraphs, which may not fully represent the scaling trend at larger sizes. To test this hypothesis, we perform a second regression exclusively on data points corresponding to subgraphs with more than $2^8 = 256$ vertices. This analysis yields the relationship $\log y \approx 0.3650 \log x - 1.7322$.

We report these coefficients to four decimal places in accordance with scientific reporting conventions. However, we acknowledge that this level of precision might overstate the certainty of the fit, given the inherent noise in empirical measurements 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 as 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 to $1/3$ or $1/2$, the overall R^2 scores remain relatively low, indicating only a moderately good fit. These fitted curves are

visualized in Figure 4.3. The fitted lines from the non-binned data (Figure 4.3a) are reproduced in Figure 4.3b to facilitate comparison with the binned averages; visual alignment may differ due to the binning process.

Observations from the binned log-log plot in Figure 4.3b suggest that the linear relationship between $\log y$ and $\log x$ is most consistent within a specific range of subgraph sizes. The data appear particularly stable for subgraph sizes x between approximately 2^7 and 2^{18} nodes. For subgraphs smaller than 2^7 nodes, a slight upward deviation 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 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 ($p \approx 1.64 \times 10^{-20}$).

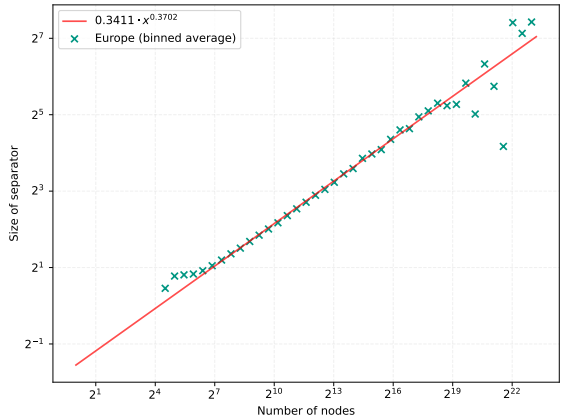


Figure 4.4: Linear regression fit to the binned data of separator size versus subgraph size, plotted on logarithmic axes. The regression considers only bins corresponding to subgraph sizes (x) in the range $2^7 \leq x \leq 2^{18}$. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

Visually, this line provides an excellent fit to the binned data points within the considered range, as illustrated in Figure 4.4. 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 high confidence that the observed scaling behavior is close to, yet slightly above, $\mathcal{O}(n^{1/3})$.

Initial Deviations in Separator Scaling Empirical studies of road networks reveal a notable deviation in separator scaling for small subgraphs. This deviation is particularly apparent for subgraphs containing approximately 2^6 vertices, where separator sizes are often larger than the general scaling trend would suggest, as can be observed in Figure 4.3a. Figure 4.5 presents a histogram that further visualizes the distribution of these separator sizes. This phenomenon corresponds with the observation of a higher meshedness coefficient for smaller, denser urban cores, as discussed later in Section 4.2. The more grid-like, densely connected structure implied by a higher meshedness may contribute to these comparatively larger separators at smaller scales.

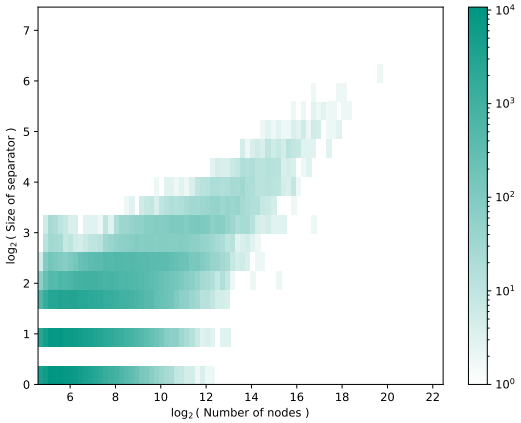


Figure 4.5: Histogram illustrating the distribution of separator sizes for road networks. A slight increase in relative separator size is observed for graphs with approximately 2^6 nodes. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

4.2 Degree Distribution

Road networks are characteristically sparse graphs. The DIMACS Europe road network dataset from PTV [PTV09], used in our experiments, exemplifies this with an average vertex degree of approximately 2.5. Figure 4.6 presents this network's degree distribution, which reveals that a vast majority of vertices (approximately 99.8%) have a degree less than 5. The maximum degree observed in this dataset is 12, attained by a single node.

Implications for Other Structural Metrics The degree distribution and overall sparsity also influence other structural metrics, particularly when considering the near-planar nature of road networks. One such metric for planar-like graphs is the meshedness coefficient, α,

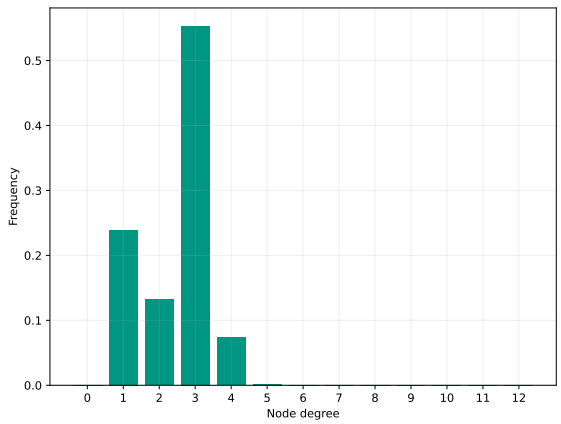


Figure 4.6: Degree distribution of the DIMACS Europe road network [PTV09]. The x-axis represents vertex degree, and the y-axis indicates the fraction of vertices.

which quantifies the density of cycles or bounded faces within a planar graph [Buh+06]. It is calculated as the ratio of a graph's actual number of bounded faces ($m - n + 1$) to the maximum possible for a planar graph with n vertices and m edges, via the formula $\alpha = \frac{m-n+1}{2n-5}$. This coefficient ranges from 0 for trees to 1 for maximal planar graphs. Buhl et al. suggest that this coefficient can also be used to gauge a network's robustness to disconnections and its cost in terms of total edge length [Buh+06]. For the DIMACS Europe road network, after applying a planarization procedure (detailed in Section 4.3), we compute a meshedness coefficient $\alpha \approx 0.1166$. This relatively low value underscores the network's general sparsity and somewhat tree-like macroscopic structure. For context, the meshedness coefficient for a square grid approaches 0.5 as $n \to \infty$, while a full Delaunay triangulation, as an almost maximal planar graph, approaches 1. In contrast to the continental scale, urban cores are typically more densely meshed. To illustrate this, we analyze the inner-city structure of Karlsruhe, a sub-region available within the PTV/DIMACS dataset. Representations of roads often include long chains of degree-2 vertices that model segments rather than structurally significant junctions. Such modeling can artificially skew the meshedness coefficient to lower values because adding these degree-2 vertices increases n without increasing the graph's cycle count. Therefore, to analyze the underlying junction-based topology independent of this modeling artifact, contracting degree-2 vertices is a valuable normalization step. After applying this contraction, the graph for Karlsruhe's inner city yields a meshedness coefficient of approximately 0.25. This higher value reflects the more grid-like layout characteristic of dense urban centers. Applying this same normalization step to the entire DIMACS Europe graph has a less pronounced impact, increasing its meshedness coefficient only slightly from 0.1166 to 0.1317.

4.3 Planarity

Road networks can be modeled as nearly planar graphs, meaning they permit an 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 on the order of $\mathcal{O}(n^{1/2})$, a

sub-linear count relative to the number of edges [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 separator properties, or if the occasional non-planar elements are merely incidental. This prompts the question of how separator sizes are affected when road networks are transformed into strictly planar graphs.

To obtain a planar representation, we begin with the existing graph structure where vertices possess associated geometric coordinates. Each edge is interpreted as the straight line segment connecting the coordinates of its incident vertices. The algorithm then identifies all geometric intersection points between these line segments. A new vertex is introduced into the graph at the coordinates of each intersection, provided this point does not coincide with an existing vertex. Any original edge containing one or more such intersection points is then removed and replaced by a sequence of new, shorter edges that connect the original endpoints and the new intersection vertices in their linear order. This process transforms the initial graph into a planar graph embedding by explicitly representing all edge crossings as vertices. For efficient execution, we utilize a spatial index over the bounding boxes of all edges. This structure enables rapid identification of potential intersections by querying for overlapping bounding boxes, which can then be verified for actual crossings. While other approaches exist, such as the Bentley-Ottmann algorithm for general line segment intersection [BO79] or linear-time algorithms tailored for graphs with a sublinear number of crossings [EGS10], our spatial index-based method is chosen for its implementation simplicity, as performance is not a critical concern for this pre-processing step. Since a single edge may cross multiple other edges, intersection points are sorted along each original edge before new sub-edges are introduced. Pseudo-code for this planarization algorithm is provided in Algorithm 4.1.

