Detecting Highways on Large Networks

Master Thesis

Submitted by

Timon Cedric Blauensteiner

at

University of Konstanz, Germany

Faculty of Mathematics and Natural Sciences

Department of Computer and Information Science

1st examiner: Prof. Dr. Michael Grossniklaus

2nd examiner: Prof. Dr. Sabine Storandt

Konstanz, 2019

Abstract

Highways are high-speed multi-lane roads which facilitate long-distance travelling in road networks. The properties of those highways yield some practical and theoretical implications which have been the motivation and foundation of many efficient route planning algorithms. In our work, we discuss the underlying structural concepts of highways and propose a formal definition for highways in networks. We provide a fast way of computing such highways based on a modification of Brandes' algorithm. Furthermore, we introduce an approximation approach which makes the computation even faster by two orders of magnitude. In our experiments, we show that our method Highway-Paths Ratio can identify up to 99% of all highway segments in real road networks. Additionally, we define a new contraction order for Contraction Hierarchies, a fast route planning algorithm based on vertex contraction. Our contraction order is solely based on our definition of a highway network and performs similarly good as the original approach Edge Difference, hence demonstrating that our highway definition finds an inherent hierarchical structure in the road network. The argument of finding an inherent structure is further supported when we compare the original road networks with random networks of the same degree distribution and the same weights. We discover a unique distribution of our highway edge scoring compared to the random networks, which in contrast to the original network follow a nearly log-normal distribution. This versatile toolkit of contraction order and value distributions in the original and random networks allows us to not only detect and verify highway-like structures in road networks but also in other types of networks. With our analysis on six networks from various domains, we lay the foundation for the highway detection and analysis in general networks and we hope to motivate research on the detection and analysis of highway-like structures on various types of networks.

Contents

Acknowledgements

There are lot of people to thank for each and every step towards my master thesis, both helping me acquire the skills and knowledge needed for accomplishing such, as well as being supportive. I want to thank you all. Additionally, I want to explicitly mention a few people who directly gave me feedback to my master thesis. First and foremost, I want to really thank Theodoros Chondrogiannis. He guided me all the way from conceptional discussions in the beginning to concrete comments on the thesis content and structure, always motivating me to take the next step. Furthermore, I want to thank Sabine Storandt. She gave me valuable feedback to my conceptional ideas and further suggestions to extend my work. Finally, I want to thank Matthias Albrecht for giving me feedback to the overall structure and style of the thesis.

1 Introduction

Shortest paths and route planning are one of the main areas of research on networks. That is because route planning algorithms play an important role in modern society beyond the area of research. We commute over large distances between home and workplaces, visit family and friends who live far away and go on road trips or drive to holiday destinations far from our home. Route planning algorithms guide us to those destinations while saving time and fuel. One common optimisation criterion is finding the path with the lowest travel time. Especially for long-distance journeys, they are often covered by a large portion of highways. That observation comes naturally, as highways allow fast, multi-lane travelling over long distances. While highways do not directly lead exactly to most destinations, they usually make up for their detour with higher capacity, higher speed limits and fewer turns, resulting in lower travel times.

As they usually have higher speed limits and often justify a detour, highways have been a motivating key property of route planning algorithms on road networks. These algorithms can leverage such a network structure [2, 6, 30, 31] to make route planning especially fast in road networks [5].

With our work, we want to introduce an intuitive definition of highways from a perspective of network analysis. We investigate and describe what highways are and justify our choice with an empirical comparison with the official highways of different regions. Furthermore, we demonstrate that our highway definition has a structural meaning, e.g. it can be used for efficient route planning. Afterwards, we want to transfer our definition to other types of networks. With this, our final goal is to explore networks from other domains and investigate whether similar highway structures exist in those networks and if we can draw conclusions from a network's highway structures.

We summarise the goals we want to achieve in this exact order throughout the thesis:

1. We want to use our learnings from existing work to find a formal definition of a *highway* from the view of network analysis.
2. We want to verify our definition of a highway network by applying it to real road networks.
3. We want to give an outlook on how we can use our verified definition of highways with other types of networks of various domains.

To accomplish these goals, we go through a number of concrete steps which lead to the following contributions:

- We do a literature review on the use of the term highway in network analysis and route planning and extract the main criteria for highways and settle on a formal definition to describe highways. We propose 4 measures to identify highway edges:

  - The existing measure Edge Betweenness [4, 8] as a baseline edge priority
  - Highway-Paths Ratio, which measures which ratio of shortest paths are within a highway network
  - Distance-Based Highway-Paths Ratio, a variant which considers the length of road segments
  - Highway Power-Distance, attempts to amplify long-distance fastest paths for selecting highways close to the border of the network.

- We propose an approximation algorithm which simplifies the network and speeds up the computation of a highway network.

- We propose an independent contraction order for Contraction Hierarchies [20] purely based on the highway structure measured with Highway-Paths Ratio.

- We design our experiments with a preprocessing pipeline which makes it easy for us to use real-world road networks from OpenStreetMap [12] for our highway network analysis.

  - We compare the identified highway edges with the officially classified highways. With our definitions, we label over 55% of highway edges correctly, with up to 99% on an extended road network of Thuringia.
  - We demonstrate that our contraction order can contract the graph with a similar number of shortcuts as the original approach Edge Difference. With that, we can verify that our definition not only finds official highways but also finds intrinsic structures of the network which can be leveraged for route planning algorithms.
  - We compare the distribution of Highway-Paths Ratio between the different road networks and their random network counterparts with the same degree distribution and weights. We find a significant difference in distributions between the original network and the random networks which all follow a nearly log-normal distribution.

- We perform preliminary experiments on networks of various domains.

  - We compare the distribution of Highway-Paths Ratio between different general networks and their random network counterparts. We discover that different types of networks behave differently when shuffled, some maintaining their distribution and some with largely different distributions.
  - We compare Contraction Hierarchies with Edge Difference and with our highway-based contraction order and discover that our own contraction order has a significantly faster preprocessing than Edge Difference for the selected networks.

2 Preliminaries

Usually, to model different optimisation criteria such as the fastest or shortest path in networks, either an undirected or a directed graph $G = (V, E)$ is used. V is called the vertex set and $v \in V$ a vertex.

For undirected graphs, the edge set is $E \subseteq \{\{v, w\} | v, w \in V \wedge v \neq w\}$ with $\{v, w\} = e \in E$ an edge connecting the vertices v and w. For directed graphs, the edge set is $E \subseteq \{(v, w) | v, w \in V \wedge v \neq w\}$ with $(v, w) = e \in E$ an edge pointing from the vertex v to the vertex w. In a directed graph, an edge $e = (v, w)$ is directed, indicating that it can only be traversed from v to w. In that case, v is called the tail of e and w is called the head of e.

The size of a network is specified by the graph's dimensions $n = |V|$ and $m = |E|$. When we discuss more than one graph, we use $V(G)$ and $E(G)$ to refer to the vertex and edge set of a graph G respectively. A graph can be a weighted graph, which means each edge has a weight $w(e) \in \mathbb{R}^{\geq 0}$ to model one particular optimisation criterion.

A sequence $p = \langle e_1 = (s, v_2), .., e_k = (v_k, t) \rangle$ is called a path from s to t iff for all $e_i = (u, v), e_{i+1} = (x, y) \in p$ holds $e_i, e_{i+1} \in E$ and $v = x$. The cardinality of a path is denoted as $|p|$ with $|p| := k$ counting the number of edges on the path. The length of a path in a weighted graph is denoted as $l(p)$ with $l(p) := \sum_{e \in p} w(e)$. In an unweighted graph, $l(p) := |p|$ is usually assumed.

$P(s, t)$ is the set of all paths from s to t and $P := \bigcup_{s,t \in V} P(s, t)$ is the set of all paths in the graph. We refer to all optimisation criteria from now on as distance and call a path $p \in P(s, t)$ where for all $p' \in P(s, t)$ holds $l(p) \leq l(p')$ shortest path from s to t. We further refer to the set of all shortest paths from s to t as $SP(s, t) := \text{argmin}_{p \in P(s,t)} l(p)$ and call the length of those shortest paths distance between s and t, with $d(s, t) := \min_{p \in P(s,t)} l(p)$.

The classical approach of finding shortest paths in graphs is Dijkstra's algorithm [14]. It is a graph exploration algorithm which queues vertices ordered by their tentative shortest distance during the current progress of exploration from a source vertex s. In each step, the vertex with the lowest tentative distance is removed from the queue, and the outgoing edges are checked for yielding a shorter path to their heads. Dijkstra's algorithm has the disadvantage that it explores all shortest paths which have a shorter or equal distance than the distance from s to our desired destination t. It yields a runtime of $\mathcal{O}(n \log n + m)$ using Fibonacci heaps [16]. Besides of the comparison with more efficient shortest path algorithms, we will revisit Dijkstra's algorithm [14] when we introduce Brandes' algorithm [9] to compute Edge Betweenness in Section 4.2.1.

We extend the set operations to tuples and and paths, which means for a path $p = \langle e_1, .., e_k \rangle$ holds $\forall 1 \leq i \leq k : e_i \in p$ and $(\neg \exists 1 \leq i \leq k : e_i = e) \implies e \notin p$. Based on that, we can apply all the other set operations. E.g. for $E' \subseteq E$ we define $E' \cap p = \{e | e \in p \wedge e \in E'\}$, the set of edges on p contained in E'.

A vertex v in a directed graph has an in-degree $deg_{in}(v)$ and an out-degree $deg_{out}(v)$, which refer to the number of edges point to or from the vertex v respectively. Formally, $deg_{in}(v) = |\{e|(u, w) = e \in E \wedge v = w\}|$ and $deg_{out}(v) = |\{e|(u, w) = e \in E \wedge v = u\}|$, where we call $deg(v) = |\{e|(u, w) = e \in E \wedge v \in e\}|$ the edge degree of v. In an undirected graph, the degree of v is $deg(v) = deg_{in}(v) = deg_{out}(v)$.

In real-world road networks, a vertex $v \in V$ usually represents an intersection and an edge $(v, w) = e \in E$ a road segment connecting two intersections v and w. Furthermore, road networks are generally mixed networks with directed edges (v, w) and undirected edges $\{v, w\}$. This way, one-way roads can be modelled with a directed edge and two-way roads can be modelled with an undirected edge. In order to run graph algorithms on those networks, we will treat an undirected edge $\{v, w\}$ as two directed edges $(v, w), (w, v)$, while keeping the same edge id on both to treat them together in any network analysis. When it comes to measuring the network dimensions, we also count the undirected edges as two edges. That is because for graph exploration algorithms, both directions have to be considered. Therefore, for $E = E_U \cup E_D$ where E_U are the undirected and E_D are the directed edges, we say $m = |E_D| + 2|E_U|$.

3 Related Work

The properties of highways have been the motivation of many route planning algorithms [2, 6, 30, 31]. While precomputing a simple lookup table of source-target pairs is the most efficient technique in terms of runtime per query, the additional space consumption is unfeasible in large networks. Leveraging the unique structure of road networks has yielded route planning algorithms with low additional space consumption and runtimes better than a traditional Dijkstra in multiple orders of magnitude [5, p. 38-46]. The empirical assessment that those route planning algorithms only work well on some types networks has further been backed by *Highway Dimension* [1], which proves upper bounds for various route planning algorithms with a pronounced highway structure.