Algorithm 4.1: 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 ⟵ { ∅ for e in E }
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_edge(e, vertices)
12     add_new_edges(e, vertices)
```

We apply this planarization method to several real-world road networks. The Karlsruhe network, with approximately 120,000 nodes, has around 2,500 crossings. The Germany network, comprising about 6 million nodes, has approximately 100,000 crossings. Finally, the Europe network, with around 18 million nodes, contains about 300,000 crossings. These numbers are a little higher than intersection counts on the order of $\mathcal{O}(n^{1/2})$ reported in prior

studies, but are of a comparable order of magnitude [EG08]. The differences could be explained by our modeling of edges as straight lines rather than more complex curves, and might be mitigated by using a more detailed road network model like OpenStreetMap.

Analysis of separator sizes shows minimal variation post-planarization. We identify $\frac{2}{3}$-balanced separators with sizes still scaling approximately as $\mathcal{O}(n^{0.37})$, aligning with the values from the original non-planar graphs. A comparison of the separator sizes in the planar and non-planar versions of the Europe network is depicted in Figure 4.7.

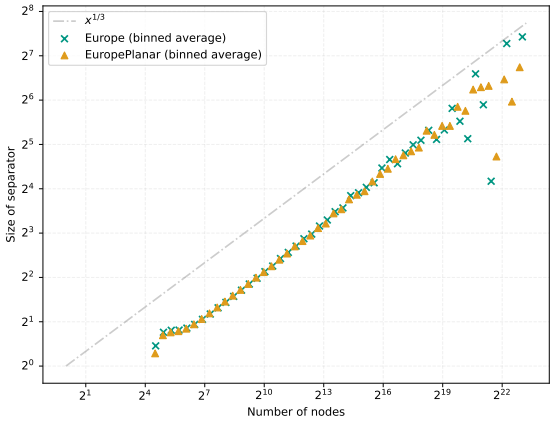


Figure 4.7: Comparison of separator sizes in the European road network: planar vs. non-planar versions. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

Our findings indicate that separators in non-planar road networks closely resemble those in their planarized counterparts in terms of overall scaling. However, a separator from the original graph is not always directly valid in the planarized version, as edge crossings can create new paths between previously separated components. We investigated this by traversing a single arm of a nested dissection and checking if the separators for the original subgraphs remain valid in their planarized versions. Many separators can be applied directly without modification, this is more probable for smaller subgraphs that are less likely to contain edge crossings. In many cases, an original separator must be augmented with additional nodes to restore the partition, resulting in a slightly larger separator. Figure 4.8 provides a real-world example, visualizing how a separator from the non-planar Karlsruhe network has to be augmented to remain valid after planarization.

The new structure created by planarization can also reveal a new, more efficient separator that is smaller than the original, as shown in the example in Figure 4.9. Conversely, there are also rare cases in which no better separator can be found in the planarized graph, as illustrated in Figure 4.10.

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 without losing essential separator properties.

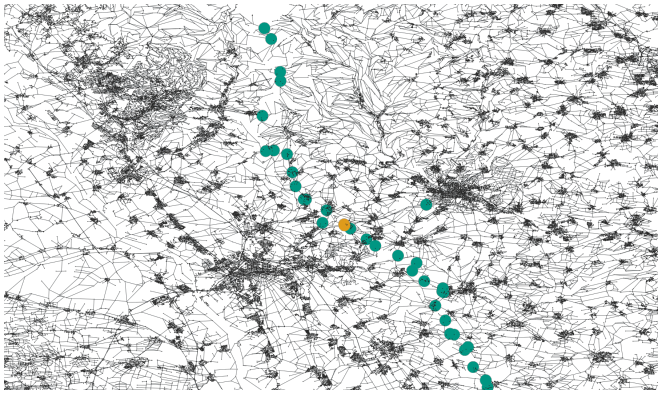
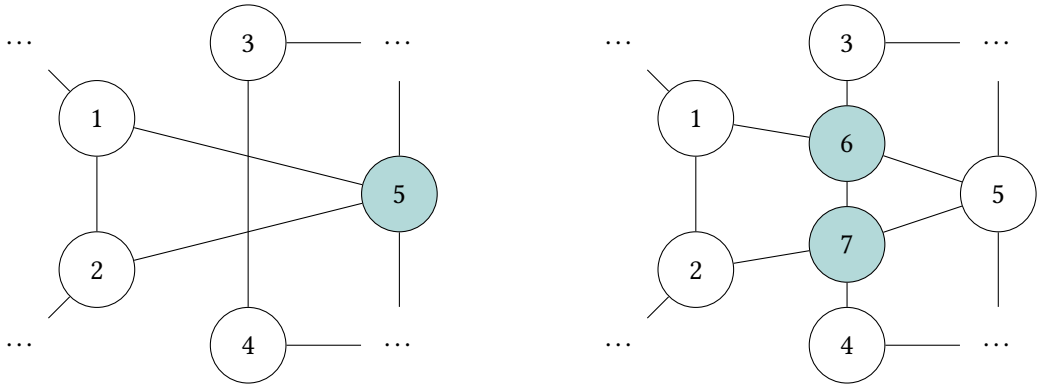


Figure 4.8: Visualization of a top-level separator for the road network of Karlsruhe. Vertices colored teal represent the separator nodes identified within the original, non-planar graph. Orange vertices indicate the additional nodes required to establish a valid separator for the planarized version of the network. The single teal vertex on the right is a highway intersection whose inclusion is necessary for the separator.

(a) Separator in non-planar graph.

(b) Separator in planarized graph.

Figure 4.9: Example of a separator that decreases in size after planarization.



(a) Separator in non-planar graph.

(b) Separator in planarized graph.

Figure 4.10: Example of a separator that increases in size after planarization.

4.4 Hierarchy

Real-world road networks 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:

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

4.5 Diameter and Distance Distributions

The diameter of a graph, defined as the longest shortest path between any pair of vertices, is a fundamental metric characterizing its overall extent and the efficiency of traversal. In our analysis, we focus specifically on the hop diameter, where each edge has a uniform weight of 1. This choice is motivated by the desire to understand the graph's topological and hierarchical structure. To ensure that the measured hop distances are structurally meaningful, we first pre-process all graphs and subgraphs by contracting vertices of degree 2. This effectively treats long chains of such vertices as single conceptual edges.

Computing the exact diameter of a general graph can be computationally intensive. A simple two-step Breadth-First Search (BFS) approach can be used. This involves performing a BFS from a random node to find the furthest vertex, and then a second BFS from that resulting vertex. However, this method is only guaranteed to find the exact diameter in certain graph classes, such as trees, and provides at best a 2-approximation for general graphs [ACIM99]. Therefore, to obtain exact diameter values, we employ the more sophisticated iFUB (iterative Fringe Upper Bound) algorithm as described by Crescenzi et al. [Cre+13]. This algorithm successfully computes the exact graph diameter in reasonable time on road networks without resorting to a full all-pairs shortest path calculation.

We investigate how the hop diameter scales with graph size by analyzing subgraphs obtained from the nested dissection process. Our empirical analysis of these contracted subgraphs indicates that their diameter scales approximately as $\mathcal{O}(n^{1/3})$. This observed scaling behavior is illustrated in Figure 4.11.

Beyond the maximum path length (diameter), we also analyze the complete distributions of both weighted distances and hop counts to understand typical path lengths. Computing all-pairs shortest paths is computationally prohibitive for large graphs. Therefore, we approximate these distributions by sampling 10^4 source nodes and performing a one-to-all shortest path query from each. For the hop distance calculation, the graph is pre-processed by contracting all degree-2 vertices, consistent with our diameter analysis. Figure 4.12 visualizes the resulting frequencies for both metrics.

The two histograms reveal a key distinction between the network's geographic and topological structure. The weighted distance distribution is strongly right-skewed with a smoothly decaying tail. In contrast, the hop count distribution exhibits a sharp peak at a low hop count and has a significantly heavier tail. We hypothesize that this heavy tail in the hop distribution

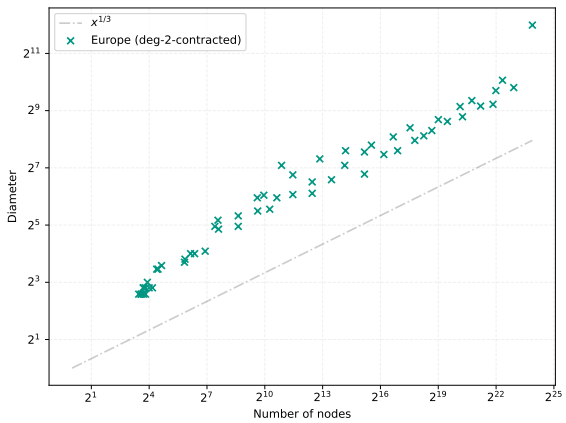
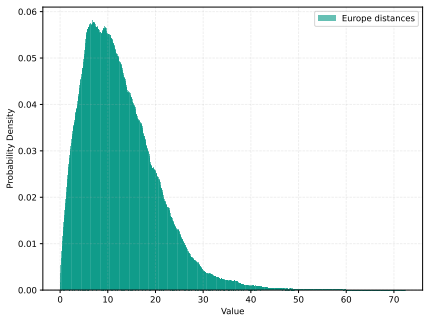
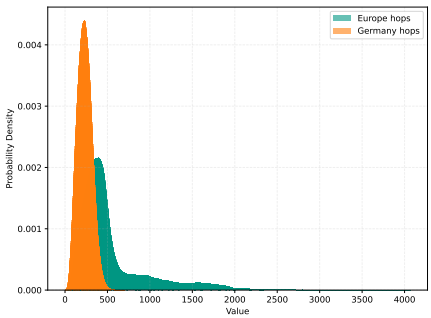


Figure 4.11: Empirical scaling of hop diameter with the number of vertices n for nested dissection subgraphs of road networks (with degree-2 nodes contracted). The observed scaling is approximately $\mathcal{O}(n^{1/3})$.

is not an intrinsic topological property of the road network's hierarchy, but rather an artifact of the non-convex, geographically elongated shape of the European continent. Even with a highly efficient network, paths between extremities, such as from Norway to Spain, are inherently long in terms of the number of intermediate junctions and segments. This hypothesis is supported by two key observations. First, the hop distribution for the more compact and convex graph of Germany does not exhibit a similarly heavy tail. Second, we performed an experiment where points were sampled uniformly at random within the geographic shape of Europe, and a edge length restricted Delaunay triangulation was constructed on these points. The hop distribution of this simple geometric graph also displays a very similar heavy tail. This strongly suggests that the observed heavy tail in the Europe road network's hop distribution is primarily a consequence of its large-scale geography, rather than a feature of its specific man-made topology.



(a) Distribution of weighted path distances for the Europe network.



(b) Comparison of hop distance distributions for the Europe and Germany networks.

Figure 4.12: Path length distributions approximated from 10,000 one-to-all queries. Left: Weighted distances for the DIMACS Europe road network. Right: Comparison of hop distributions for the Europe network versus the more compact Germany sub-network.

5 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.

5.1 Degree Distribution

Our initial focus is on isolating the effect of the degree distribution. To examine whether a low average degree 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 our empirical observations on road networks. This observed diameter growth of these trees is visualized in Figure 5.1.

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, similar to road networks, is reached. 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

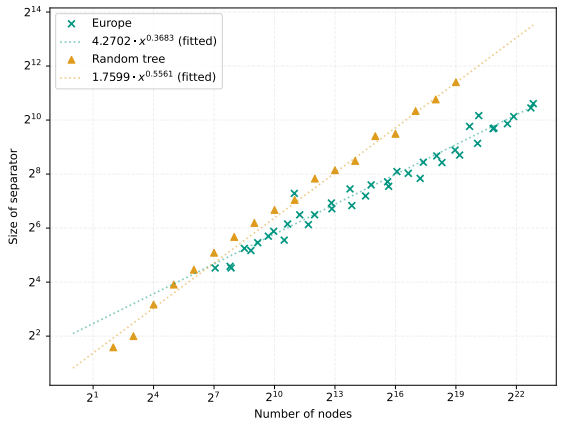


Figure 5.1: The plot shows the diameter (y-axis) as a function of the number of nodes n (x-axis, log scale).

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 5.2.

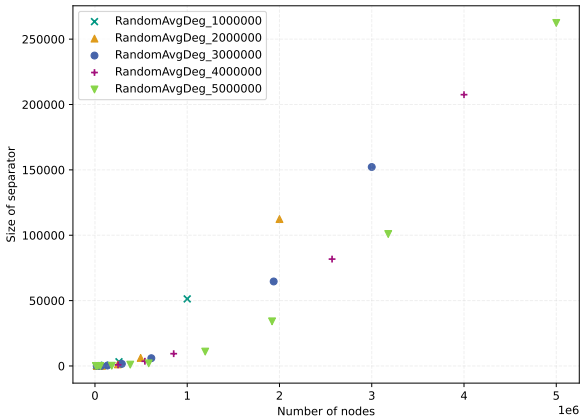


Figure 5.2: Separator sizes observed in random graphs generated with an average degree matching the DIMACS Europe road network ($\approx$ 2.5). Separators were computed using FlowCutter (tests with KaHIP yielded similar asymptotic scaling).

We also conducted experiments to generate graphs that match not just the average degree but also the specific degree distribution of a reference road network. This process involved starting with a random tree and then adding edges subject to constraints: an edge (u, v) was incorporated only if doing so would not cause either vertex u or v to exceed the degree count specified for them by the target distribution, which was sampled from the road network. The separator results from this more refined approach were largely similar to those from graphs matching only the average degree. It is important to note, however, that precisely replicating

a target degree distribution with this method is not possible. While the maximum degree in the generated graphs generally aligns with that of the reference network, discrepancies emerge, particularly for nodes with a higher degree. The underlying random spanning tree structure tends to produce a higher proportion of such nodes compared to actual road networks. For instance, in synthetic graphs approximately 1.8% of vertices had a degree of 5 or greater, exceeding the 0.2% observed in the real dataset. Thus, our process achieves an approximation of the target distribution. Nevertheless, we posit that this approximation is sufficiently close for our investigation, as the relatively small fraction of vertices with degrees deviating from the target constraints is unlikely to fundamentally alter the observed separator scaling behavior for this graph class. These findings reinforce the suggestion that the specific degree distribution, in isolation, is likely insufficient to fully explain the empirically observed small separators. Nevertheless, the low average degree remains a noteworthy property. While a lower average degree may not fundamentally alter the inherent asymptotic separator scaling characteristic of a specific graph class, it often correlates with smaller constant factors in the observed separator sizes, the practical effect of sparsity on separator magnitudes will be discussed further in a subsequent section. Beyond the influence of average degree on constant factors, it is also crucial to recognize that neither a low average degree nor a specific sparse degree distribution is a strict prerequisite for achieving small separators. Indeed, graphs can be constructed that possess both deliberately small separators (as a function of n) and a high average degree. Consider, for instance, the following recursive graph generation scheme designed to produce a graph on n vertices with a target separator of size $s(n)$ (e.g., $s(n) = n^{1/3}$): First, a clique S on $s(n)$ vertices, denoted $K_{s(n)}$, is designated. Then, two subgraphs, G_1 and G_2, each containing approximately $(n - s(n))/2$ vertices, are generated recursively by the same procedure (with a base case, such as returning a clique K_n, for small n, e.g., when $n \leq 2$). Finally, every vertex in G_1 is connected to every vertex in S, and likewise, every vertex in G_2 is connected to every vertex in S. By construction, the set S (forming a $K_{s(n)}$) is a separator of size $s(n)$ that partitions the graph into G_1 and G_2. Despite this small, built-in separator, the average degree of the resulting graph can be substantial. For example, using $s(n) = n^{1/3}$, a graph of $n = 10^4$ vertices generated via this method has an average degree of approximately 169 and a maximum degree of 6202. This example illustrates that specific structural properties enabling small separators can coexist with high overall edge density, suggesting that the global organization of edges, beyond mere sparsity, significantly influences separator characteristics.

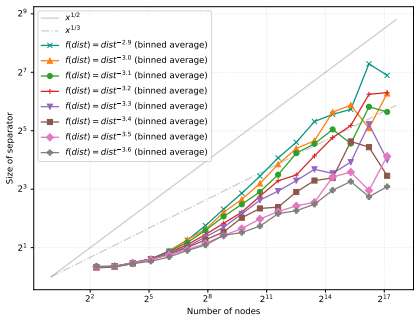
5.2 Locality

Our investigation into locality begins by exploring how locality defined on an existing network structure, instead of geometric proximity, influcences separator scaling.

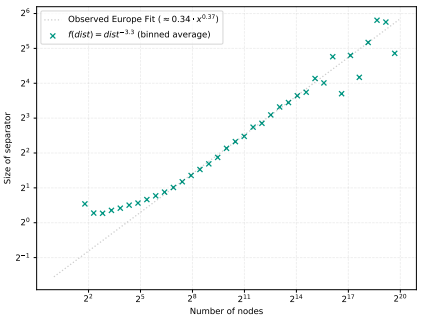
Tree Locality For this first approach, we simulate locality by considering distances within an initial random spanning tree, which serves as a foundational sparse connected graph. The premise is that vertices closer to each other along paths in this underlying tree are more likely candidates for new, direct connections, potentially reflecting how topological proximity in an existing network might drive the addition of new links. To implement this, 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 is achieved. While many experiments in this thesis use a target average degree of approximately 2.5 to emulate road networks, a different value is chosen for this specific model. The choice of average