In this section, we take a look at some route planning algorithms focusing on and using the term *highways*. With this, we can extract properties of highways and how they are used to make those route planning algorithms efficient. We will see that road networks have a hierarchical structure that can be utilised to quickly answer shortest path or distance queries [2, 30, 31] in road networks. While the definitions surrounding highways are different in each approach, they all agree on highways being included in many shortest paths, especially long ones. With all that, it is safe to say that highways play an important role in optimising route planning and describing the structure of road networks. Only some of the papers we present motivated their definitions empirically. Therefore, an intuitive single definition should be found describing what a highway is. This definition should have an empirical foundation closely matching the real highways of road networks.

3.1 Highways and Route Planning Algorithms

Highway Hierarchies [30], Dynamic Highway-Node Routing [31] and Contraction Hierarchies [20] will be discussed together because they build upon each other. Highway Hierarchies build the foundation by defining a hierarchical structure. This is used by Dynamic Highway-Node Routing to allow a route planning algorithm with fast updates of edge weights. Contraction Hierarchies combines ideas from both approaches with a similar traversal method as Highway Hierarchies.

At the end of this sub-section, we will look at a work independent from those three, Pruned Highway Labelling [2]. Pruned Highway Labelling is based on Hub Labels [19] and uses highways for the choice of its hubs.

3.1.1 Highway Hierarchies

Highway Hierarchies [30] is the first approach which guarantees exact shortest paths with low preprocessing time in the order of milliseconds. In order to achieve that, it requires preprocessing in the order of hours on the road networks on large road networks. The algorithm uses a hierarchical structure of sub-graphs, or highway networks, of the original graph G. To query a shortest path, an explorative traversal

at the original graph starts from both source and target with a Bi-directional Dijkstra. The traversal can either stay in the same hierarchical level or take an upwards edge into the next level. The exploration of the same hierarchical level is limited to the H closest vertices. That splits the exploration into two alternating modes: Performing a local search with at most H vertices, and going to a higher level of highway network to reach vertices which are farther away. Because there is no way to move back down the hierarchy, the backwards search of Bi-directional Dijkstra is needed to explore the path from the target to the highest point of the shortest path in the highway hierarchy.

The highway hierarchy is built on multiple levels of *highway networks*, which consists of *highway edges*. A highway edge (u, v) is defined as an edge that lies on a shortest path $\langle s, .., u, v, .., t\rangle$ and $v \notin N_H(s)$ and $u \notin N_H(t)$, where $N_H(w)$ denotes the set of vertices which have at most the distance of the H closest nodes to w. In other words, a highway edge is an edge that lies on a sufficiently long shortest path outside the local range of s and t.

A highway network G_1 of G is the induced sub-graph of highway edges of G. But instead of taking all highway edges, only those belonging to a *canonical shortest path selection* are part of the highway network. A *canonical shortest path selection* $\mathcal{SP}$ is a subset of all shortest paths such that there is only one path from s to t in $\mathcal{SP}$ (we refer to that path as $\mathcal{SP}(s, t)$ now) and that for $\langle s, .., u, .., v, .., t\rangle = \mathcal{SP}(s, t)$ holds that $\mathcal{SP}(u, v)$ is a sub-path of $\mathcal{SP}(s, t)$.

After building a highway network G_1 of all canonical highway edges, the highway network is contracted to the *contracted highway network* G'_1. As any graph, the highway network G_1 consists of a more-densely connected core graph and its appendices, called *components*. The contracted highway network G'_1 is chosen to be the core of G_1. For that, the 2-core of G_1 is taken, a reduction of the graph which removes all vertices with vertex degree lower than 2 until there are only vertices of degree 2 or higher. Afterwards, vertices of degree 2 are removed, and their incident edges merged to a single edge. That reduces long paths of vertices with degree 2 to single edges. The resulting graph is a 3-core, which yields the contracted highway network G'_1. All vertices and edges in G_1 not part of the contracted highway network G'_1 are the components of G_1. Iteratively, G'_i is used to find the canonical edges for the highway network G_{i+1}, which is then contracted to G'_{i+1}. All those graphs G_i form the highway hierarchy. To find highway edges on a canonical shortest path selection, the paper proposes a modification of Dijkstra's algorithm.

To query the shortest path from a source s to a target t, a bi-directional Dijkstra is started from the lowest level $G_0 = G$ in the highway hierarchy for both source and target. The edges of within a level are called *horizontal edges*. Furthermore, you can go from any vertex which also exists in the next higher level through a directed *vertical edge* with edge weight 0. For each level, the entrance vertex into the level v^* is kept track of. If a vertical edge (v_i, v_{i+1}) is settled, v_{i+1} becomes an entrance point of the level of v_{i+1}. Furthermore, if a horizontal edge (v, w) is settled and v lies on a component and w on the core, w is also considered an entrance vertex into the level of w. s and t are considered entrance vertices of level 0. Horizontal edges can only

be traversed if the end point lies within $N_H(v^*)$. Additionally, components cannot be entered with horizontal edges. In other words, the H closest vertices around an entrance vertex v^* can be explored within a level's core, allowing a local search. For everything beyond the local search of H vertices, a vertical edge to the next higher level needs to be traversed to reach more distant parts of the network.

The paper further argues that the highway hierarchy can be reduced to a single super-graph where they store the maximum level of each edge to decide whether to traverse it.

3.1.2 Dynamic Highway-Node Routing

Dynamic Highway-Node Routing [31] improves and modifies the idea of Highway Hierarchies. Its main goal is to quickly adapt to network changes, such as traffic jams or road maintenance without a large amount of preprocessing. Instead of fully modelled reduced graphs which belong to certain levels, we assign vertices to different levels arbitrarily. While having an effect on preprocessing and query time, the algorithm provides optimal shortest paths independent of the choice of levels for the vertices. Due to that and the assumption that updates on edge weights yield graphs with similar shortest paths with few exceptions, the authors expect that a good selection of vertex levels for an original graph also work well with an updated graph. In other words, edge weight updates which have a minor influence on the entire network structure would not require much preprocessing to yield a fast query answering, while previous approaches would require you to redo the entire preprocessing for an updated weight function.

The hierarchy is enforced by highway vertex sets where V_0 is the lowest level and $V = V_0 \supseteq .. \supseteq V_L$. A vertex v belongs to level i if for all $i < j \leq L$ holds $v \notin V_j$. For each level $l > 0$, an edge set is induced which forms $G_l = (V_l, E_l)$ with $E_l := \{(s,t) \in V_l \times V_l \mid \exists p = SP(s,t) \text{ in } G_{l-1} : \forall v \in p \setminus \{s,t\} : v \notin V_l\}$. That means if there is no vertex in the current level l on the shortest path between s and t, an edge is created. With that, shortest paths and connectivity between all vertices at a certain level are maintained, even if intermediate vertices on the path between them do not exist on that level.

The authors claim that vertices which lie on many shortest paths should be chosen to be on a higher level to achieve low query times. For their first implementation, they use Highway Hierarchies [30] to select those vertex sets. The authors take the vertices of a Highway Network of level l to be the vertices of the highway vertex set of level l. Afterwards, they construct an overlay graph for G which consists of all vertices and edges from all levels. The overlay graph is constructed bottom up from level 1 to L where L denotes the highest level. To figure out if a vertex of the current level is still on the shortest paths between two vertices of the level, they propose different approaches. They have a conservative approach that gives exact results, but can be relatively slow in some scenarios. So instead, they use an approach that finds a superset of *covering* vertices for shortest paths, which can introduce more edges on a level than necessary while reducing the preprocessing costs. That leads to a trade-off between preprocessing time and query time.

To query a shortest path, a modified Bi-directional Dijkstra is performed on the overlay graph of all levels. For any currently settled vertex v with a level l, only edges in $\bigcup_{i=l..L} E_i$ are relaxed. For some approaches which only compute supersets of covering vertices, further steps have to be taken to assure optimality.

While the level of vertices can be kept after updating edge weights, some shortest paths might change. If those new shortest paths have intermediate vertices on the same level, it can lead to different edges for a graph G_i, which leads to a different overlaying graph. For few weight changes, the authors propose a method to redo the construction of the overlaying graph only for all vertices involved in edge weight updates. For replacing the entire edge costs, the overlaying graph has to be recomputed, but the choice of the highway vertex sets can be kept. That reduces preprocessing costs for different edge weights as long as the hierarchical structure of the network remains similar with the new edge weights.

3.1.3 Contraction Hierarchies

Finally, that brings us to Contraction Hierarchies [20]. With milliseconds of querying time and preprocessing in the order of minutes for large road networks such as Western Europe or the US, Contraction Hierarchies hit a sweet spot of balancing preprocessing time and querying time making them highly versatile. While the approach loses its focus specifically on highways, the theoretical runtime and space complexity is proven to be good in networks with a strong highway structure [1]. The authors of Contraction Hierarchies use many approaches of the previous papers to build a hierarchical structure on the network and perform traversals only up from the source and back down to the target with a Bi-directional Dijkstra.

The hierarchical structure has $|V| = n$ levels, each vertex forming its own level. The level is determined by its rank $r(v)$. All vertices of the graph are contracted iteratively to construct an overlay graph which is used for shortest path queries. Given by the rank $r(v)$, starting with $r(v_0) = 0$ and contracting v before w with $r(v) < r(w)$, each vertex is removed from the graph of the previous contraction step. To maintain the shortest paths between all remaining vertices in the network, shortcut edges are added between all neighbours of v for all shortest paths which go through v. The edge weights of a shortcut $e_s = (u, w)$ is then the distance of those neighbours u and w, where $w(e_s) = w(u, v) + w(v, w)$. Additionally, for each shortcut, the edges it replaces is stored to be able to reconstruct the original paths when answering shortest path queries. The original graph with all additional shortcuts yields the overlay graph G'.

The overlay graph G' can be seen as two graphs, the up-graph $G^{\uparrow}$ which only contains edges (u, v) where $r(u) < r(v)$, and the down-graph $G^{\downarrow}$ which only contains edges (u, v) where $r(u) > r(v)$. Those graphs are not modelled separately but instead are a result of the Bi-directional Dijkstra used to answer shortest path queries. A shortest path query for a source s and target t is answered with a forward search from s and a backward search from t. The forward search is only allowed to relax edges of the up-graph, while the backward search is only allowed to relax edges of the down-graph. Geisberger et al. prove that any shortest path completely consists of

edges in the up-graph followed by edges in the down-graph. That means that forward and backward search meet at the vertex of the shortest path with the highest rank.

Space and time complexity largely depend on the number of shortcuts in the graph, which again depends on the order in which vertices are contracted. The original approach *Edge Difference* measures the resulting number of edges after the contraction of a vertex, $ED(v) = \#\text{shortcuts} - \deg(v)$. The vertices are contracted greedily from the lowest Edge Difference to the largest Edge Difference while updating the Edge Difference after each contraction step.

To further support the empirical runtime of Contraction Hierarchies, there are several works on proving the asymptotical runtime and space complexity of Contraction Hierarchies, using Bounded Growth [18], Highway Dimension [1] and Tree Width [7].

3.1.4 Pruned Highway Labelling

Pruned Highway Labelling [2] is a modification of Hub Labelling. Hub Labels [19] use vertex labels $L(v) \subseteq V$ for each vertex v. The vertex labels have to fulfil the *cover property* to be able to fulfil shortest path queries. That means for any vertex pair $s, t \in V$, there must be a vertex $v \in L(s) \cap L(t)$ such that v lies on the shortest path from s to t. After vertex labels fulfilling the cover property are selected, additional information needs to be computed to be able to answer shortest path queries for a vertex pair s, t. For that, the distance from and to a vertex v to its labels $L(v)$ is computed. The labels are sorted by some vertex ordering, e.g. by their vertex id. That allows the algorithm to compute $L(s) \cap L(t)$ in linear time, simply by scanning both ordered label sets. The distance can be retrieved by scanning the labels of both the source and target and finding the vertex $v \in L(s) \cap L(t)$ such that the sum of the distance from $d(s, v) + d(v, t)$ is minimum. The additional space consumption of Hub Labels is limited to the labels. Choosing $L(v) = V$ for all vertices $v \in V$ would fulfil the cover property but the space consumption would be $\mathcal{O}(n^2)$, defeating the advantage of Hub Labels over a simple distance lookup table for all vertex pairs. Instead, a sub-linear number of labels per vertex is needed for a low space consumption.

Pruned Highway Labelling allows a much lower preprocessing time than previous Hub Label approaches, allows less space consumption by sacrificing some query time, being slower than classic Hub Label approaches. Instead of having vertex labels $L(v) \subseteq V$ for each vertex v and storing a pair $(u, d(v, u))$ for undirected graphs, Pruned Highway Labelling has shortest path labels to a shortest path p_i storing a triplet $(i, d(p_{i,1}, p_{i,j}), d(v, p_{i,j}))$ for some j, where $p_{i,j}$ is the jth vertex on p_i. The shortest paths p_i come from a *highway decomposition*, which is a set of shortest paths. For two p_i, p_j from a highway decomposition, $p_i \cap p_j = \emptyset$ has to be fulfilled. That means that two shortest paths from a highway decomposition must not share any vertices. Furthermore, the union of all shortest paths of the highway decomposition has to contain all vertices in the graph, $\bigcup p_i = V$.

To answer distance queries with a source s and a target t, where there is a label

for $p_{i,j} \in L(s)$ and $p_{i,k} \in L(t)$, the path $s \to p_{i,j} \to p_{i,k} \to t$ is a candidate for a shortest path. The distance $d(s, p_{i,j})$ is stored with the label of s, the distance of $d(p_{i,k}, t)$ is stored with the label of t, and the distance $d(p_{i,j}, p_{i,k})$ can be computed with $|p_{1,j} - p_{1,k}|$, which is also stored with the labels. $s \to p_{i,j}$, $p_{i,j} \to p_{i,k}$ and $p_{i,k} \to t$ separately are shortest paths, but combined they do not have to be. Assuming that there are two labels in $L(s)$ and $L(t)$ such that the shortest path from s to t goes from the source s to the vertex $p_{i,j}$ of the shortest path p_i, to the other vertex $p_{i,k}$ on the shortest path p_i, to t, they propose a method to find those two labels in linear time.

A shortest path from s to t does not have to lie on p_i, but it could be a detour instead. Therefore, each vertex v has a label to every shortest path p_i in the highway decomposition. Furthermore, even if the shortest path from s to t lies on p_i, it does not necessarily lie on the segment from $p_{i,j}$ to $p_{i,k}$. Having labels for all intermediate vertices of a shortest path leads to having a label from every vertex to every other vertex, yielding a space consumption of $\mathcal{O}(n^2)$ Instead, the authors propose a pruning mechanism to only select some end vertices $p_{i,j}$ on the shortest paths of the highway decomposition.

First of all, the number of shortest paths itself plays an important role in the label sizes. Therefore, a highway decomposition should be found with few shortest paths necessary to cover all vertices of the graph. For that, they use a unique property of road networks: speed limits or travel speed. The authors claim that many shortest paths of the vertices incident to a fast edge go through that edge. It is inspired by the intuitive idea that we use highways for many shortest paths, especially for long distances. Therefore, the authors group vertices by the speed of their incident edges and propose a strategy to choose paths p_i which hit many shortest paths early in the construction of the highway decomposition.

That can be done by searching shortest paths within the vertices of a certain level of high speed limits, adding more vertices of the next lower speed limit if there are not enough vertices. From the shortest path tree created by running Dijkstra's algorithm with a random source vertex, the vertices with the most descendants are picked to select one path from the source of the exploration to one of its leaves. Afterwards, the path is picked as one shortest path of the highway decomposition, and the procedure is repeated with the remaining vertices.

Highway-based Labelling has a lower preprocessing time than usual Hub Label approaches. Furthermore, the space consumption is lower, because many distances to other vertices can be computed between the vertices within a shortest path without explicitly storing them. On the other hand, query time is larger than the classical Hub Labelling approach, because a simple scan of both label lists is not sufficient to compute the tentative distances between two vertices s and t.

3.2 Highway Dimension

Highway Dimension [1] and the introduced structures surrounding it build a framework to prove the runtime and space complexity of various route planning algorithms.

Based on the observation that many long shortest paths have common intermediate vertices, the Highway Dimension h describes the number of vertices required to *hit* a vertex on every shortest path of a certain minimum length around any vertex v for any radius $r > 0$.

Using Highway Dimension with Contraction Hierarchies for computing the contraction order, a query can be answered in $\mathcal{O}((h \log D)^2)$ time, where D denotes the *diameter* of the road network, the shortest path with the maximum length in the entire network. Furthermore, with the method proposed, the number of shortcuts is limited by $nh\lceil \log D \rceil$. The same bounds are shown for the RE Algorithm [21] using the same arguments as for the shortcuts of Contraction Hierarchies. The Highway Dimension can further be used to construct the labels of Hub labels [19] and show an upper bound for the labels' size $h(\log_2 D + 1)$ per vertex. That yields a space consumption of $\mathcal{O}(nh \log D)$ and a query time of $\mathcal{O}(h \log D)$. Finally, the Highway Dimension can also be used to choose the access nodes for Transit Nodes, with $\mathcal{O}(h)$ many access vertices per vertex. That leads to a space consumption of $\mathcal{O}(nh)$ and a query time of $\mathcal{O}(h^2)$.

The Highway Dimension h is the number of vertices needed to hit all $(r, 2r)$-close r-significant paths for any vertex $v \in V$ and any radius $r > 0$. Hitting a path p means to have a hitting set $H \subseteq V$ where there is a vertex $v \in H$ such that v lies on the path p. An r-significant path is a shortest path which has an r-witness. The r-witness is a shortest path which is an extension of its r-significant path by optionally one edge on either ends. Intuitively, an r-significant is a shortest path that has roughly length r, with at most the length of its two edges on both ends missing to fulfil that length r. An r-significant path is (r, d)-close to a vertex v if its r-witness contains a vertex u such that $d(v, u) \leq d$.

The Highway Dimension h says that for a minimal local hitting set H for a vertex v to hit all the r-significant paths in a $2r$ radius, $|H| \leq h$ is guaranteed. The authors show that a global hitting set C with minimal size to hit all r-significant paths in the entire graph is furthermore locally sparse, calling it *sparse hitting set*. That means that $|C \cap B_{2r}(v)| \leq h$, where $B_{2r}(v)$ is the ball around v with radius $2r$, which is the set of all the vertices w such that $d(v, w) \leq 2r$. The authors claim that finding that sparse hitting set and determining h is NP-hard, although they propose a polynomial time approximation.

As the radius r can be arbitrary, the paper further introduces a *multi-scale sparse hitting set* which discretises all minimal global hitting sets of an arbitrary radius to radii of doubling size. The sparse hitting set C_i is then defined to be the corresponding hitting set for the radius 2^{i-1}. All sets C_i with $0 \leq i \leq \lceil \log_2 D \rceil$ together form the multi-scale sparse hitting set, starting with C_0 which hits all $1/2$-significant paths and ending with $C_{\lceil \log_2 D \rceil}$ which hits all $D/2$-significant paths.

For Contraction Hierarchies, the multi-scale sparse hitting set is used to define the contraction order. A vertex v of level i, where for all $j > i$ holds $v \notin C_j$, is contracted before any vertex of level $j > i$. The authors show that for an arbitrary $v \in V$ and

level j, there are at most h many outgoing shortcut edges to a vertex w of level j. That limits the maximum degree to $h + h\lceil \log D \rceil$ and the overall number of shortcut edges to $nh\lceil \log D \rceil$. With that, the paper further manages to prove the previously introduced upper bound for the asymptotic space complexity on distance queries.

The multi-scale sparse hitting set can further be used to define the labels for the previously introduced Hub Label algorithm. The multi-scale sparse hitting set can be used to define the labels of Hub Labels as $L(v) := \bigcup_{i=0}^{\lceil \log_2 D \rceil} C_i \cap B_{2^i}(v)$. One of the properties of multi-scale sparse hitting sets is that they are locally sparse, so $|C_i \cap B_{2^i}(v)| \leq h$ for any i and $v \in V$. Therefore, the label size is at most h per level i, yielding a maximum total label size of $h(\log D + 1)$.

4 Highways in Road Networks

To detect highways and highway-like structures, we begin our work with looking at real road networks A definition for highways should be based on the observations in road networks and closely resemble their structure. Therefore, we need to make some basic observations of the special properties of highways. Based on such observations, we introduce a formal definition for the highways of a network, which we can verify at a later point.

To get an understanding of what a highway is, we start with the definition of *motorways* in OpenStreetMap, the road type with which we will compare the resulting highway of our highway definition. OpenStreetMap describes motorways as "the highest-performance roads within a territory". It is restricted to "roads with control of access, or selected roads with limited access depending on the local context and prevailing convention" [34]. With this definition, we can extract two properties of highways in road networks.

First of all, a high-performance road means they are usually designed for fast multi-lane traffic. In other words, highways can have higher speed limits and therefore reduce travel time over the same distance compared to subordinate road types. A consequence is that parts of highways are more likely to be selected on shortest paths than other types of roads. The longer a shortest path gets, the more time benefit is achieved using highways. This might compensate for even bigger detours when choosing the highway instead of the most direct path with the shortest distance.

The second property is the controlled access with on-ramps and off-ramps. While it does not necessarily mean that there are fewer connecting points between highway segments, it is an indication that it yields fewer connecting points over the same distance as other road types where an intersection is much easier to build. For the representation of the network as a graph, it means that there are fewer vertices on highways with a degree greater than 2 over the same distance than on other road types. All vertices and edges with degree 2 can be contracted to one edge with the accumulated travel time and distance. That means that highways can overall have fewer edges to cover the same distance.