degree often just influences the constant factors of the separator scaling, rather than the underlying asymptotic growth of the graph class. In other contexts, an average degree of 2.5 effectively approximates the constant factors of road network separators. For this tree-distance-based locality model, however, an average degree of approximately 3 is required to achieve a comparable alignment of these constant factors.

To incorporate locality, the probability of adding an edge between two vertices u and v is made dependent on their distance within the initial spanning tree T, $\text{dist}_T(u, v)$. 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. For each randomly selected vertex x and potential neighbor y, a Breadth-First Search (BFS) is performed 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 prove computationally intensive for larger graphs. However, the approach is highly parallelizable since each BFS is an independent computation on the fixed tree structure. Capping the BFS search for y (e.g., after a fixed number of hops or visited nodes, such as 50,000), while significantly reducing computational overhead did not achieve the desired locality effect. We observe an artificial steep cutoff in the observed separator characteristics once graph or component sizes effectively exceed the scale imposed by this search limit. Experiments exploring various decay functions f yield diverse results. Simple functions, like $f(\text{dist}) = 1/\text{dist}$, result in graphs that still exhibit large top-level separators scaling as $\mathcal{O}(n)$, although subgraphs from nested dissection show the previously noted superlinear decrease in separator size. Conversely, rapidly decaying functions like $f(\text{dist}) = 2^{-\text{dist}}$ produce graphs with separators of almost constant size, likely because the graph structure remains very close to that of the initial tree, with insufficient long-range connections. Through further experimentation and fine-tuning, it is found that a function of the form $f(\text{dist}) = 1/\text{dist}^{3.3}$ is able to produce separator sizes that closely approximate the desired $\mathcal{O}(n^{0.37})$ scaling (see Figure 5.3b). Figure 5.3 provides a visual comparison of the separator sizes for various decay functions.



(a) Separator scaling for various decay functions f.



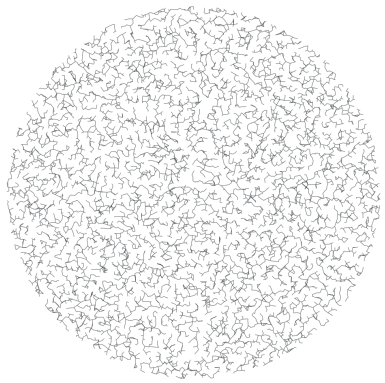
(b) Separator scaling for $f(\text{dist}) = 1/\text{dist}^{3.3}$ with line fit.

Figure 5.3: Separator size scaling for graphs generated with tree-distance-based locality, comparing various decay functions and highlighting the fit for $f(\text{dist}) = 1/\text{dist}^{3.3}$. Separators were computed using FlowCutter (tests with KaHIP yielded similar asymptotic scaling).

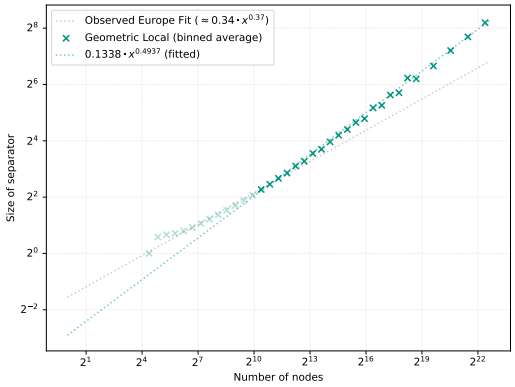
Despite achieving the target scaling with $f(\text{dist}) = 1/\text{dist}^{3.3}$, this result offers limited insight into the intrinsic structural properties of real road networks that lead to their small separators. The success with this specific, empirically derived exponent appears to be more a consequence of parameter tuning within this particular generative model (a random tree backbone augmented by edges based on tree distance) rather than an explanation of fundamental graph characteristics. While the general concept of incorporating locality based on distances within an existing network structure remains potentially insightful, as such a metric might capture aspects beyond mere Euclidean distance, like travel time reduction incentives, our preliminary results with simpler functions indicate a high sensitivity to the choice of f. Achieving the target $\mathcal{O}(n^{0.37})$ scaling with an empirically derived function like $f(\text{dist}) = 1/\text{dist}^{3.3}$ appears to be an outcome of model-specific parameter tuning rather than a revelation of intrinsic road network properties responsible for their small separators. Given these considerations, we opt not to pursue an exhaustive search for an optimal or more interpretable tree-distance decay function within the scope of this work.

Geometric Locality 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 locality, 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 in $\mathbb{R}^2$). 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. 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 5.4.

An alternative strategy we investigate for incorporating geometric locality is the construction of k-Nearest Neighbor (k-NN) graphs. In this model, each vertex connects via an edge to its k geographically closest neighbors, with proximity defined by the Euclidean distance between their embedded point coordinates. This approach does not utilize an underlying tree structure like some previously discussed methods. Connectivity in a k-NN graph is not guaranteed, however, the size of the largest connected component generally increases with the value of k. For instance, with uniformly sampled points, a k as small as 3 often yields a primary connected component encompassing approximately 98% of all vertices. The average degree of the resulting graph is directly influenced by k. For point sets with non-uniform distributions, a larger k is typically required to achieve a similar level of overall connectivity. Our objective is to identify the smallest k that results in a largely connected graph, while also aiming for a low average degree, comparable to those of sparse road networks. The



(a) Visualization of a graph with 10^4 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 separators.

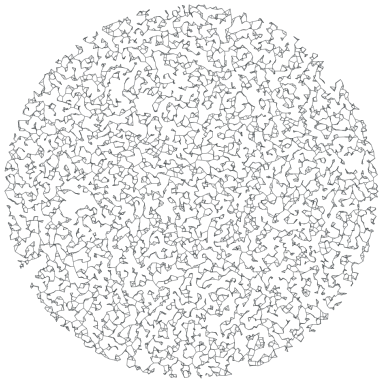
Figure 5.4: Synthetic graph generation using geometric locality and analysis of separator sizes. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

construction of k-NN graphs can be performed efficiently: following an initial $\mathcal{O}(n \log n)$ precomputation to build a spatial index such as a k-d tree, identifying the k nearest neighbors for each of the n vertices takes approximately $\mathcal{O}(k \log n)$ time per vertex, resulting in a total construction time of roughly $\mathcal{O}(nk \log n)$. Our experiments show that k-NN graphs generated in this manner also exhibit separator sizes that scale approximately as $\mathcal{O}(n^{1/2})$ as seen in Figure 5.5.

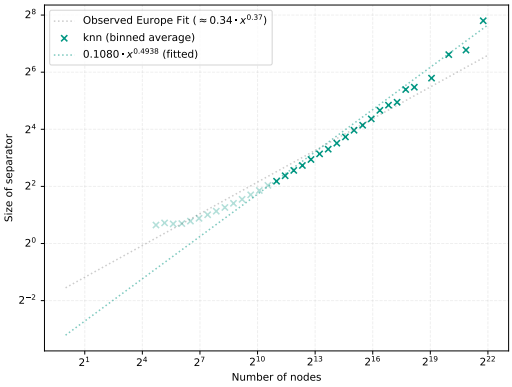
The results from the geometric locality approach suggest that merely combining low average degree with this specific form of locality is insufficient to reproduce the cubic root separator sizes of road networks.

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 $c_2 \ll c_1$, 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 Our synthetic graphs generated using the geometric locality approach exhibit a notable initial deviation in their separator scaling at small graph sizes. As visualized in the histogram in Figure 5.6, there is an initial peak in relative separator size for small subgraphs, which then decreases before the data align with the more dominant scaling trend observed for larger graphs. This behavior is similar to the initial scaling deviations present in real road networks. This correspondence is noteworthy, as it



(a) Visualization of a k-NN graph (for $k = 3$) generated from $n \approx 10^4$ uniformly sampled points.



(b) Separator size scaling for k-NN graphs $k = 3$, showing approximately $\mathcal{O}(n^{1/2})$ behavior.

Figure 5.5: Analysis of k-Nearest Neighbor (k-NN) graph of uniformly sampled points and separator scaling. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

suggests that our geometric locality model, despite failing to replicate the overall asymptotic separator scaling, may capture certain structural properties that are relevant at smaller graph scales and also present in actual road networks.

Minimum Spanning Trees from Higher-Dimensional Embeddings Observing that real road networks often exhibit diameters smaller than the $\mathcal{O}(n^{1/2})$ found in our random tree models. We sought to determine if an initial tree structure with an intrinsically smaller diameter could lead to graphs with correspondingly smaller separators after edge augmentation. This motivated an exploration of generating initial trees as Minimum Spanning Trees (MSTs) of points sampled in d-dimensional Euclidean space, since the diameter of such an MST on n random points in d-space (for $d \geq 2$) is typically $\mathcal{O}(n^{1/d})$. Our experiments focused on the case where $d = 3$, for which the MST diameter is approximately $\mathcal{O}(n^{1/3})$. This involved sampling n points uniformly in 3D space and computing their MST. When edges were subsequently added to such 3D-MSTs to achieve a target average degree, the resulting graphs exhibited separators that scaled as $\mathcal{O}(n^{2/3})$. This scaling aligns with a general geometric intuition: a d-dimensional hypervolume containing n points is naturally bisected by a $(d-1)$-dimensional separator. If the characteristic linear extent of the volume is proportional to $n^{1/d}$, the $(d-1)$-dimensional measure (e.g., area for $d = 3$, length for $d = 2$) of such a separator would be proportional to $(n^{1/d})^{d-1} = n^{(d-1)/d}$. For $d = 2$, this yields the familiar $\mathcal{O}(n^{1/2})$ scaling for planar-like separators, and for our $d = 3$ experiment, it corresponds to the observed $\mathcal{O}(n^{2/3})$. While this suggests a general hypothesis that augmenting MSTs from d-dimensional point sets might lead to graphs with $\mathcal{O}(n^{(d-1)/d})$ separators, this avenue was not pursued further. Since for $d = 3$ the $\mathcal{O}(n^{2/3})$ separators do not offer an improvement over the $\mathcal{O}(n^{1/2})$ separators of planar graphs in the context of finding exceptionally small separators (and higher dimensions $d > 3$ would yield even larger exponents as $(d-1)/d \to 1$), this specific line of inquiry was not extended.

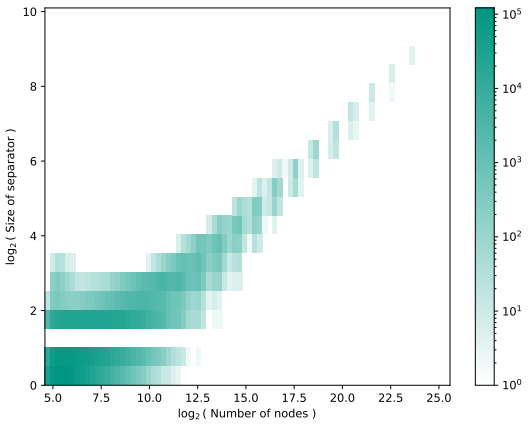


Figure 5.6: Histogram of separator sizes for synthetic graphs using geometric locality, illustrating an initial peak in relative separator size for small subgraphs (around 2^6 nodes). Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

5.3 Planarity

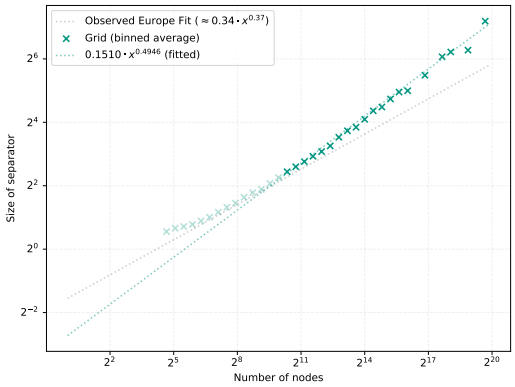
We examine separator properties in two classes of planar graphs: grids and Delaunay triangulations. These serve as fundamental theoretical models for planar structures relevant to the study of road networks due to their near-planarity. In our study, these graph classes are sparsified to achieve a low average degree (approximately 2.5), reflecting the sparsity observed in road networks. It is important to note that both of these models inherently introduce strong geometric locality, a property not required by all planar graphs. Our focus on these specific classes therefore introduces a bias towards investigating planar graphs that also possess this structural characteristic.

5.3.1 Grids

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 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 5.7 illustrates a sample sparse grid and the observed separator scaling.

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



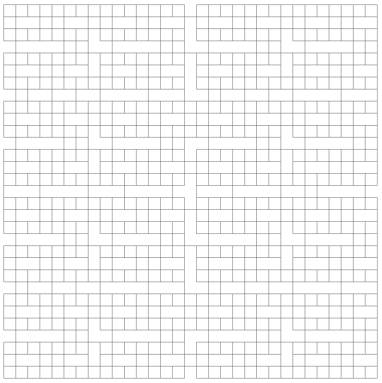
(b) Separator size scaling for sparse grid graphs (average degree 2.5). Separator sizes scale approximately as $\mathcal{O}(n^{1/2})$, but with a small constant factor leading to absolute sizes comparable to road networks at practical scales. The fit shown excludes subgraphs with $\leq$ 1000 vertices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

Figure 5.7: Analysis of sparse grid graphs with average degree 2.5. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

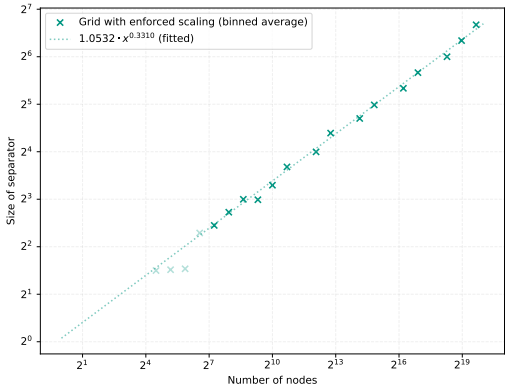
Analogous to the earlier example where dense graphs were constructed with a specific separator function, it is also possible to engineer grid-like graphs to exhibit a target separator scaling, such as $\mathcal{O}(n^{1/3})$. This approach involves a recursive construction: two smaller grid-like subgraphs, G_1 and G_2, each comprising approximately $n/2$ vertices, are generated recursively. These subgraphs are then interconnected. Instead of forming dense connections or connecting all border vertices, a controlled number of edges, specifically chosen to match the desired separator size (e.g., approximately $n^{1/3}$ for the combined graph of size n), are added between G_1 and G_2. For instance, this can be achieved by selecting approximately $n^{1/3}$ nodes along the boundary of one subgraph and connecting each to its closest corresponding node on the boundary of the other. The set of these interconnecting edges (or one of their incident vertices) is, by construction, a separator of the desired $\mathcal{O}(n^{1/3})$ size. An example of such a graph and its resulting separator scaling is illustrated in Figure 5.8. While this method demonstrates that specific separator sizes can be achieved by deliberate construction in grid-like structures, it offers limited insight into which intrinsic graph properties of real road networks lead to their empirically observed small separators. The desired scaling is explicitly engineered into the generation process here, rather than emerging naturally from more fundamental characteristics.

5.3.2 Delaunay Triangulations and Their Sparse Subgraphs

As a generalization of grid-like structures derived from point sets, Delaunay Triangulations (DT) are fundamental in computational geometry and serve as a basis for generating planar graphs. A DT for a given set P of discrete points in a plane is a specific triangulation such that



(a) A recursively constructed grid with engineered $\mathcal{O}(n^{1/3})$ interconnections.



(b) Separator scaling for the engineered grid graph, exhibiting the constructed $\mathcal{O}(n^{1/3})$ behavior.

Figure 5.8: Grid-like graph with engineered separators designed to scale as $\mathcal{O}(n^{1/3})$. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

for any triangle in $DT(P)$, its circumcircle (the unique circle passing through its three vertices) contains no other points from P in its interior. This "empty circumcircle" property ensures that DTs maximize the minimum angle of all triangles in the triangulation, thereby avoiding overly elongated triangles. Our process for generating a baseline Delaunay graph involves sampling n points uniformly at random in a two-dimensional space and then computing their full Delaunay triangulation. Such a DT typically has an average vertex degree around 6. This observation is consistent with Euler's formula for planar graphs ($|V| - |E| + |F| = 2$), for large triangulations, where most faces are triangles and each internal edge is shared by two faces ($3|F| \approx 2|E|$), the average degree $2|E|/|V|$ approaches 6.

Since an average degree of 6 is considerably denser than that of typical road networks (which is closer to 2.5), sparsification is necessary to create more comparable synthetic graphs. We explore several methods for this:

Random Edge Deletion One straightforward sparsification technique, similar to that applied in our grid experiments, involves randomly deleting edges from the full Delaunay triangulation until the target average degree of approximately 2.5 is achieved. The largest connected component of the resulting graph is then used for analysis.

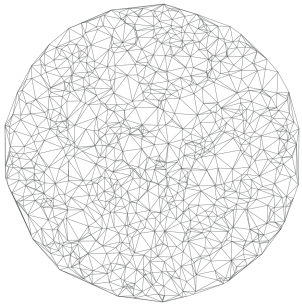
Systematically Defined Subgraphs Alternatively, sparser graphs can be derived from the DT by retaining only those edges that satisfy stricter geometric conditions. Prominent examples of such Delaunay subgraphs include Gabriel Graphs (GG) [GS69] and Relative Neighborhood Graphs (RNG) [Tou80]. An edge (u, v) is part of a GG if the closed disk whose diameter is the segment uv contains no other point from the original set P. An edge (u, v) belongs to an RNG if no third point $w \in P$ is simultaneously closer to both u and v than u and v are to each other, that is, the lune formed by the intersection of two circles of radius $\text{dist}(u, v)$ centered at u and v must be empty of other points. These definitions lead to the known hierarchical relationship: RNG $\subseteq$ GG $\subseteq$ DT. The RNG is particularly interesting from a network modeling perspective, as its construction criterion can be loosely interpreted in terms

of economic viability for infrastructure: a direct road between two points u and v might not be constructed if an alternative path via a nearby third point w (e.g., $u \to w \to v$) exists without a significant detour. For uniformly sampled random points, the RNG yields an average degree of approximately 2.6 [Buh+06], which is close to the average degree of road networks.

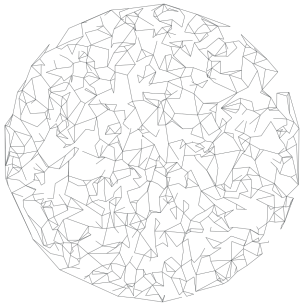
Separator Analysis of Delaunay Variants We analyzed the separator properties of these different Delaunay-derived graphs: the full DT, the randomly sparsified DT (target average degree 2.5), the Gabriel Graph, and the Relative Neighborhood Graph, all generated from identical initial point sets to ensure comparability. Visualizations of these graph structures and their comparative separator scaling are presented in Figure 5.9. Despite the variations in density, ranging from an average degree of approximately 6 for the full DT down to about 2.6 for the RNG, our experiments indicate that the asymptotic separator scaling for all these Delaunay-derived planar graphs remains consistent with $\mathcal{O}(n^{1/2})$. The method of sparsification, whether through random edge deletion or by applying systematic geometric rules like those for GG or RNG, primarily influences the constant factors associated with the separator size. However, it does not fundamentally alter the $\mathcal{O}(n^{1/2})$ scaling behavior tied to their underlying planar geometric nature. Even when the average degree matches that of road networks (as in the RNG case), the separator scaling does not approach the $\approx \mathcal{O}(n^{0.37})$ observed empirically for road networks.

5.4 Highway Dimension

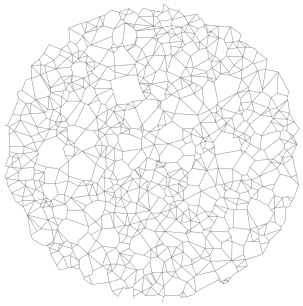
Given that the simpler graph models explored in previous sections do not reproduce the observed $\mathcal{O}(n^{0.37})$ separator scaling of road networks, we turn our attention to more complex generative processes designed to capture other relevant structural properties. One such property is highway 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]. The generation process, based on the description in [BKMW10], 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 (in our case a disk in $\mathbb{R}^2$) 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



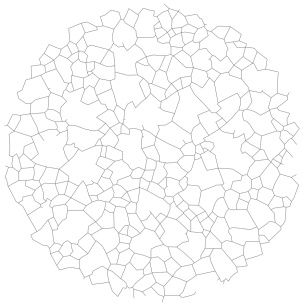
(a) Full Delaunay Triangulation (avg. deg. ≈ 6)



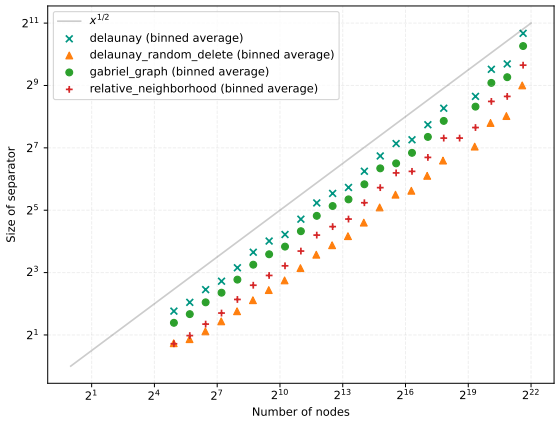
(b) Sparse DT (Random Deletion, avg. deg. ≈ 2.5)



(c) Gabriel Graph (GG)



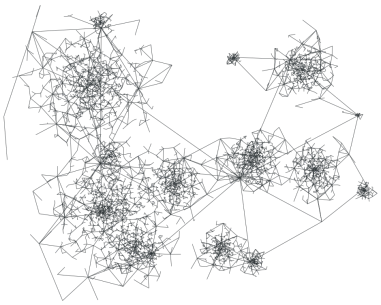
(d) Relative Neighborhood Graph (RNG)



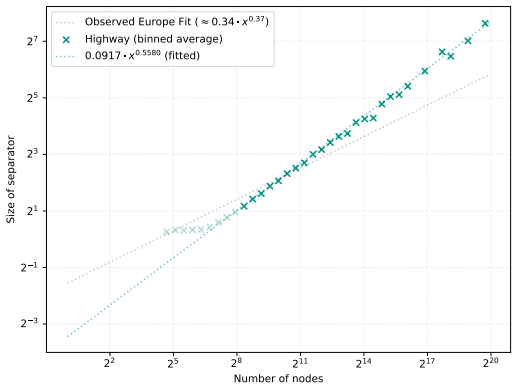
(e) Separator scaling for Delaunay variants.

Figure 5.9: Comparison of Delaunay Triangulation and its sparse subgraphs derived from the same point set: visualizations (a-d) and separator scaling (e). Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

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 5.10 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 10^4 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 separators.

Figure 5.10: Analysis of synthetic graphs generated using the ABR algorithm [AFGW10]. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

5.5 Hierarchical Structure

In the following sections, we explore four distinct approaches to generating synthetic graphs with hierarchical structures.

5.5.1 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

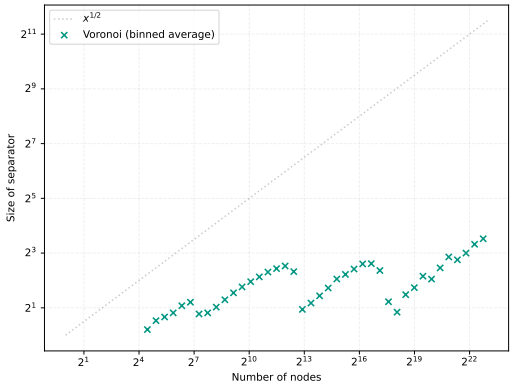
extent of the synthetic network. Points, designated as sites P, are then generated within this polygon according to a chosen sampling strategy. One strategy involves sampling a specified number of points uniformly at random throughout the polygon's interior. Alternatively, to better emulate settlement 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. The vertices of the graph correspond to the points where 3 or more Voronoi regions meet. 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 decreasing 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]. For completeness, we briefly mention an alternative pruning strategy for a different generative model that will be detailed in a later section (Section 5.5.3). This method tends to produce graphs with greater visual resemblance to real-world road networks than the Bauer et al. approach, though this comes at a higher computational cost.

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) cannot 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 $\mathcal{O}(n^{1/2})$. Figure 5.11 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.

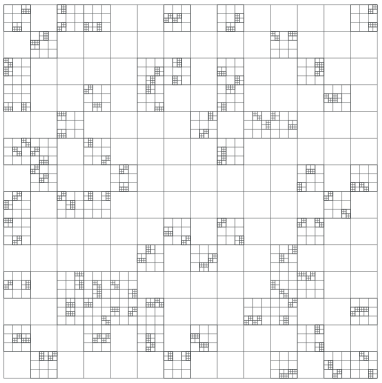
Figure 5.11: Analysis of synthetic graphs generated using the hierarchical Voronoi approach from Bauer et al. [BKMW10]. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

5.5.2 Nested Grids

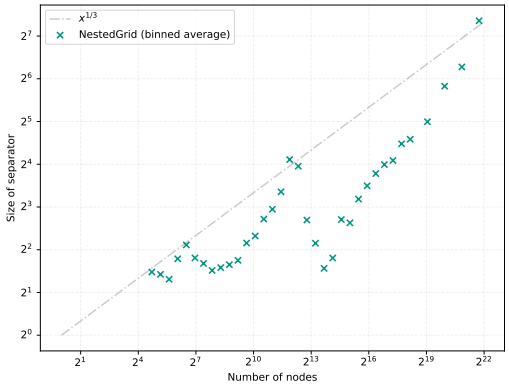
To explore hierarchical effects in a more controlled manner, we also investigate a simpler but conceptually related approach using nested grids. This method aims to construct a hierarchical graph by recursively embedding grid graphs within the cells of a parent grid, potentially allowing us to deliberately engineer specific separator scaling properties. The fundamental idea is recursive: a level-1 structure is a standard square grid. A level-i nested grid (for $i > 1$) is then constructed from a higher-level parent grid, where selected cells are replaced by entire instances of level-$(i-1)$ nested grids. This process can be repeated for any number of layers, creating a fractal-like structure of grids within grids. An optional refinement can be applied to make the graph sparser by removing redundant edges from parent grids. An edge of a parent grid is considered redundant if the cell it borders has a subgraph embedded within it. This is because the boundary of the embedded subgraph provides a new, more detailed path between the endpoints of the parent grid edge. Removing these "overlapped" parent edges reduces the graph's overall edge density slightly without altering its asymptotic separator scaling. Figure 5.12 shows a small conceptual example of a nested grid with four layers.

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 estimate the total number of vertices of a subgraph of the nested grid structure V_{sub} based on the top level grid width w of the current layer, the number k of subgraphs placed within its cells, and the size s of each subgraph:

$$|V_{\text{sub}}| \approx w^2 + k \cdot s$$



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



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

Figure 5.12: Analysis of synthetic graphs generated using the nested grid approach. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

This estimate is approximate as it may double-count vertices along the boundaries where subgraphs connect to the parent grid. The estimate also assumes we are only looking at perfect nested grids, which is not true as we are looking at general subgraphs from a nested dissection. To enforce $\mathcal{O}(n^{1/3})$ scaling specifically at this layer transition, we can equate the separator size (approximated by w) with $|V_{\text{sub}}|^{1/3}$:

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

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 in Figure 5.12b 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 5.13 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.

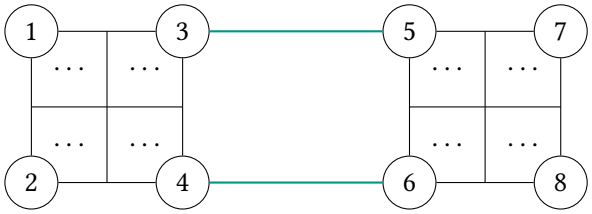


Figure 5.13: Simplified illustration of connectivity between components from adjacent hierarchical levels. The components (nodes 1-4 and 5-8) are connected only by the two green edges $\{3, 5\}$ and $\{4, 6\}$. Removing one endpoint from each edge (e.g., 3 and 6) forms a size-2 vertex separator, irrespective of component sizes.

5.5.3 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 of Bauer et al. [BKMW10]. A significant artifact observed in some Voronoi-based hierarchical models was that major transport arteries, analogous to motorways, could not be crossed by lower-level roads. This limitation arose because new expansion sites were generated strictly within the polygonal cells defined by the higher-level Voronoi tessellation, effectively making top-level Voronoi edges hard boundaries. Our revised core idea allows new centers of expansion 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 predefined areal polygons.

Initial Approach An initial concept explored an iterative Delaunay approach. This process would commence with a Delaunay triangulation of an initial point set. In subsequent hierarchical levels, a subset of existing points would be selected as *expansion sites*. New points would then be sampled in the vicinity of each of these sites. Following this, a new Delaunay triangulation would be performed independently for each expansion site. This iterative process is designed to emulate a self-reinforcing concentration dynamic, which is analogous to preferential attachment models. The rationale is to replicate how real-world settlement and infrastructure patterns evolve, where new growth is more likely to occur in or around already dense regions. By making points in these areas more probable candidates for subsequent expansion, the model naturally gives rise to a hierarchical landscape of concentrated clusters, mimicking the formation of urban centers. This recursive generation would continue for a specified number of levels. The result at this stage is a collection of disjoint, small, triangulated graphs embedded in the plane, whose edge lengths differ significantly based on the geometric scale at which they were generated.

As a final step, a planarization process, as described in Algorithm 4.1, is applied: edges, interpreted as straight line segments, are checked for intersections. Any intersections found are resolved by introducing new vertices at these points and subdividing the original edges

accordingly, thus making the graph connected and planar. This method showed initial promise in creating structures that could yield separators with scaling behavior similar to that of road networks.

Refined Generation Building upon insights from this strategy, we developed a more streamlined method, which is the primary focus of this section. Instead of performing Delaunay triangulations at each hierarchical level followed by a final planarization step, we first generate the complete set of points across all desired levels. Only after all points have been generated a single, global Delaunay triangulation is performed on the entire final point set. The pseudocode for this hierarchical Delaunay generation process is detailed in Algorithm 5.1. The method `uniformRandomPointsInCircle(center, radius, k)` samples k points uniformly at random within a circle of given radius centered at the specified point. A deliberate design choice in our hierarchical point generation process (Algorithm 5.1) is that candidates for new expansion sites at each level i are selected from the entire set of currently existing points P, rather than exclusively from points generated in the immediately preceding level $i-1$. This approach reflects the real-world phenomenon where new settlements, even smaller ones, can emerge around and connect to established, major infrastructure hubs that might have formed at much earlier stages of network growth, for instance, a small town developing near a major motorway interchange. We also experimented with an alternative strategy where expansion sites for a given level were chosen only from the points generated in the previous level. However, this restriction did not yield any noticeable differences in the overall structural properties or separator characteristics of the resulting graphs.

Algorithm 5.1: 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,
for $1 \leq i \leq L$

Output: Geometric graph G