The arguments used in the efficient route planning algorithms we introduced earlier in the Related Work section further emphasise that highways are roads or road segments which are part of many shortest paths, especially long ones. Furthermore, it should be noted that the number of intersections within a certain area is limited. Road networks are a representation of a real-world system where the space to put roads in a certain area is physically limited. Transferring this conclusion to the graph representation, the number of local vertices is also limited, aside from vertices of degree 2 which can be contracted as they do not represent an actual intersection. This limiting factor called *Bounded Growth* has been the subject of studies for upper bounds of efficient route planning algorithms [18]. For our analysis of highways, it means that most vertices are non-local to a vertex, which again means that most shortest paths are long-distance paths. As highways are designed to cover large distances, the property

of most shortest paths being long-distance paths further emphasises that highways are overall used for most of the shortest paths in a network.

To sum up the observations and properties we will take into consideration with a formal definition of a highway structure:

- Highways usually have high speed limits.
- The farther two vertices are apart, the more benefit using a highway has, as it compensates an additional detour from the most direct path.
- The high speed limits make highways optimal candidates for parts of shortest paths, especially for long shortest paths.
- Most shortest paths in a road network are long shortest paths. Long shortest paths are even more likely to contain highway segments. Therefore, most shortest paths will contain some highway.
- Highways have controlled access with on-ramps and off-ramps. Hence, highways usually have fewer edges over the same distance than other road types.

Note that those observations are generalised and that they do not have to apply to the full extent. Roads are the result of historical and political developments. Those networks are not always optimised for efficiency but are influenced by external constraints. How well those general properties hold and can be utilised to detect highways in real road networks is subject of this work as part of its evaluation.

4.1 Defining Highways

With the previously listed properties of road networks, we can consider a few definitions on finding the best candidates for the highways of a road network. Not all properties of real highways can be used in a general network analysis. Some information such as speed limits might not even be present in road networks depending on the data source, and they are especially not present in other types of networks. Therefore, we focus on criteria which can be generally used without additional information about the network. We will introduce a few definitions and give some detailed explanation for their choice, what we first expected and how they played out to work in practice.

For our road networks, we need to assume that we have two different edge weights: distance and travel time. The travel time is generally used to determine the shortest paths in the network, yielding the fastest paths. The distance is needed for some of the following definitions of a highway network, as well as for the evaluation of the distance coverage in comparison with the official highways. Additionally, we assume that all shortest paths in a road network are unique. While this does not have to apply to all shortest paths, the number of shortest paths between a source-target pair is small. That is because road segments have a length and travel time whose discretisation is not exhausted with the given network sizes. In other words, even two

similar paths between the same source-target pair have a small difference in length, just because it is unlikely in the real world for two roads to have the exact same lengths. Since the number of alternative shortest paths in real road networks is so small, its impact on our results is also negligible. If unique shortest paths need to be guaranteed, it can be artificially enforced by altering the weights at a low decimal place.

4.1.1 Edge Betweenness Highways

One of the first things that might come to mind for finding parts of the network with many shortest paths is looking at betweenness centrality. Betweenness centrality measures the importance of a vertex for the shortest paths by counting the shortest paths that go through that particular vertex. The basic idea has first been introduced to describe the impact of a middleman in the communication between humans in the field of sociology [8] and has then been formalised as *rush* [4]. Finally, it received its today's name and definition as *betweenness centrality* from Freeman [17].

The betweenness centrality ranks vertices based on their appearance in shortest paths. In our graph representation of road networks, vertices represent intersections. However, for finding highways we are less interested in finding the most important intersections, but instead the most important road segments. For that, there is a variation of betweenness centrality, called *Edge Betweenness*. It has been introduced early on by Anthonisse [4] as the rush of an edge. The definition of Edge Betweenness used in our work can be found in Definition 4.1. Edge Betweenness satisfies many of our previously elaborated properties of road networks. For each edge, it counts the number of shortest paths in which the edge is included. With this, we can create a ranking of the edges which are on the most shortest paths. Together with the property that most shortest paths are long shortest paths, Edge Betweenness both counts edges on shortest paths and implicitly emphasises on long shortest paths, as edges connecting two distant regions are expected to be used much more often than edges only connecting local areas. The following definition of Edge Betweenness [4, 8] is based on the definition of betweenness centrality by Brandes [9].

Definition 4.1 (Edge Betweenness)**.** Let $\sigma_{s,t} := |SP(s,t)|$ be the number of shortest paths from $s \in V$ to $t \in V$ and $\sigma_{s,t}(e) := |\{p \mid p \in SP(s,t) \wedge e \in p\}|$ the number of shortest paths from s to t via $e \in E$. The Edge Betweenness $C_{EBTW}(e)$ of $e \in E$ is defined as:

$$C_{EBTW}(e) := \sum_{s \neq v \neq t} \frac{\sigma_{s,t}(e)}{\sigma_{s,t}}$$

To extract what we believe is a highway, we take the k edges with the highest Edge Betweenness. However, since real road networks are mixed networks which may contain both undirected and directed edges, we do not want to take the original Edge Betweenness values. It would give directed edges a disadvantage over undirected edges, as they are only selected in one direction while undirected edges can be selected in both directions. On the other hand, we want to be able to compare our highways with the original network. Replacing undirected edges with directed edges might

put us in a situation where one direction is selected as a highway, and the other direction is not selected. In that case, the highways cannot be compared to those of the original network where by definition the entire undirected edge is either a highway or no highway. As a compromise, we divide the Edge Betweenness by two for any undirected edge. That gives our first highway definition.

Definition 4.2 (Edge Betweenness Highways)**.** Let k be the number of edges for the highway network. $H_{EBTW}(k)$ is the highway edge set of top k edges with highest Edge Betweenness value:

$$H_{EBTW}(k) := \max_{E' \subseteq E, |E'|=k} \sum_{e \in E'} \frac{C_{EBTW}(e)}{\begin{cases} 2 & e \in E_U \\ 1 & \text{else} \end{cases}} \qquad m \geq k \in \mathbb{N}$$

For the following definitions, we will exclude the discrimination between undirected and directed edges. We exclude it to make the definitions easier to read, however we will perform this discrimination in our evaluation on the other definitions, too.

As Edge Betweenness is an established metric of network analysis, we consider the Edge Betweenness as the baseline for further definitions. To compare the highways resulting from this definition with the real highways of the underlying road network, we choose k to be equal to the number of edges which are officially classified as highways.

4.1.2 Highway-Paths Ratios

The selection of highway edges with Edge Betweenness is a local way of looking at highway networks. It takes each edge individually and selects them based on their rank. If some edges of the same path appear together, it is only because they are sub-paths of many shortest paths in the network. While it yields decent results in finding paths which are part of many shortest paths, it does not explicitly optimise for that criterion.

We are looking for a definition which better describes the highway network. For that, we zoom out from a local perspective and look at the highway network as a whole. We want to find a sub-network within the entire road network. We want to specify this further. In real road networks, there can be independent highways which connect different areas. To get from one highway to another, you might have to use a subordinate road type. Hence, we do not require the highway network to be connected. We want to select the edges of the highway network to 'cover' as many of the shortest paths of the network as possible.

Covering most the shortest paths can mean different things. The simplest definition measures the ratio of edges of each shortest path within the highway network. Then, we choose the highway network which has the highest ratio of shortest path edges inside the highway network. However, keep in mind that for a general definition, there can be non-unique shortest paths. That means, for a source-target pair, there are multiple shortest paths with at least one different edge from each other. As

long as a path is a shortest path, we can choose any other shortest path instead. In other words, a highway network does not to have most shortest paths, but at least most of one shortest paths between a source-target pair. Hence, we are content if a large portion of one shortest path of a source-target pair is included in the highway networks. This path can always be preferred over the others. Therefore, we only want to consider the shortest path with the biggest proportion of edges inside the highway network E'. With all these considerations, we conclude the following definition:

Definition 4.3 (Highway-Paths Ratio)**.** Let k be the number of edges for the highway network, then is $H_{HPR}(k)$ the highway edge set based on Highway-Paths Ratio:

$$H_{HPR}(k) := \max_{E' \subseteq E, |E'|=k} \sum_{s,t \in V, s \neq t} \max_{p \in SP(s,t)} \frac{|p \cap E'|}{|p|} \qquad m \geq k \in \mathbb{N}$$

Like with Edge Betweenness, the choice of k determines the size of the highway network. For a comparison with the real highways, k is chosen to be the same as the number of edges of the officially classified highways.

A logical next step is a variant which measures the ratio of distance inside and outside the highway network. For this, we do not only need the travel time as an edge weight for the graph representation of our road network, but also the distance of the endpoints of a road segment. We will indicate this by l_d, in contrast to l for the travel time. That brings us to Definition 4.4 of Distance-Based Highway-Paths Ratio.

Definition 4.4 (Distance-Based Highway-Paths Ratio)**.** Let k be the number of edges for the highway network, then is $H_{HPR}(k)$ the highway edge set based on Distance-Based Highway-Paths Ratio:

$$H_{DHPR}(k) := \max_{E' \subseteq E, |E'|=k} \sum_{s,t \in V, s \neq t} \max_{p \in SP(s,t)} \frac{l_d(p \cap E')}{l_d(p)} \qquad m \geq k \in \mathbb{N}$$

However, the Distance-Based Highway-Paths Ratio has some practical flaws which we will discuss in the evaluation of Section 5.1.2. What we learn from the evaluation of Distance-Based Highway-Paths Ratio leads to the definition of Highway Power-Distance.

4.1.3 Highway Power-Distance

According to previous work mentioned in the Related Work section, highways are parts of the network with many shortest paths, particularly long ones. The number of long shortest paths is larger than the number of short ones. Hence, long shortest paths get emphasised more with measurements like the Edge Betweenness. However, road networks do not stop at state or country borders. A highway leaving a region has no use inside of the region's network. This highway only makes sense in the embedding of a larger road network, where regions are usually inter-connected. When we compute the Edge Betweenness for a regional network, it will only consider parts

of the network which connect many source-target pairs within the regional network. This behaviour detects highways closer to the centre of the network well. However, highway segments closer to the border of the network are only used for few source-target pairs. That means those edges will not be selected as highways within the regional network.

If we assume that a regional network is isolated and does not attempt to connect places outside of its borders, you can argue that those roads should not be highways, as their construction costs outweigh their benefits. However, real road networks usually want to connect places outside of their state borders. That means we observe those networks close their border in real road networks. Yet, we want to be able to detect those highways. One way to achieve that is by taking a larger network surrounding the region we want to evaluate. Then, we run our computations and only select the highways which are within the region's border. That way, the usefulness of a connection is not only limited to the region, but a road is also selected as a highway if it connects many paths outside of the region. However, this approach can only be applied if we have access to a surrounding network. Furthermore, an extended network including surrounding areas have larger network dimensions. With the computational complexity of Edge Betweenness in $\mathcal{O}(nm + n^2 \log n)$ [9, 10], the computational effort is significantly larger on an extended network.

With Highway Power-Distance, our goal is to simulate more long-distance connections by weighting long shortest paths higher than short ones. We achieve that by taking the length in terms of distance into consideration. For that, we use the distance l_d, just like Distance-Based Highway-Paths Ratio in Definition 4.4 on page 23. A parameter $q \geq 0$ parameterises the amplification of the distance.

Definition 4.5 (Highway Power-Distance)**.** Let $q \geq 0$ be the amplifier of the path distance and k be the number of edges for the highway network. $H^q_{HPD}(k)$ is the highway edge set based on Highway Power-Distance amplified by q:

$$H^q_{HPD}(k) := \max_{E' \subseteq E, |E'|=k} \sum_{s,t \in V, s \neq t} \left| \operatorname*{argmax}_{p \in SP(s,t)} l_d(p)^q \right| \max_{p \in SP(s,t)} l_d(p)^q \qquad m \geq k \in \mathbb{N}$$

4.2 Methods on Finding Highways

In order to find highways in road networks, we have to take the definitions introduced earlier and create algorithms to compute their results. All of the variants solve an optimisation criterion where we have a highway edge set E' of a previously determined size $|E'| = k$ which contains the best edges on highways, based on the different criteria how to select them.

4.2.1 Edge Betweenness and Brandes' Algorithm

First, we need to compute the Edge Betweenness of all edges for our baseline approach taking the top k edges with the highest Edge Betweenness in the network. For this, we

use Brandes' algorithm which has been created for betweenness centrality [9] with a variant for Edge Betweenness introduced later [10]. Brandes' algorithm goes through many runs of a single-source shortest path exploration (SSSP) based on Dijkstra's algorithm [14]. Each run traverses the graph from a source to all other vertices in the graph. In contrast to the original Dijkstra's algorithm, it does not maintain a data structure to exactly one preceding vertex v on the shortest path to the currently settled vertex w, which is updated when a shorter path is found leading to w. Instead, it maintains a list of preceding vertices with all the predecessors which yield the same distance to v. That way, all the shortest paths to a vertex are counted, not only one of them. Furthermore, a stack S tracks the order in which vertices are settled by Dijkstra's algorithm, by pushing the settled vertex of every iteration on the stack. That order is used later in an accumulation step. Furthermore, a counting data structure σ is maintained which keeps track of the number of shortest paths leading to v. For each new predecessor w, $\sigma(v)$ is updated to also be reachable by all the shortest paths which lead to w, so $\sigma(v) := \sigma(v) + \sigma(w)$.

After one full traversal from a source s, we retain a shortest path structure. It carries the information of the different shortest paths from s to any vertex in the structure. With that, the accumulation step can compute either Betweenness Centrality or Edge Betweenness depending on the implementation of the accumulation. The shortest paths are explored backwards by going through the elements of the stack S which kept track of the settling order of all vertices. The accumulation takes the computed values σ for each vertex and uses it to compute either the betweenness centrality or Edge Betweenness (see Definition 4.1 on page 21) depending on the implementation.

This procedure of traversing the graph from a vertex s and then accumulating the betweenness values for all vertices or edges is repeated for all vertices $s \in V$. The asymptotic exploration runtime of each step has not increased compared to Dijkstra's algorithm because all additional operations of Brandes' algorithm in an iteration can be executed in constant time. The accumulation phase goes through the stack of settled vertices w, where an operation is performed for each predecessor of w. The stack size is n, while the number of predecessors for all elements from the stack is upper bounded by the number of edges in the graph. There cannot be more predecessors than edges, as a vertex is only a predecessor because there is an edge leading away from that vertex. One iteration of the SSSP phase plus the accumulation phase has therefore a time complexity of $\mathcal{O}(\underbrace{n \log n + m}_{\text{SSSP}} + \underbrace{n + m}_{\text{accumulation}})$, which performed for all vertices $s \in V$ yields a total runtime complexity of $\mathcal{O}(n^2 \log n + nm)$. For a detailed description of the algorithm, see Algorithm 1.

To compute $H_{EBTW}(k)$ (see Definition 4.2 on page 22), we can run Brandes' algorithm for weighted graphs with the accumulation for Edge Betweenness, then divide all undirected edges by 2, sort the edges by that value and take the k edges with the highest Edge Betweenness value. Note that computing the Edge Betweenness is the

dominating factor, so computing it once and storing the results before choosing k makes it faster to use different values of k, therefore varying the size of the highway network.

Algorithm 1 Brandes' algorithm for Edge Betweenness in weighted graphs [10]

```
function EDGE_BETWEENNESS( G = (V, E), w : E → ℝ^{≥0} )
    C ← 0 for all edges e ∈ E
    for s ∈ V do
        settled ← false for all vertices v ∈ V                ▷ SSSP exploration phase
        d ← ∞ for all vertices v ∈ V
        σ ← 0 for all vertices v ∈ V
        pred ← [] for all vertices v ∈ V
        S ← empty stack, Q ← empty queue sorted v by d[v]
        d[s] ← 0, enqueue s on Q
        while Q not empty do
            Dequeue v from Q
            Skip iteration if settled[v]
            settled[v] ← true
            Push v on S
            for e = (v, w) ∈ E do
                if d[v] + w(e) < d[w] then
                    d[w] ← d[v] + w(e)
                    σ[w] ← 0, pred[w] ← []
                end if
                if d[v] + w(e) = d[w] ∧ w ≠ s then
                    σ[w] ← σ[w] + σ[v], Add v to pred[w]
                end if
            end for
        end while
        δ ← 0 for all vertices v ∈ V                          ▷ Accumulation phase
        while S not empty do
            Pop w from S
            for v ∈ pred[w] do
                c ← (σ[v]/σ[w])(1 + δ[w])
                δ[v] ← δ[v] + c, C[e] ← C[e] + c
            end for
        end while
    end for
    return C
end function
```

4.2.2 Highway-Paths Ratio

We will show how to simplify the Highway-Paths Ratio in the context of road networks to make it easier to compute. After the simplification of the definition, we can use a modification of Brandes' algorithm to compute edge-based weights which we can use to optimise the highway network E' according to the criteria of Highway-Paths Ratio. For that, we need to assume that shortest paths in road networks are unique, as described in the introduction of Section 4. We show the simplification for Highway-Paths Ratio (Definition 4.3, page 23), which can be equally applied to Distance-Based Highway-Paths Ratio (Definition 4.4, page 23).

Lemma 4.1. *Under the assumption that shortest paths are unique, $H_{HPR}(k)$ can be simplified such that it can be computed for each edge separately.*

Proof.

$$H_{HPR}(k) = \max_{E' \subseteq E, |E'|=k} \underbrace{\sum_{s,t \in V, s \neq t} \max_{p \in SP(s,t)} \frac{|p \cap E'|}{|p|}}_{=:h_{HPR}(E')} \tag{1}$$

$$\begin{aligned} h_{HPR}(E') &= \sum_{s,t \in V, s \neq t} \max_{p \in SP(s,t)} \frac{|p \cap E'|}{|p|} \\ &\text{Unique shortest paths:} \\ &= \sum_{s,t \in V, s \neq t, p \in SP(s,t)} \frac{|p \cap E'|}{|p|} \\ &= \sum_{p \in SP} \frac{|p \cap E'|}{|p|} \\ &= \sum_{p \in SP} \sum_{e \in p \cap E'} \frac{1}{|p|} \\ &= \sum_{e \in E'} \underbrace{\sum_{p \in SP, e \in p} \frac{1}{|p|}}_{=:h_{HPR}(e)} \end{aligned} \tag{2}$$

$$H_{HPR}(k) = \max_{E' \subseteq E, |E'|=k} \sum_{e \in E'} h_{HPR}(e) \qquad (2) \text{ in } (1)$$

□

With the simplification of Highway-Paths Ratio $H_{HPR}(k)$ in Lemma 4.1, we can compute the value $h_{HPR}(e)$ before choosing the highway network E'. This makes the definition flexible, as we can precompute h_{HPR} for all edges before we choose the highway network E'. That means, we split the computation of the ratio from the

choice of the optimal highway network. If this were not possible, we would have to compute the ratios for each choice of highway edges. Afterwards, we can take the k edges with the highest $h_{HPR}(e)$ value to retrieve $H_{HPR}(k)$. Due to that, we can optimise the highway network efficiently.

Furthermore, $h_{HPR}(e)$ is a weighted count of the number of shortest paths going through e by summing up the inverse of the number of edges on those shortest paths $\frac{1}{|p|}$. What follows is the proof that we can compute this with a modification of Brandes' algorithm.

Definition 4.6. The shortest path tree resulting from the exploration with the source s of Dijkstra's algorithm is called $T(s)$. The sub-tree of $T(s)$ starting at u is called $T(s|u)$.

To retrieve $h_{HPR}(e)$ for all edges $e \in E$, we have to go through all shortest paths from any $s \in V$ to any $t \in V \setminus \{s\}$ and sum up the inverse of their number of edges $\frac{1}{|p|}$. However, doing this for each pair s, t is costly, as there are $n(n-1)$ many pairs, and each edge on the path from s to t would have to be updated. Instead, we can traverse the shortest path trees $T(s)$ from the exploration step in Brandes' algorithm for the source s from the leaves to the root s. On the way to the root s, we can accumulate the values of $\frac{1}{|p|}$ for all shortest paths from s to v, where v resides in the sub-tree $T(s|u)$. We will now show how that can be done and prove formally that it yields $h_{HPR}(e)$.

Definition 4.7. We define $C(s,t) := |p|$ as the number of edges of the only shortest path $p \in SP(s,t)$ from any $s \in V$ to $t \in V$.

Definition 4.8. We define $\sigma_s(v) := \sum_{(v,w) \in E(T(s))} \left(\frac{1}{C(s,w)} + \sigma_s(w) \right)$ for any $v \in V$.

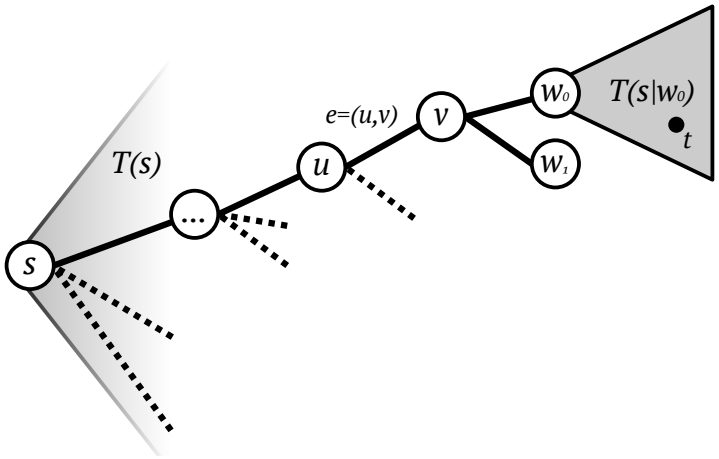


Figure 1: *h_{HPR} can be computed from an aggregation $\sigma_s(w)$ of $\frac{1}{C(s,t)}$ of all t in a sub-tree $T(s|w)$ of w.*

Lemma 4.2. *Let $e \in E$, then $h_{HPR}(e = (u, v)) = \sum_{s \in V} \left(\frac{1}{C(s,v)} + \sigma_s(v) \right)$.*

Proof.

$$
\begin{aligned}
h_{HPR}(e = (u,v)) &= \sum_{p \in SP, e \in p} \frac{1}{|p|} && \text{by def., see Lemma 4.1} \\
&= \sum_{s \in V} \sum_{t \in V} \sum_{p \in SP(s,t), e \in P} \frac{1}{|p|} && \text{SP between } s,t \text{ pairs} \\
&= \sum_{s \in V} \sum_{t \in V(T(s|v))} \frac{1}{C(s,t)} && * \\
&= \sum_{s \in V} (\underbrace{\frac{1}{C(s,v)}}_{\text{Path } s \text{ to } v} + && \text{split trees at } v \\
&\qquad \underbrace{\sum_{(v,w) \in E(T(s))} \sum_{t \in V(T(s|w))} \frac{1}{C(s,t)}}_{\text{Path } s \text{ to vertices in sub-trees of children of } v}) \\
&= \sum_{s \in V} \left(\frac{1}{C(s,v) + \sigma_s(v)} \right) && \text{see Lemma 4.3}
\end{aligned}
$$

* All shortest paths from s to any $t \in V$ which go through e also have to go through v. Therefore, $t \in T(s|v)$. As we assume unique shortest paths, we only have one shortest path between s and t. Therefore, we can drop the inner sum in favour of $C(s,t)$.
For an overview of the components of the proof, see Figure 1. □

Lemma 4.3. *Let $s \in V$, then $\sigma_s(v) = \sum_{(v,w) \in E(T(s))} \sum_{t \in V(T(s|w))} \frac{1}{C(s,t)}$ for $v \in V$.*

Proof. We prove this by induction using the height of sub-trees $T(s|v)$. The height h of a tree is the maximum number of edges from the root to any vertex.