```
1 P ⟵ {p_random}
2 forall i ∈ [1, L] do
3     C_i ⟵ P.chooseRandom(⌊f_i · |P|⌋)
4     forall center ∈ C_i do
5         P ⟵ P ∪ uniformRandomPointsInCircle(center, r_i, k_i)
6 G ⟵ delaunay(P)
7 pruneEdges(G)
8 return G
```

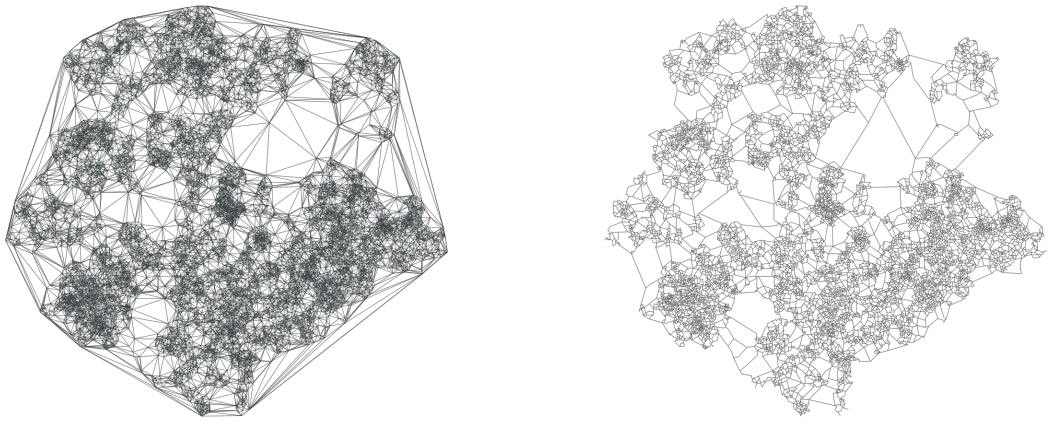
Pruning Analogous to the Delaunay triangulation approach in Section 5.3.2, the global Delaunay triangulation performed in Algorithm 5.1 results in a graph with an average vertex degree of approximately 6, which is denser than typical road networks. To better emulate the characteristics of real road networks, such as an average degree around 2.5 and fewer nodes with very high degrees, we apply an edge pruning step. While Bauer et al. [BKMW10] also

suggested a pruning methodology, we adopt a different strategy tailored to our generated graph structures. In our approach, edges are considered in order of increasing Euclidean length (shortest first). An edge (u, v) is removed if the length of the shortest path between its endpoints in the remaining graph does not exceed its original direct length by more than a factor of α, i.e. if $\text{dist}_{G\setminus\{(u,v)\}}(u, v) \leq \alpha \cdot \text{length}(u, v)$. This edge removal criterion approximates economic viability considerations, as connections that offer marginal utility or are largely redundant would likely not be constructed in real-world road networks. Through experimentation, we found that a pruning parameter of $\alpha = 2.5$ effectively reduces the average degree to the target range comparable to that of road networks. The pseudocode for this edge pruning process is given in Algorithm 5.2. A visual comparison of graph structures before and after pruning is provided in Figure 5.14.

Algorithm 5.2: Edge pruning based on path length redundancy.

Input: Graph $G = (V, E)$,
Pruning parameter α
Output: Pruned version of G