Base case
Let $h(T(s|v)) = 0$, i.e. v is a leaf of $T(s)$.

$$
\begin{aligned}
\sigma_s(v) &= \underbrace{\sum_{(v,w) \in E(T(s))}}_{v \text{ is leaf } \implies (v,w) \notin E(T(s))} \left(\frac{1}{C(s,w)} + \sigma_s(w) \right) && \text{see Definition 4.8} \\
&= 0 \\
&= \underbrace{\sum_{(v,w) \in E(T(s))}}_{v \text{ is leaf } \implies (v,w) \notin E(T(s))} \sum_{t \in V(T(s|w))} \frac{1}{C(s,t)}
\end{aligned}
$$

Induction step
Let $h(T(s|v)) = n + 1$, i.e. v is not a leaf in $T(s)$ and $(v, w) \in E(T(s))$ for some $w \in V(T(s))$.

$$\begin{aligned}
\sigma_s(v) &= \sum_{(v,w)\in E(T(s))} \left(\frac{1}{C(s,w)} + \sigma_s(w) \right) && \text{see Definition 4.8} \\
&= \sum_{(v,w)\in E(T(s))} \left(\frac{1}{C(s,w)} + \sum_{(w,w')\in E(T(s))} \sum_{t\in V(T(s|w'))} \frac{1}{C(s,t)} \right) && * \\
&= \sum_{(v,w)\in E(T(s))} \left(\sum_{t\in V(T(s|w))} \frac{1}{C(s,t)} \right) && **
\end{aligned}$$

* IH on w, $h(T(s|w)) \leq n$.
** For w' with $(w, w') \in E(T(s))$ holds $w' \in V(T(s|w))$. Furthermore, $V(T(s|w)) \setminus V(T(s|w')) = \{w\}$. Hence, the only additional target t with summing over $V(T(s|w))$ is w. Therefore, we can contract the sums. □

Theorem 4.4. *$h_{HPR}(e)$ can be computed with a modification of Brandes' algorithm.*

Proof. According to Lemma 4.2, we can compute $h_{HPR}(e)$ for all edges $e \in E$ by having access to the number of edges $C(s, t)$ of each shortest path from any pair of vertices $s, t \in V$, and to $\sigma_s(v)$ according to Definition 4.8 for any $s, v \in V$.

This can be done with a modification Brandes' algorithm: We can maintain a data structure C by simply counting the number of edges of the tentative shortest paths, like the data structure d from Algorithm 1 on page 26. Furthermore, Brandes' algorithm maintains a stack to keep track of the settling order of all vertices. If we compute $\sigma_s(v)$ in the order of the stack elements, $\sigma_s(w)$ for any outgoing edge of v is already computed by the time v is popped from the stack. That way, we can iterate through all elements in the stack and compute $\sigma_s(v)$, and compute $h_{HPR}(e = (u, v), s)$ as soon as $\sigma_s(v)$ is computed. We sum up $h_{HPR}(e, s)$ for all $s \in V$ to yield $h_{HPR}(e)$. For a detailed description, see Algorithm 2. □

For Distance-Based Highway-Paths Ratios, Algorithm 2 has to be modified to keep track of the distances instead of the number of edges on the tentative shortest paths.

Algorithm 2 Modified Brandes' algorithm for h_{HPR}

```
function HPR( G = (V, E), w : E → ℝ^{≥0} )
    H ← 0 for all edges e ∈ E
    for s ∈ V do
        settled ← false for all vertices v ∈ V                ▷ SSSP exploration phase
        d ← ∞ for all vertices v ∈ V
        c ← ∞ for all vertices v ∈ V
        pred ← −1 for all vertices v ∈ V
        S ← empty stack
        Q ← empty queue sorted v by d[v]
        d[s] ← 0
        c[s] ← 0
        Enqueue s on Q
        while Q not empty do
            Dequeue v from Q
            if settled[v] then
                continue
            end if
            settled[v] ← true
            Push v on S
            for e = (v, w) ∈ E do
                if d[v] + w(e) < d[w] then
                    d[w] ← d[v] + w(e)
                    c[w] ← c[v] + 1
                    pred[w] ← v
                end if
            end for
        end while
        σ ← 0 for all vertices v ∈ V                          ▷ Accumulation phase
        while S not empty do
            Pop w from S
            v ← pred[w]
            σ[v] ← σ[v] + σ[w] + 1/c[w]
            H[e] ← H[e] + σ[w] + 1/c[w]
        end while
    end for
    return H
end function
```

4.2.3 Highway Power-Distance

Highway Power-Distance can be similarly simplified like Highway-Paths Ratio in Lemma 4.1. That yields a definition of a local Highway Power-Distance for each edge, where the edges with the top k values of h^q_{HPD} can be selected, just like with Highway-Paths Ratio and Edge Betweenness:

Definition 4.9 (Local Highway Power-Distance)**.** For an edge $e \in E$ and an amplifier $q \geq 0$, the Local Highway Power-Distance h^q_{HPD} is

$$h^q_{HPD}(e) := \sum_{p \in SP, e \in p} l_d(p)^q$$

The Local Highway Power-Distance can be computed with a modification of Brandes' algorithm similar to Distance-Based Highway-Paths Ratio. Note that for $q = 0$ in networks with unique shortest paths, $h^q_{HPD}(e)$ is equal to the Edge Betweenness $C_{EBTW}(e)$.

4.2.4 Result Optimisations

After simplifying the introduced definitions to be computable with our modification of Brandes' algorithm, we performed further steps to improve the results. One step has already been mentioned with the definition of Edge Betweenness Highways in Definition 4.2 on page 22. All approaches suffer from un unfair treatment of edges in a mixed network in favour of undirected edges. Therefore, after calculating the local values $h(e)$ for all approaches, the value is divided by two if the edge is undirected.

Furthermore, besides using the original road network of a particular region, we also consider computing the results for a larger network surrounding the desired region. With this, we hope to have better results on selecting highway segments on the border of the desired region, which are not selected otherwise. That can happen because a highway on the border of the network is built to connect different regions rather than connecting places within a region. For this, we consider different radii around a road network to be included in the computation of the highway network and discard all edges which lie outside of the desired region only taking those with a high value inside that particular region.

4.3 Approximating Results with Graph Reduction

There are several algorithms proposed to approximate the computation of Betweenness Centrality, some of which could be modified to compute an approximation of Edge Betweenness. The approximation methods can be categorised in three types: source sampling methods, vertex-pair sampling methods and bounded traversal methods [3]. Source sampling does not traverse the graph for all source vertices but only for some of them. Those sources can be selected randomly or based on other strategies, e.g. vertices with a high degree [11]. Vertex-pair sampling methods choose pairs

$s, t \in V$ and determines shortest paths between those pairs to approximate the betweenness centrality [29]. Bounded traversal limits the traversal step of each iteration of Brandes' algorithm to some maximum hops or distance [27].

While our definitions in Section 4.1 are based on Brandes' algorithm and might be approximated similarly to Edge Betweenness, we went for a more general approximation toolset which also works with a definition that is not strictly based on Brandes' algorithm. Our approximation works well for any algorithm which generally measures the overall shortest paths in the network disregarding individual endpoints. This includes the computation of the edge-based values $h(e)$ in the simplification of our definitions in Section 4.2.2.

Our approximation method uses a graph partitioning inspired by Transit Nodes [6]. We start with a vertex and perform an exploration outwards until a certain partition size is reached while having few border vertices which connect the partition with the remaining graph. Afterwards, we create an overlaying skeleton graph inspired by Customisable Route Planning [13] to maintain the original shortest paths between border vertices. Small partitions are merged with their smallest neighbouring partitions to maintain a minimum partition size. By taking the skeleton graphs of all resulting partitions and connecting the border vertices of each neighbouring partitions with each other, the result is a reduced graph with fewer vertices and edges than the original graph while still maintaining the original shortest paths between border vertices of all partitions (see Figure 2). With this, the original shortest paths between border vertices of all partitions can be computed while still having all the necessary information of distance, number of edges and which edges are involved in a shortest path. The skeleton graph can then be used like any other graph to perform operations. The edges of the skeleton graphs are contracted from multiple original edges. We keep track of those original edges, therefore specifically for edge counting algorithms, we can reassign the results of the edges in the skeleton graph back to all edges in the original graph.

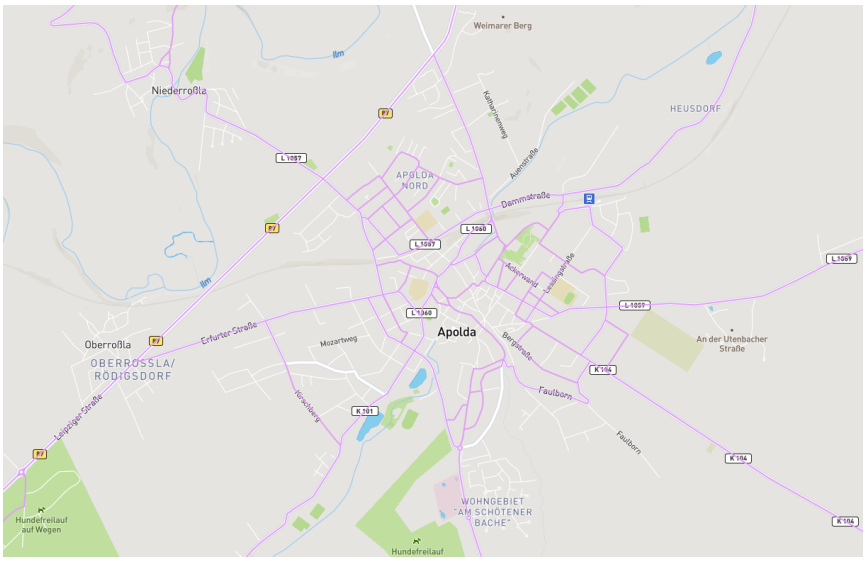


Figure 2: *Extraction of the overlaying graph of the German state Thuringia around the town of Apolda. The skeleton excludes minor roads which are not necessary to connect intersections which are further apart. Visualised with Mapbox.*

Partitioning the Graph

First, we determine a desired maximum partition size k. In our implementation, we choose $k = \lfloor \sqrt{n} \rfloor + 1$. With this, we guarantee that $k \in \Theta(\sqrt{n})$ and as long as the partition size is in $\Theta(\sqrt{n})$, the number of partitions is also bounded by $\frac{n}{\Theta(\sqrt{n})} = \Theta(\sqrt{n})$.