```
1 forall (u, v) ∈ E sorted by increasing length do
2     if dist_{G\{(u,v)}}(u, v) ≤ α · length(u, v) then
3         G ⟵ G \ {(u, v)}
```



(a) Graph structure before pruning.

(b) Graph structure after pruning.

Figure 5.14: Visual comparison of a hierarchical Delaunay graph before and after edge pruning.

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 5.15 illustrates that the separator size scaling remains similar for both pruned and unpruned graphs.

Given that sequential pruning can be computationally intensive due to numerous shortest path computations (typically Dijkstra's algorithm, where path precomputation is infeasible as the graph changes), we also explored an approximate parallel version. In this variant, instead of looking at edges one by one, we process the edges in the same order as before, but

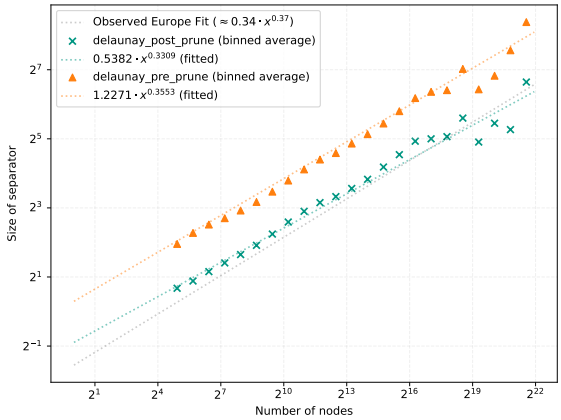


Figure 5.15: Comparison of separator size scaling in hierarchical Delaunay graphs before and after the pruning step. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

in chunks of size corresponding to the number of available processing threads. While this approach does not reduce the overall computational complexity, it allows the workload to be distributed across multiple threads. 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 simultaneously. This parallelization introduces an approximation: an edge (u, v) might be removed because its shortest alternative path relied on another edge (x, y) from the same chunk, which is also concurrently identified for removal. If (x, y) is removed, the true shortest path for (u, v) in the updated graph might have become longer than the $\alpha \cdot \text{length}(u, v)$ threshold, but this would not be detected as the checks were based on the graph state at the beginning of the chunk processing. In extreme cases, this could even lead to the graph becoming disconnected. 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 outcome is likely because edges are sorted by length before being chunked, thus, edges within a single chunk are often spatially distributed across the graph rather than being highly locally interdependent, minimizing negative interference from concurrent removals.

Parameterization The number of hierarchical layers, denoted L, is a key parameter for the generation process detailed in Algorithm 5.1. Similar to Bauer et al., we also found $L = 4$ to be a suitable choice for generating graphs with realistic road network properties [BKMW10]. For $L = 4$, the algorithm requires parameters for level expansion fractions f_i, points per expansion site k_i, and expansion radii r_i, for $i \in [1, L]$, totaling $3L$ (i.e., 12 for $L = 4$) parameters. However, the effective number of free parameters is lower. The specific value of the first expansion radius, r_1, primarily scales the entire embedding without fundamentally altering its topological structure, effectively reducing the count to 11 if r_1 is fixed or normalized. Furthermore, the algorithm specifies that the first level expansion fraction $f_1 = 1.0$ (ensuring the initial point always seeds the first level of expansion), reducing the free parameters to 10. If a target total number of vertices, n_{target}, is desired, one additional parameter (e.g., k_L) can be adjusted to meet this target, potentially leaving 9 free parameters for a 4-layer model.

The total number of points after level j, denoted n_j (where $n_0 = 1$ is the initial seed point, so $n_1 = k_1 + 1$ represents the total points after the first expansion), is given by the recurrence relation:

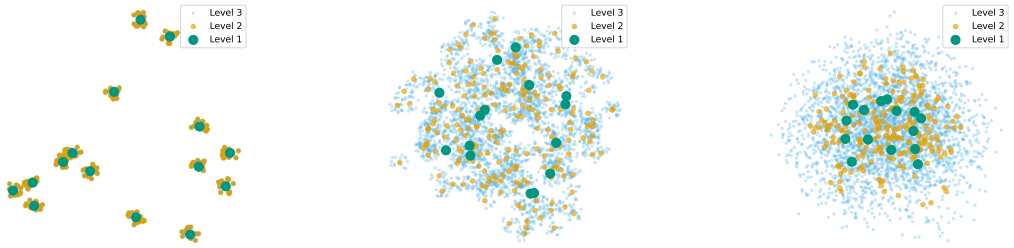
$$n_j = \begin{cases} k_1 + 1 & \text{if } j = 1 \\ n_{j-1} + n_{j-1} \cdot f_j \cdot k_j & \text{if } j > 1 \text{ and } j \leq L \end{cases}$$

For example, the total graph size n_4 for a 4-layer model is calculated as:

$$\begin{aligned} n_1 &= k_1 + 1 \\ n_2 &= n_1 + n_1 \cdot f_2 \cdot k_2 \\ n_3 &= n_2 + n_2 \cdot f_3 \cdot k_3 \\ n_4 &= n_3 + n_3 \cdot f_4 \cdot k_4 \end{aligned}$$

Parameter Interplay and Heuristics for Separator Properties Achieving road network-like separator scaling with the hierarchical Delaunay generator involves a nuanced interplay between the expansion fractions f_i, points per site k_i, and expansion radii r_i. A critical aspect for developing well-behaved separators appears to be ensuring sufficient interaction or *overlap* between regions expanded from different sites. This helps to avoid overly simplistic connections between hierarchical components, which can lead to pronounced local minima in separator scaling plots. We observe that decreasing expansion radii r_i contributes to desirable separator properties. The median edge length in a Delaunay triangulation of n uniformly random points within a region of radius r scales as $\Theta\left(r \cdot n^{-1/2}\right)$. In order for the nested structures to contribute to geometric locality the radii r_i should also decrease in a similar manner to approximate the median edge length. For example, constant radii would not yield any geometric locality and points would look uniformly distributed, fading out towards the edge of the graph. Our empirical observations suggested that without an exponential decrease in radii across levels, achieving consistently small separators is not possible. However, if radii become too small too quickly, the graph might develop insufficient connectivity between clusters originating from different parent sites. Consider a thought experiment: if each expansion site has an infinitesimally small influence radius for generating points for the next level, these new points primarily connect among themselves and back to their parent site from the higher level. The number of points on the effective "perimeter" of such an expanded site (those able to connect to other, distant parts of the graph) may grow much more slowly than the number of points generated internally, thereby keeping its external connectivity limited, this effect is amplified by the hierarchical sampling which tends to concentrate points internally. In such a scenario, one might find a separator for the graph defined by the top-level points (e.g., n_1 points after the first level) with a size scaling as $\mathcal{O}(n_1^{1/2})$. If each of these top-level separator nodes is a gateway to a large hierarchical substructure that is only sparsely connected back to this top-level framework (via few "perimeter" connections or the gateway node itself), the separator for the entire graph G_L might be estimated based on this top-level cut, extended by these few perimeter vertices. Given that the total number of nodes n_L is typically much larger than n_1, such a separator (e.g., related to a $\mathcal{O}(n_1^{1/2})$ scaling) would likely be disproportionately small for n_L. This issue can also manifest at subsequent levels. An attempt to achieve a desired global $\mathcal{O}(n_L^{1/3})$ scaling can involve setting parameters so that the initial $\mathcal{O}(n_1^{1/2})$ separator scaling corresponds to this overall target. This, however, can lead to a different problem: distinct top-level expansion sites, even if spatially close, might end up connected by very few paths. This sparse interconnectivity

between major components results in the re-emergence of pronounced local minima in the separator scaling plots at transitions between hierarchical levels, an issue similar to that encountered in simpler nested grid or Voronoi generation schemes where components are too easily fractured. Therefore, for generating graphs with well-behaved separators, it is critical that the expansion areas originating from different sites *overlap* or interact sufficiently to build an integrated and robustly connected graph structure, rather than a collection of dense clusters weakly linked through a sparse higher-level skeleton. The critical nature of this balance is illustrated in Figure 5.16. Using a radius decay that is too aggressive (too small) results in the previously discussed local minima, where the graph fractures along sparse inter-level connections (Figure 5.16a). Conversely, using constant or too slowly decaying radii leads to excessive overlap that overwhelms the hierarchical structure, causing the separator scaling to revert to $\mathcal{O}(n^{1/2})$ (Figure 5.16c). Only a well-tuned, approximately exponential decay of radii yields the desired consistent scaling behavior (Figure 5.16b).



(a) Too rapid radii decay. **(b)** Well-tuned radii decay. **(c)** Constant radii.

Figure 5.16: Illustration of the impact of different radii decay strategies on separator scaling. All other parameters held constant.

The degree of spatial overlap between regions expanded from different sites is fundamental to achieving desirable separator characteristics, as it dictates the strength of connectivity between hierarchical components. The parameters f_i, k_i, and r_i jointly determine this overlap: A higher expansion fraction f_i results in a greater number of nodes from the parent level being selected as expansion sites. This reduces the average distance between chosen sites, causing their respective areas of influence for the current level to naturally adjoin or overlap significantly. Increasing k_i, the number of points generated per expansion site, primarily makes each local cluster denser. Such high local density implies that partitioning these individual dense regions themselves requires larger separators, reflecting their increased internal complexity. Simultaneously, by populating the same expansion radius r_i with more points, the average distances between points decrease on this level, increasing the likelihood of inter-cluster connections. Compared to the influence of f_i and r_i, the effect of k_i is minor. The effect of the expansion radius r_i is direct: a larger radius straightforwardly increases the spatial extent of each new cluster, leading to more substantial overlap with adjacent expansion zones and, consequently, also contributing to larger overall separators due to increased linkage. Radii at deeper hierarchical levels must also be scaled appropriately relative to those at higher levels to ensure desired structural properties and avoid unintended consequences on separator sizes. Therefore, a careful balance of these parameters, f_i, k_i, and r_i, is essential to cultivate a graph structure with the intended connectivity and separator properties. The parameterization offers flexibility. For instance, to simulate prominent “urban centers” at

higher levels, a larger initial radius r_1 might be employed. This would typically be balanced by adjusting subsequent parameters, such as reducing f_2 (selecting fewer centers from the points generated by these large initial expansions), to maintain overall structural integrity and desired separator characteristics. However, such compensations have practical limits, if radii do not shrink sufficiently across levels, adjustments to f_i or k_i may not fully mitigate adverse effects on separator scaling. For generating structures visually akin to road networks, allowing the number of active expansion sites to decrease at deeper hierarchical levels often yields favorable results. This complements the inherent tendency of the generation process where all existing points are candidates for seeding new expansions, helping to manage density while still fostering hierarchical differentiation. Plausible structures can emerge even with constant k_i values across levels, provided f_i and r_i are appropriately tuned. For instance, the parameter set comprising level expansion fractions $f_i = (1.0, 0.3, 0.2, 0.1)$, points per expansion site $k_i = (50, 50, 50, 50)$, and expansion radii $r_i = (1000, 424, 120, 17)$ produces graphs whose separator characteristics are illustrated in Figure 5.17.

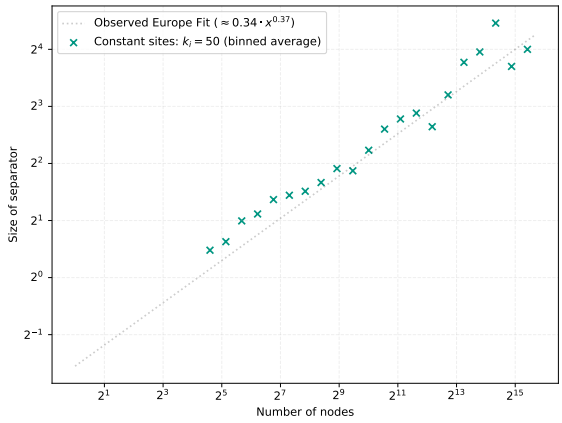
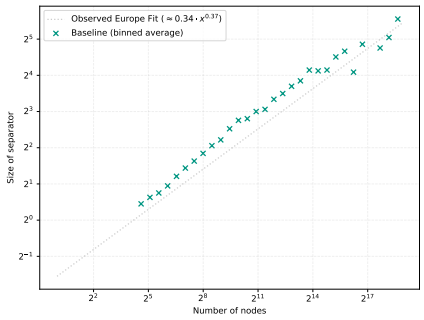
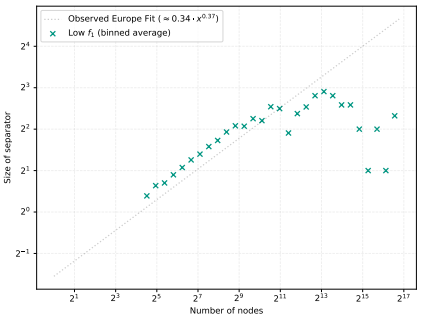


Figure 5.17: Separator size scaling for a hierarchical Delaunay graph generated with constant expansion sites: $f_i = (1.0, 0.3, 0.2, 0.1)$; $k_i = (50, 50, 50, 50)$; $r_i = (1000, 424, 120, 17)$. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

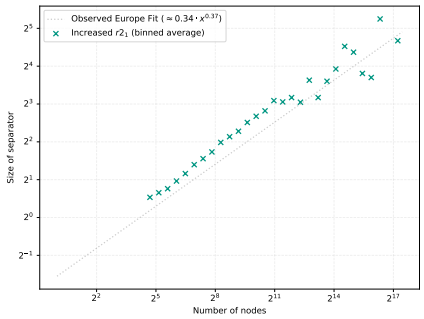
To further illustrate the interplay of these parameters, Figure 5.18 presents separator scaling plots for four different parameter configurations. The baseline configuration, shown in Figure 5.18a, is chosen to produce separators with characteristics similar to those observed in real road networks. Figure 5.18b then demonstrates the impact of significantly decreasing the expansion fraction for the second level (f_2) while other parameters are held constant relative to the baseline, this results in a notable drop in separator sizes, indicative of a less interconnected structure. Subsequently, Figure 5.18c shows that this reduction in separator size due to a low f_2 can be counteracted by proportionally increasing the expansion radius of the second level (r_2) and the subsequent radii (r_3, r_4). Alternatively, Figure 5.18d illustrates that increasing the number of points generated at the first level (k_1) can also compensate for the reduced f_2, again achieving separator scaling similar to the baseline by enhancing overall density and connectivity from the initial expansion phase.



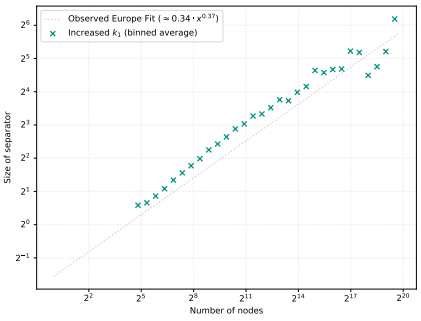
(a) Baseline: $f_i = (1.0, 0.5, 0.4, 0.3)$; $k_i = (200, 50, 35, 25)$; $r_i = (1000, 141, 26, 6)$.



(b) Low f_2: $f_i = (1.0, 0.1, 0.4, 0.3)$; $k_i = (200, 50, 35, 25)$; $r_i = (1000, 141, 26, 6)$.