To create partitions for the graph, we start with a random vertex $s \in V$ which does not belong to a partition yet. Then, we explore the graph from s. We call a vertex *stalled* if it can only be reached via a vertex which is part of another partition already. Only vertices which are not stalled are later added to the partition. That way, we can guarantee that all vertices in a partition are connected with each other without leaving the partition, making the partition in itself connected. As soon as $\frac{k}{2}$ vertices are settled, the exploration keeps track of the ratio of number of border vertices to the full partition size and which vertices have been added. The expansion of the partition is extended until k vertices are settled. Since we keep track of the ratio of number of border vertices to the full partition size and the vertices added, we can take the partition of size between $\frac{k}{2}$ and k with the lowest ratio. That way, we minimise the number of border vertices while still attempting to keep the partition size between $\frac{k}{2}$ and k while limiting the exploration size to k. That means, as long as there are $\Theta(k)$ unassigned vertices among the k nearest vertices to s, the partition size is in $\Theta(k)$. A partition which contains fewer than $\frac{k}{4}$ vertices is called a *small partition*. Picking a random vertex $s \in V$ is repeated until all vertices are assigned to exactly one partition.

After all partitions are created, we will merge small partitions with their bigger

neighbours. For each small partition, we pick the neighbouring partition with the fewesut of vertices. A neighbouring partition is a partition which can be reached through a border vertex. A merged partition contains the vertices of both partitions and maintains all border vertices except those between the two partitions. That means that for two merged partitions, the number of vertices is bigger, but the number of border vertices is smaller. Hence, the ratio of border vertices to partition size increases for those partitions. With this merging, we guarantee the partition size to be in $\Omega(k)$ which for $k = \sqrt{n} + 1$ means the number of partitions is guaranteed to be in $\mathcal{O}(\sqrt{n})$.

Creating an Overlaying Graph with Skeletons
In Customisable Route Planning [13], a skeleton needs to maintain all the options of shortest paths independent of the edge weights. However, in our scenario, the cost function is statically predetermined as the travel time. Therefore, we can perform a simpler strategy to create a shortest path skeleton: Shortest paths between all border vertices in a partition are computed. This computation is limited to vertices within the partition. While the path between two border vertices within a partition does not have to be optimal, the edges can still be included. The optimal shortest path between border vertices can then either reached within the partition with the skeleton or via another partition. Any other partition also has a functional skeleton maintaining all possible shortest paths between its border vertices. By computing all the shortest paths between all the border vertices of one partition, we have a superset of vertices and edges needed to maintain the shortest paths between all border vertices in the entire graph. These vertices and edges together form a skeleton.

We can further reduce the skeleton's size by contracting vertices of in-degree and out-degree 1 on that skeleton graph. For that, we replace the vertex v with its edges (u, v) and (v, w) in the skeleton by a single shortcut edge (u, w). Additionally, we keep track which edges have been replaced by this new edge. That yields a skeleton graph where all inner vertices (partition vertices which are not border vertices) have an in- or out-degree of greater than 1. The remaining vertices represent intersecting points of different potential shortest paths from different border vertices. With this, long paths in the skeleton can be reduced to single edges while maintaining all structures necessary to keep and reconstruct the original edge sequences for the shortest paths between the border vertices of a partition.

By definition, a border vertex is connected to another partition through an edge in the original graph. As we maintain all border vertices and the shortest paths between them, we can create a complete skeleton graph which guarantees exact shortest paths between the border vertices of all partitions by taking the vertices and edges of all reduced skeleton graphs of each partition and adding the edges connecting the border vertices.

An example of an original graph and the resulting reduced skeleton graph can be seen in Figure 3. Each orange area describes a partition. The black vertices and edges are from the original graph, the coloured ones are the result of the reduction. The green vertices represent the border vertices of a partition, the blue vertices are inner vertices

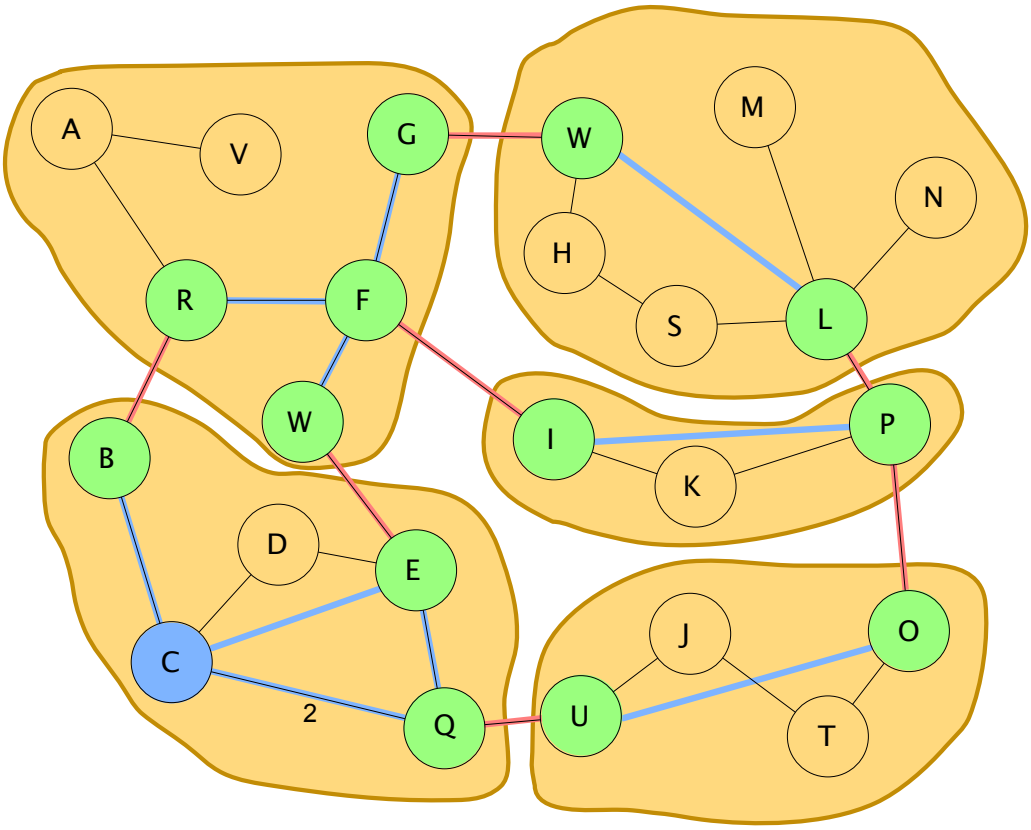


Figure 3: *Example graph with a resulting reduced skeleton graph.*

which could not be contracted in the reduction of the skeletons. For the partition for the border vertices W and L, no intermediate vertices are needed and the border vertices are directly connected with a new edge. Contrarily, the partition with the border vertices B, E and Q require the inner vertex C to maintain the shortest path structure between all border vertices. Overall, the skeleton graph consists of border vertices (green), necessary skeleton vertices (blue), original edges and shortcut edges to connect border vertices (blue) and inter-partition edges between border vertices (red).

We summarise that the overlaying skeleton graph guarantees the exact shortest paths of the original graph between all border vertices. Therefore, the shortest paths can be explored with algorithms like Brandes' algorithm. The Edge Betweenness computed in this overlaying graph only considers shortest paths between the remaining vertices of the skeleton graph. However, the Edge Betweenness values can then be mapped back to the originally contracted edges. Note that shortest paths between the vertices within the partitions are not guaranteed. Nevertheless, in the worst case that means that for paths between skeleton vertices within one partition, traversal to and from the

border vertices are emphasised more. This is not a problem in our experimentation. Overall, the approximation yields an order of Edge Betweenness rather than the actual values, as there are fewer shortest paths in the skeleton graph. If the actual values for Edge Betweenness or some of the other measurements is needed, the partition sizes could be used to amplify the results. For that, a shortest path from within a partition could be assumed to equally likely go through all border vertices. Those estimations of shortest paths from vertices within partitions to vertices in other partitions could then be used to factorise the number of shortest paths going through the edges of the border vertices between those partitions. However, in our scenario where we solely choose edges and paths based on their order and do not need the actual value, it is sufficient for both Edge Betweenness and the other definitions of Section 4.1 to retrieve the computed values from the skeleton graph and assign each value from the shortcut edges back to the contracted edges.

Runtime Considerations We choose $k \in \Theta(\sqrt{n})$. As previously mentioned, the number of partitions is guaranteed to be $\frac{n}{\Omega(k)} = \mathcal{O}(\sqrt{n})$.

To create partitions, an exploration is performed for all vertices which are not part of a partition yet. The exploration is performed locally by settling only k vertices. As road networks are sparse, we can assume that $m \in \Theta(n)$. Hence, for the local exploration of k vertices we can also assume that the number of traversed edges is bounded by $\Theta(k)$. Therefore, each exploration can be done in $\mathcal{O}(k \log k)$ time, which means overall we have a runtime complexity for the initial partitions of $\mathcal{O}(n\sqrt{n} \log \sqrt{n})$. For computing the skeletons, we only explore within one partition, so the exploration is in practice much smaller than exploring the entire graph. We define P to be the set of vertices of a partition and B the set of border vertices of the same partition. Merging partitions can be performed in $\Theta(|B_{min}|)$ time in the size $|B|$ of the smaller partition. For that, the vertices of the smaller partition can just be added to the bigger partition. The border vertices of the smaller one need to be checked against the bigger partition and if they are kept or removed from the list of border vertices by checking for external neighbours. The construction of a partition skeleton is bounded by $\mathcal{O}(|B||P| \log |P|)$, one exploration from each border vertex of the partition.

While the size of the partitions cannot be guaranteed to be in $\Theta(k)$ but only in $\Omega(k)$ due to the neighbour merging, the number of total border vertices decreases by the merging procedure. For now, we can only evaluate empirically how many border vertices there are.

5 Evaluation

We split our evaluation into two parts. In the first part, we verify the definitions we introduced on real road networks and justify our use of the term highways by detecting highways on large networks. We demonstrate that the definition is suitable for detecting both real highways and an intrinsic hierarchical structure which can be leveraged, e.g. for efficient route planning. We do this with a toolkit of a highway-based contraction order for Contraction Hierarchies and a null model comparison between the original and random networks.

After verifying our approaches on road networks, we perform an explorative analysis on other types of networks. For this, we use the toolkit introduced in the first part. Our goal here is to give an outlook of a highway analysis extending the area of road networks. With this, we hope to motivate a further analysis of different types of networks and finding highway-like structures in some of them.