(c) Fix 1 (Increase radii for low f_2): $f_i = (1.0, 0.1, 0.4, 0.3)$; $k_i = (200, 50, 35, 25)$; $r_i = (1000, 283, 52, 11)$.



(d) Fix 2 (Increase k_1 for low f_2): $f_i = (1.0, 0.1, 0.4, 0.3)$; $k_i = (1000, 50, 35, 25)$; $r_i = (1000, 141, 26, 6)$.

Figure 5.18: Illustration of parameter interplay on separator size scaling in hierarchical Delaunay graphs. Graphs (b),(c) and (d) have a decreased f_2 compared to the baseline (a). Paramter changes to fix the scaling are highlighted green. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

5.5.4 Physical Barriers

To investigate the influence of physical features of varying scales on road network structure and separator sizes, ranging from large-scale elements like mountains or lakes to smaller features like rivers or parks, we explore a generative approach based on iterating Perlin noise functions across multiple scales. The core idea is to use procedurally generated noise to define regions that are less favorable for node placement, simulating natural or man-made obstacles. The foundational component of these approaches is Perlin noise, a type of gradient noise widely used for procedural texture generation [Per85]. Unlike value noise, which interpolates random values assigned to a grid, Perlin noise generates a pseudo-random, continuous field by interpolating between random gradients assigned to grid points. For any given point in space, its noise value is determined by its position relative to the surrounding grid points and the dot product with their corresponding gradient vectors. A key characteristic of Perlin noise is that it produces a smooth, natural-looking texture with its energy concentrated around a specific frequency. To create more complex patterns with details at multiple scales, several layers of Perlin noise with different frequencies and amplitudes are typically combined. Our initial attempt utilizes a common method for generating additive fractal noise (often termed pink noise). This process involves summing multiple layers of Perlin noise, where at each step, the sampling frequency doubles and the amplitude halves. Sampling at a given point (x, y) at scale s can also be thought of as sampling a universal noise at $(x \cdot s, y \cdot s)$. Multiplying the point coordinates by the scale larger than 1 effectively compresses the noise pattern, increasing the frequency of the noise. To avoid potential artifacts from sampling identical coordinates across different scales, we apply both scaling and translational offsets to the input coordinates for each noise layer: for instance, for a point (x, y) and a given scale, the noise is sampled at $((x + 3) \cdot \text{scale}, (y + 3) \cdot \text{scale})$. If the scale values are chosen as powers of two, adding $3 \cdot \text{scale}$ to a point, ensures that the noise samples of different scales do not overlap. Points are then sampled in a domain, and their acceptance probability is modulated by the resulting smooth, cloud-like noise value at their location. However, this additive approach produces noise that is too smooth, and the resulting "obstacles" are not sufficiently "hard" or prohibitive. Graphs generated by accepting points based on this smooth additive fractal noise exhibit separator sizes scaling as $\mathcal{O}(n^{1/2})$. Alternative procedural noise types, such as Simplex noise or Brownian noise, likewise fail to produce the desired combination of relatively sharp boundaries and distinct features across multiple scales. To create more defined, "harder" obstacle boundaries across multiple scales, we develop an alternative approach using multiplicative Perlin noise. For each point p, Perlin noise is first sampled at L different scales (often referred to as octaves), where the frequency typically doubles at each successive scale. The output from the Perlin noise function for each of these L scales, usually in the range $[-1, 1]$, is then normalized to $[0, 1]$. Finally, the overall noise value for the point is determined by the product of these L normalized values. As the number of noise scales increases, the resulting product tends towards zero. To transform this continuous output into a binary map of allowed versus disallowed regions, we apply a noise threshold (e.g., 0.5^L). A point is considered in an "allowed" region if its final product value is greater than the noise threshold, otherwise, it is "disallowed". This method yields a binary noise field with sharp boundaries and features at various frequencies, as Figure 5.19 illustrates. One could also consider normalizing the product of the noise values by taking the L-th root of the product and using this value as a probability if a point is in an allowed region or not. This however yields quadratic scaling behavior for the separator size, as the resulting noise field is too smooth and does not produce sufficiently distinct obstacle regions.

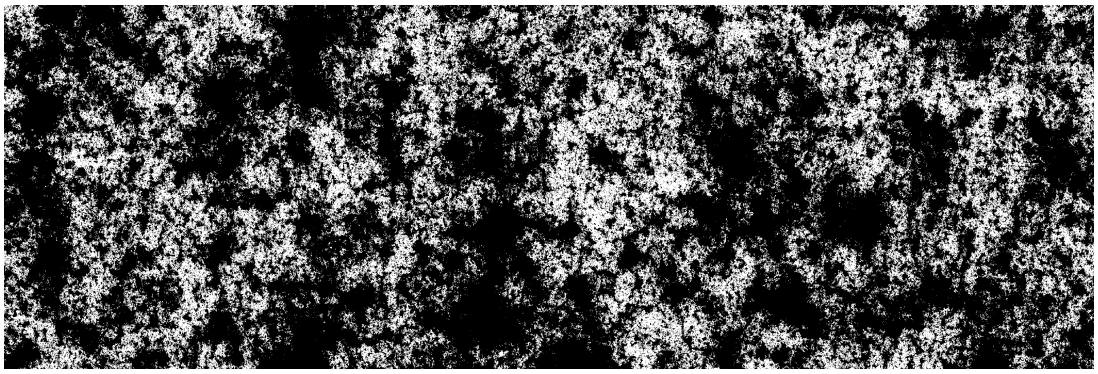


Figure 5.19: Visualization of the multiplicative binary Perlin noise used to define obstacle regions across multiple scales.

The graph generation algorithm then utilizes this multiplicative binary Perlin noise to place vertices, as detailed in Algorithm 5.3. Candidate points are randomly sampled within a predefined domain (e.g., a unit circle). The noise value, as described above, is computed at each candidate point's location. If this value indicates an "allowed" region, the point is accepted and added to the graph's vertex set, otherwise, it is rejected, and a new candidate is sampled. This process continues until the desired number of n vertices is collected.

Once the n points are generated, a global Delaunay triangulation is performed on this point set. Subsequently, the same edge pruning step detailed in the hierarchical Delaunay Generation section (see Algorithm 5.2) is applied to achieve a target average degree and refine the graph structure. An example of a graph generated using this method and its observed separator scaling are shown in Figure 5.20.

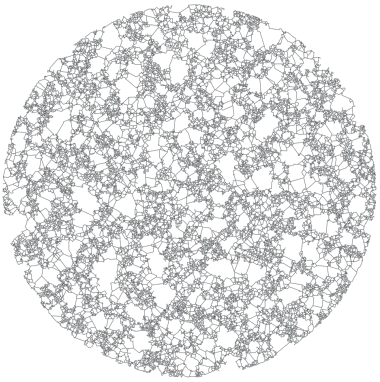
Achieving the desired separator scaling for a larger number of vertices appears to necessitate an increased number of noise scales. No clear upper bound on the number of applicable scales is apparent. Increasing the number of scales introduces a slight computational overhead. However, this does not substantially affect the overall performance of the graph generation process. Our implementation successfully generates graphs with vertex counts $n \in \{10^4, 10^5, 10^6, 10^7\}$ using $L = 11$ scales. The generation method demonstrates scalability, producing graphs with up to 300 million nodes that still exhibit the desired separator scaling. For larger graphs, we primarily rely on using the Relative Neighborhood Graph (RNG) creation instead of a Delaunay triangulation with the following edge pruning because the RNG is more efficient to compute for larger point sets. Larger experiments are constrained by limitations in underlying libraries.

This generation process does not prevent Delaunay edges from spanning the "obstacle" regions similar to how mountain passes or bridges span natural barriers in real-world road networks. However, it ensures that no vertices, and thus no potential navigation hubs or major junctions, are located within these simulated forbidden zones. Furthermore, the subsequent edge pruning step tends to remove many of the long, redundant edges that might span these empty regions, preserving only those deemed more critical for connectivity, akin to bridges over rivers or passes through mountainous terrain.

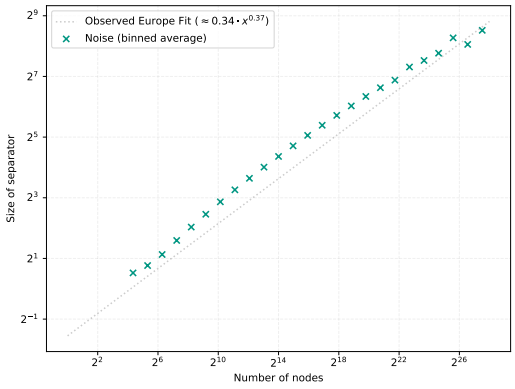
Algorithm 5.3: Graph generator using Multiplicative Binary Perlin Noise

Input: Target number of points n,
L noise scales s_i
Output: Geometric graph G

$P \longleftarrow \emptyset$
while $|P| < n$ **do**
  $p \longleftarrow$ randomPointInUnitCircle()
  noise $\longleftarrow \prod_{i=1}^{L}$ perlinNoise$(p \cdot s_i)$
  if noise $> 0.5^L$ **then**
    $P \longleftarrow P \cup \{p\}$
$G \longleftarrow$ delaunay(P)
pruneEdges(G)
return G



(a) Graph generated using multi-scale Perlin noise-based point sampling ($n = 10^5$).

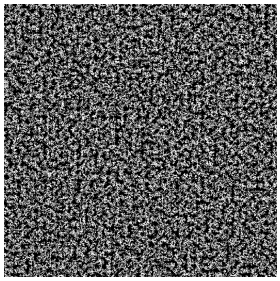


(b) Separator scaling for the multi-scale Perlin noise graph.

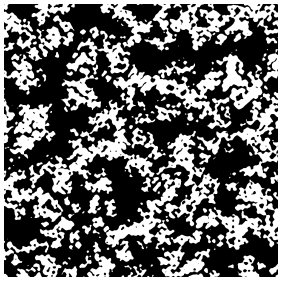
Figure 5.20: Synthetic graph generated using multiplicative binary Perlin noise and its separator scaling. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

The primary success of the multi-scale Perlin noise model lies in its ability to naturally generate graphs with the desired separator properties. Requiring only the selection of a suitable range of scales, rather than extensive fine-tuning, this method produces graphs whose separator sizes scale approximately as $\mathcal{O}(n^{0.37})$. This result is crucial, as the exponent aligns almost exactly with the scaling empirically observed in real-world road networks.

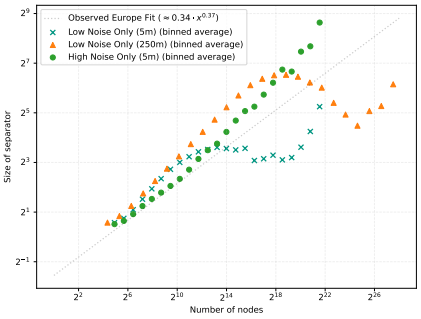
Ablation Studies on Noise Scales We also investigate whether the full spectrum of noise scales is necessary to achieve this result. Experiments are conducted using only high-frequency scales and only low-frequency scales. The resulting noise patterns and separator scaling comparisons are illustrated in Figure 5.21. When using only high-frequency scales, the separators for smaller subgraphs (identified through nested dissection) align well with the desired trend. However, as the graph size increases, the separator sizes revert to an $\mathcal{O}(n^{1/2})$ trend, because large-scale obstacle features (which would be defined by low frequencies) are absent. Conversely, the model using only low-frequency scales exhibits a more complex, multi-stage scaling behavior. For smaller subgraphs, which fit entirely within the open regions defined by the noise, separators tend to scale as $\mathcal{O}(n^{1/2})$, similar to standard Delaunay triangulations in unobstructed space. As subgraphs become large enough to span these regions, the low-frequency obstacles themselves begin to define the partitions, causing a "drop-off" from this initial trend to relatively smaller separator sizes. However, for the final top-level separators, we observe a steep increase in size. To investigate this further, we generated a much larger graph with ≈ 250 million nodes. This larger experiment confirms the same overall pattern of a drop-off followed by an increase for the largest separators, but the transition points are shifted to larger subgraph sizes. This indicates that the subgraph size at which the low-frequency obstacles become effective separators is relative to the total number of nodes in the graph being partitioned.



(a) Noise from high-frequency scales only.



(b) Noise from low-frequency scales only.



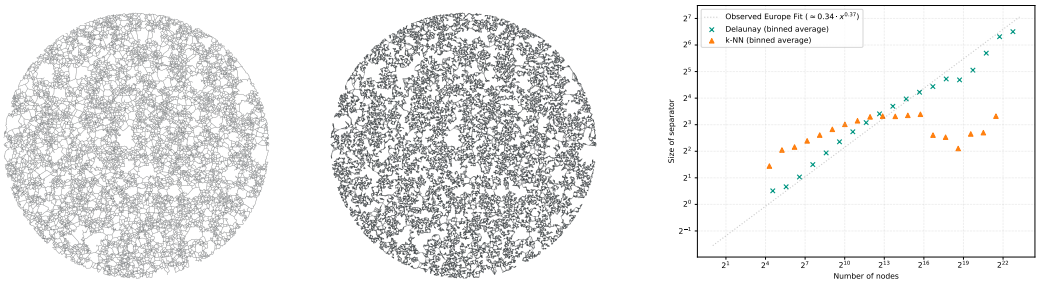
(c) Separator scaling: high-freq. vs. low-freq. only.

Figure 5.21: Impact of using only high-frequency versus only low-frequency Perlin noise scales on (a, b) noise patterns and (c) separator scaling. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

These studies therefore show that using a combination of multiple noise scales, representing obstacles of various sizes, is key to consistently achieving small separator sizes across a wide range of graph dimensions with this multi-scale Perlin noise method. This reliance on features defined across different scales to create a complex, layered obstacle landscape is conceptually similar to the explicit hierarchical graph generation methods explored earlier

(e.g., Section 5.5.3), in both approaches, such multi-scale or hierarchical structuring is vital for achieving the desired separator properties, as is demonstrated in our noise model experiments where the favorable separator scaling breaks down if an insufficient range of scales is employed.

Alternative Graph Construction: k-Nearest Neighbors While the primary approach described above utilizes a Delaunay triangulation followed by pruning on the noise-sampled points, we also explored an alternative graph construction method using k-Nearest Neighbors (k-NN) on the same underlying point sets. For these non-uniformly distributed points, a consequence of the noise-based acceptance criteria, a $k = 3$, which often suffices to ensure reasonable connectivity for uniformly sampled points, proved insufficient. We found that a value of $k = 5$ was generally necessary to achieve a largely connected graph. Interestingly, k-NN graphs constructed from these noise-sampled points exhibited separator sizes that were substantially smaller than our target $\mathcal{O}(n^{0.37})$. Figure 5.22 illustrates this comparison. Visually comparing a k-NN graph (Figure 5.22b) with a graph derived from the Delaunay-based approach on the same noise-sampled point set (Figure 5.22a) reveals that the k-NN graph possesses significantly fewer long edges. This characteristic arises because the k-NN construction inherently limits connections to only the k closest neighbors. Consequently, edges are less likely to span large low-noise regions (our simulated obstacles), particularly if a point's k nearest neighbors all reside on the same side of such a zone. Marginally increasing k does not fundamentally alter this effect, as it still restricts connections to a local neighborhood. The limited long-range connectivity of the k-NN rule appears to cascade through recursive partitioning. This leads to disproportionately small separators for larger graphs, while smaller, more internally-connected subgraphs can exhibit comparatively larger separators. For substantially larger values of k, such as k=50, the separators exhibit quadratic scaling.



(a) Delaunay subgraph on noise-sampled points.

(b) k-NN graph ($k = 5$) on the same noise-sampled point set.

(c) Separator scaling: Delaunay-based vs. k-NN based.

Figure 5.22: Comparison of graph structures and separator scaling for Delaunay-derived graphs versus k-NN graphs on identical noise-sampled point sets. Separators were computed using InertialFlowCutter (tests with FlowCutter and KaHIP yielded similar asymptotic scaling).

This comparison highlights a distinction between the two graph construction methods. A hierarchical point distribution, as generated by our multi-scale noise, appears to be a necessary but not sufficient condition for achieving the target separator scaling. The resulting separator scaling also appears to depend on the specific connectivity pattern induced by the graph construction algorithm. In contrast to the k-NN graph, the Delaunay triangulation permits the

creation of long-range edges that span large, empty regions (our simulated obstacles). These "bridge" edges appear to contribute to a more globally integrated graph structure, which may explain why this model avoids the higher degree of fragmentation and the smaller separator sizes observed with the more locally-restricted k-NN approach.

Comparison with Real-World Hop Distribution Given the success of the multi-scale Perlin noise model in replicating separator scaling, we further investigate its potential as a general-purpose synthetic road network generator by comparing its structural properties to a real-world benchmark. For this comparison, we focus on the hop distance distribution rather than the weighted distance distribution. A direct comparison of geographic path lengths is not as insightful here, as this metric is heavily influenced by the overall shape and scale of the embedding domain. The hop distribution, in contrast, better reflects the intrinsic topological structure and hierarchical connectivity of a graph, making it a more suitable metric for evaluating our model.

To proceed with this comparison, we analyze four key graphs. First, the real road network of Germany is pre-processed by contracting its degree-2 vertices. We choose Germany as a representative region as it is sufficiently large while being more geographically compact, thus mitigating the heavy-tailed hop distribution seen for the entire European continent. Second, to establish a baseline for dense geometric graphs, we generate a full Delaunay triangulation with the same number of nodes as the contracted Germany graph. Third, we generate a graph using our primary noise-based method: starting with a number of points equal to the original uncontracted Germany network, we perform a Delaunay triangulation, apply our pruning step from Algorithm 5.2, and finally contract any resulting degree-2 nodes. As a fourth comparator, we generate a graph using the same process but replace our pruning step with the one described by Bauer et al. [BKMW10], which is known to preserve significantly more long-range edges.

The resulting hop distributions for these graphs are illustrated in Figure 5.23. We observe that the hop distribution of the full Delaunay triangulation closely resembles that of the real Germany graph. In contrast, the graph generated with our proposed pruning method exhibits a significantly less efficient hop distribution, requiring many more hops on average to connect node pairs. This suggests that our pruning strategy, while effective for achieving a target average degree and visual appeal, removes too many critical long-range edges. The model using the pruning step from Bauer et al. performs better by preserving more of these long-range connections, resulting in a hop distribution much closer to that of the real Germany graph. However, it is still not as topologically efficient as either the full Delaunay triangulation or the actual Germany network.

While this analysis indicates that our pruning strategy does not capture the topological efficiency required of a general-purpose road network generator, this limitation does not affect the noise model's primary success, as the overall asymptotic separator scaling is independent of the pruning method.

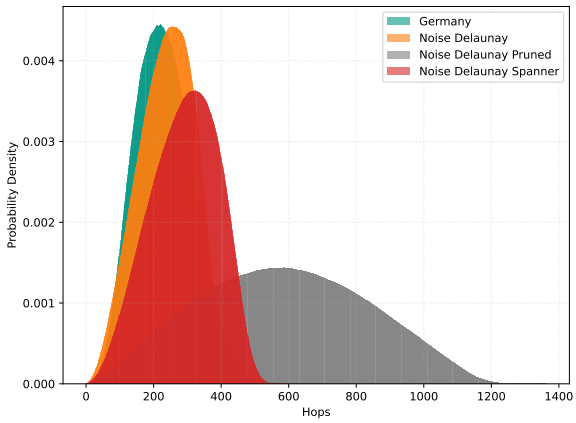


Figure 5.23: Comparison of hop distance distributions for the contracted Germany network and three synthetic variants: full Delaunay, our pruned noise-based model, and a noise-based model using the pruning strategy from Bauer et al.

6 Conclusion

This chapter summarizes the key findings from the various models tested. It then addresses the limitations of our experimental approach and outlines several promising directions for future research.

6.1 Summary of Findings

We demonstrate that the separator size of road networks s scales with graph size n according to $\mathcal{O}(n^{0.37})$. This exponent is slightly larger than the previously suggested $\mathcal{O}(n^{1/3})$ [DSW16] but remains substantially better than the $\mathcal{O}(n^{1/2})$ bound for planar graphs.

Our investigation systematically demonstrates the insufficiency of simple, isolated graph properties to explain this behavior. Simple models based solely on properties like degree distribution or a two-dimensional embedding consistently scale as $\mathcal{O}(n^{1/2})$ or worse.

Furthermore, planarizing real road networks has a negligible impact on their separator sizes, suggesting that the non-planarity of these graphs is not a primary factor in the observed small separators.

In contrast, models incorporating a hierarchical structure show significant promise. Our proposed hierarchical Delaunay generator defines a parameterized class of graphs. By carefully tuning its parameters (which govern expansion fractions, points per site, and radii across multiple levels), this model can also replicate the observed $\mathcal{O}(n^{0.37})$ scaling. Notably, the resulting graph structures often bear a strong visual resemblance to those generated by the physical barrier model.

The most successful synthetic model developed in this thesis, however, is one based on simulating physical barriers using multi-scale Perlin noise. This approach generates a layered landscape of obstacles at various scales, constraining where vertices can be placed. Remarkably, graphs generated using this method, which combines noise-based point sampling and a Delaunay triangulation, naturally produce separators that scale approximately as $\mathcal{O}(n^{0.37})$. An optional pruning step can be employed to reduce the average degree, but this is not strictly necessary to achieve the desired scaling. This result is achieved without extensive parameter fine-tuning, suggesting that this model captures a more fundamental generative principle. Ablation studies further underscore this finding, demonstrating that the full spectrum of noise scales, from large, regional barriers to small, local ones, is crucial for achieving this specific scaling across a wide range of graph sizes.

It is important to qualify, however, that the need for a multi-scale or hierarchical structure is not strictly necessary if one allows for extreme parameter fine-tuning. The tree-locality model, for instance, could replicate the desired scaling using a highly specific distance-decay function $f(\text{dist}) = 1/\text{dist}^{3.3}$. This case serves as an interesting exception, highlighting that specific correlation structures can be enforced without an explicit hierarchy, though this offers less insight into the natural emergence of such properties.

Revisiting the central research question, "Do small separators in road networks arise from intrinsic graph properties, or from real-world physical barriers?", our findings provide a nuanced answer. The evidence strongly suggests that small separators are not a consequence

of any single, simple graph property but are an emergent feature of a multi-scale structural organization. This multi-scale structure can be interpreted as an explicit hierarchy of road types or, perhaps more fundamentally, as the result of physical barriers existing at all geographic scales. The success of the multi-scale Perlin noise model indicates that the constrained placement of nodes by geography is a powerful generative mechanism. Road networks are built upon a landscape permeated by obstacles at every scale, from mountain ranges down to small urban features like parks. This forces the graph to be composed of densely connected regions that are themselves sparsely interconnected through well-defined corridors, a structure highly amenable to small separators. Therefore, we conclude that the small separators in road networks are a result of their adaptation to a multi-scale, obstacle-rich environment.

6.2 Limitations and Future Work

The conclusions presented in this thesis are primarily derived from experimental analysis using heuristic separator algorithms. A key methodological consideration is that finding a minimal vertex separator is an NP-hard problem for general graphs. Consequently, our analysis relies on state-of-the-art partitioning heuristics, primarily using InertialFlowCutter [GHUW19] and cross-validating findings with KaHIP [SS13] and FlowCutter [HS18]. While all three tools yield consistent asymptotic scaling results for the graphs studied, none guarantees that the separators found are of the minimum possible size. The separator sizes reported throughout this work should therefore be interpreted as empirical upper bounds on the true minimal separator sizes.

Beyond this, while our results strongly indicate the importance of hierarchy and multi-scale barriers, the inability of simpler, non-hierarchical models to reproduce the desired scaling does not constitute a formal disproof of their potential under different assumptions. A notable limitation is that our most successful models rely on a final Delaunay triangulation. The observation that k-Nearest Neighbor graphs built on the same hierarchical point sets fail to produce the correct scaling suggests that the specific connectivity pattern of Delaunay graphs, which allows for both local and some structured long-range connections, plays a significant, though not yet fully explored, role.

These limitations and findings open several avenues for future research. A primary direction is the pursuit of a rigorous theoretical analysis of the multiplicative Perlin noise model. Such a study could potentially provide a formal derivation for the empirically observed $\mathcal{O}(n^{0.37})$ scaling. Further work could also seek to improve the efficiency of the generative algorithms. For instance, the tree-locality generator is hampered by its slow, BFS-based distance calculations; developing an efficient method to sample edges according to weights derived from tree distances would make this model more viable for large-scale tests. The synthetic generators developed here can serve as a basis for creating realistic, large-scale benchmarks for evaluating not only route planning algorithms but also other algorithms that depend on graphs with small separators. An important extension of this work would be to adapt the noise generation algorithm to more closely replicate other key road network metrics beyond separator sizes, such as hop distributions, to create an even more robust synthetic road network generator.

Bibliography

[ACIM99] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. “Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication)”. In: *SIAM Journal on Computing* Volume 28 (Jan. 1999), pp. 1167–1181. ISSN: 0097-5397, 1095-7111. DOI: *10.1137/S0097539796303421*.

[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*.

[Buh+06] C. Buhl, J. Gautrais, N. Reeves, R. V. Solé, S. Valverde, P. Kuntz, and G. Theraulaz. "Topological patterns in street networks of self-organizedurban settlements". In: *The European Physical Journal B - Condensed Matter and ComplexSystems* Volume 49 (Feb. 1, 2006), pp. 513–522. ISSN: 1434-6036. DOI: *10.1140/epjb/e2006-00085-1*.

[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*.

[Cre+13] Pilu Crescenzi, Roberto Grossi, Michel Habib, Leonardo Lanzi, and Andrea Marino. "On computing the diameter of real-world undirected graphs". In: *Theoretical Computer Science* Volume 514 (Nov. 25, 2013), pp. 84–95. ISSN: 0304-3975. DOI: *10.1016/j.tcs.2012.09.018*.

[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*.

[GS69] K. Ruben Gabriel and Robert R. Sokal. "A New Statistical Approach to Geographic Variation Analysis". In: *Systematic Biology* Volume 18 (Sept. 1, 1969), pp. 259–278. ISSN: 1063-5157. DOI: *10.2307/2412323*.

[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*.

[HS18] Michael Hamann and Ben Strasser. “Graph Bisection with Pareto Optimization”. In: *ACM Journal of Experimental Algorithmics* Volume 23 (Nov. 15, 2018), pp. 1–34. ISSN: 1084-6654, 1084-6654. DOI: *10.1145/3173045*.

[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*.

[Per85] Ken Perlin. “An image synthesizer”. In: *ACM SIGGRAPH Computer Graphics* Volume 19 (July 1985), pp. 287–296. ISSN: 0097-8930. DOI: *10.1145/325165.325247*.

[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*.

[Tou80] Godfried T. Toussaint. “The relative neighbourhood graph of a finite planar set”. In: *Pattern Recognition* Volume 12 (Jan. 1, 1980), pp. 261–268. ISSN: 0031-3203. DOI: *10.1016/0031-3203(80)90066-7*.