5.1 Highway Networks

To evaluate the detection of highways in road networks with the underlying definitions from Section 4.1, we use a number of different approaches and variants. First of all, we measure how well we identify the officially labelled highways of road networks. We are interested in the number of correctly labelled highway segments, which we measure with precision/recall. Additionally, we want to know to which extent we correctly label the highway network in terms of overall highway length, which we measure with coverage.

After we demonstrate that we can identify real highways with our approaches, we take the evaluation one step further. Our goal is to be able to evaluate our approaches in networks where we do not have information about which edges are part of a highway. Furthermore, we want to have an evaluation toolkit which can even be used on other types of networks where this information is non-existent. For that, we introduce a contraction order solely based on the highway structure with the computed local highway weights h_{HPR}. We compare it with the original contraction strategy, Edge Difference, both in terms of preprocessing time and number of shortcuts. Our goal here is to demonstrate that our definition of a highway is not only capable of identifying real highways but finding an intrinsic hierarchical structure of the network which can be leveraged, e.g. for efficient route planning. To further back our argument that we can detect such structures in road networks, we do a comparative evaluation where we take random networks generated with a variant of the switching method as a null model. We compare the distribution of the local highway weights h_{HPR} between the original networks and randomly generated ones. With this, we want to demonstrate that road networks have a significantly different structure than random networks with the same edge weights and degree distribution.

5.1.1 Evaluation Setup

All of our definitions have been tested with different real-world road networks taken from OpenStreetMap data via Geofabrik dumps [12]. Figure 4 describes how the real-world road networks are retrieved from OpenStreetMap and transformed into a graph structure, which can then be used for further steps. Figure 5 shows the pipeline after the import. It describes loading the road network graph into memory and how to run algorithms on them and evaluate them.

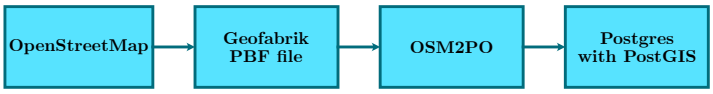


Figure 4: *Pipeline of retrieving the real road network for a particular region. Data is taken from OpenStreetMap via Geofabrik [12]. PBF files are imported into a Postgres database and transformed into a graph with OSM2PO [26].*

Figure 5: *Pipeline for running algorithms and evaluating them for a particular region. Data is loaded from Postgres into a Grph [23] in-memory graph. Afterwards, it needs to be preprocessed and cleaned. From there, we run Brandes' algorithm and custom algorithms. The results are then stored back into the Postgres database or a file. The results are then either evaluated with an external tool, in Java or with SQL and a custom-built webtool for that purpose.*

Preprocessing The raw graph created by OSM2PO, stored in Postgres and later into memory has some artefacts which need to be removed before it can be used for our evaluation. First of all, most of our algorithms only are reasonable and work well in a connected graph. However, the original graphs tend to be disconnected. It can be caused by either islands present in a region, but also by data errors or separate non-public roads.

To solve this problem, we detect the largest connected component by the highest number of vertices and remove all vertices and edges which do not belong to the largest connected component as a first step of preprocessing. This results in a single connected graph which contains most of the original network, excluding those phenomenons of isolated islands, non-public roads or some data errors. Afterwards, we make the graph simple, as some of our algorithms only expect one edge (u, v) between two vertices $u, v \in V$ and expect no loops (u, u) for $u \in V$. Both of those can appear in real road networks. Imagine a residential road which goes in a loop, or two roads being connected to the same intersections on both sides.

The core algorithms we use are based on the shortest paths between points. Therefore, removing a loop or any edge longer than the shortest one between two vertices does not affect the shortest paths in the graph. Hence, we remove loops and all but the edge with the lowest weight between two vertices. However, in real-world networks we also have to deal with both undirected and directed edges. That makes things more complicated, as there can be a directed edge in one direction and a second undirected edge. It is challenging to delete the undirected edge, as the connection in inverse direction of the directed one would get lost. To solve this problem, we say that in a case where a directed edge has a lower weight in one direction and the undirected one has the lowest weight in the other direction, we make the undirected edge directed in the direction for which it has the lowest weight. That way, even a mixed graph can be simplified. See Figure 6 for an illustration of the simplification process.

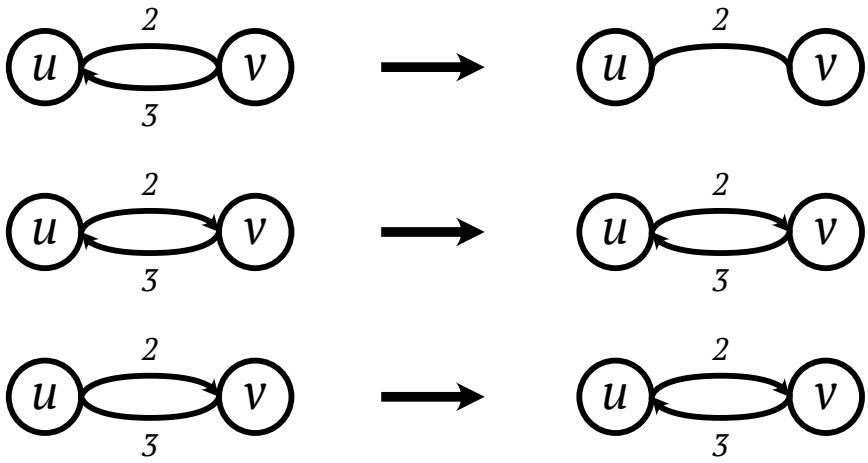


Figure 6: *Different simplification scenarios. Left side before simplification, right side after simplification. Unnecessary edges are removed. If necessary, undirected edges are turned into directed ones.*

During the entire preprocessing, we keep track of the original edge id. That makes it possible to trace back the edge to the original ones. That is especially important when we want to measure how many highway edges we identified correctly, as this information is available for the original edge ids. After all these preprocessing steps, the graphs can be used with our algorithms.

Evaluation set To be able to compare the computed highways with the officially labelled *real* highways, we have to specify what an officially labelled highways is and how we compare the results from the definitions in Section 4.1 of a highway network with the real highway networks. The road segments retrieved from OSM2PO [26] have an attribute *clazz*. They correspond to various road types from OpenStreetMap [33]. We count the number of *motorway* edges from OpenStreetMap according to

their *clazz* attribute. They correspond to the highest tier roads in a road network [34].

Additionally, we count the number of edges of type *motorway link*. Those are connecting sections of motorways either between different motorways or they model their off and on ramps. We define $h :=$ #motorway edges + #motorway link edges and $h' :=$ #motorway edges for each network. While the motorway links are officially also part of the highways of a road network, we will consider both variants comparing our computed highways with the motorway edges, including links and excluding links. For our algorithms, we then specify $k = h$ or $k = h'$ depending on which variant we want to check against.

Selection of networks Our goal is to measure how well the classified highway segments match the officially labelled ones on various road networks. For that, we choose a number of road networks of various sizes and from different parts of the world to run our algorithms on.
We chose four administrative units for our evaluation:

- Thuringia, a state in the centre of Germany.
  $n = 145,363$, $m = 325,131$, $h = 3965$, $h' = 2757$
- Hesse, a state in the centre of Germany.
  $n = 311,430$, $m = 707,322$, $h = 8905$, $h' = 5427$
- Massachusetts, a state in the north-west of the United States.
  $n = 318,825$, $m = 752,186$, $h = 11,197$, $h' = 5578$
- Singapore, a city-state in South-East Asia.
  $n = 56,244$, $m = 90,204$, $h = 3559$, $h' = 1571$

For each network except Singapore, we consider an extended network in addition to the original network. Singapore is surrounded by water and only connected with a few bridges. Therefore, we expect the network to be more isolated than the other networks. For the other networks, we take the surrounding network, e.g. Germany for Thuringia and Hesse, or the US north-east for Massachusetts. Then, we include all vertices and edges in the network where either the source or target of the edge is within a given radius around the original network's border. For our analysis, we will only do this for the two smaller networks Thuringia and Hesse because of the computational effort needed to run the algorithms on those extended networks. Note that by extending a network by a radius, the number of vertices and edges increase significantly. We will later see that this has a large impact on the runtime of the algorithms. For our analysis, we will consider the radii 10, 25, 50, 75 and 100 kilometres. Our goal with extending those networks outside of their administrative borders and run our algorithms on the extended network is to improve the accuracy of detecting highways close to the border of a road network. For those two networks, we will compare both the results of the original networks and their extensions.

Later, we want to use a highway-based contraction order to create shortcuts for Contraction Hierarchies. We will then use the number of shortcuts as a measure to evaluate the contraction order. For that, we will use the four road networks mentioned. Furthermore, we want to evaluate it in one additional network, the road network of Colorado from the 9th DIMACS Implementation Challenge [15]. This case is special because the network is directly taken from a preprocessed file by DIMACS. That means, the Colorado network is not extracted from OpenStreetMaps and loaded via the pipeline of Figure 4 needs to be loaded from a file instead of the Postgres database. By using that existing and publicly available dataset, we want to make our results comparable to other research about Contraction Hierarchies. However, we cannot use the Colorado network from DIMACS for our first evaluation because it contains no road type information which we need for our classification analysis.

Parallel computation All algorithms based on Brandes' algorithm can be computed in parallel by running each exploration and accumulation independently for each source $s \in V$. We use Java's parallel streams to distribute the iterations of Brandes' algorithms with a map operation on all available CPU cores. In the reduction step, we merge the results of two independent runs from two given $s, s' \in V$ by taking the sum of the results for each edge with a value in any of the two runs. Summing up the betweenness values over all explorative runs from an $s \in V$ is done by the original Brandes' algorithm. By doing the explorations separately and summing up in a reduction step, we do not alter the behaviour of the algorithm.

Hardware setup All computations are performed on a machine with two AMD EPYC 7351 16-Core Processors with 2.8 GHz and a total amount of 64 logical cores and 503GiB memory.

5.1.2 Comparison with Real Highways in Road Networks

To validate the results of our algorithms based on the definitions of Section 4.1, we compare the selected highway segments on a real road networks with the official road type labels of real road networks. Our goal is to classify as many real highways correctly as possible.

To quantify our results, we take a traditional evaluation technique from classifications in statistics and information retrieval, precision and recall [22, p. 3-5].

Definition 5.1 (Precision & Recall)**.** Let TP be the number of edges that have been classified as highways correctly, TN the number of edges which have been classified as non-highways correctly, FP the number of edges which have been falsely classified as highways and FN the number of edges which have been falsely classified as non-highways, we define

$$Precision = \frac{TP}{TP + FP}$$

and

$$Recall = \frac{TP}{TP + FN}$$

Note that we compare h (or h') real highway edges to h (or h') predicted highway edges from our algorithms. Each edge that has been falsely selected as a highway segment is a false positive. At the same time, we select one fewer highway edge, since we select exactly h (or h') edges in total. That means, we also have a false negative for every edge that has been falsely selected. Therefore, in this particular scenario, we have $FP = FN$, which means $Precision = Recall$. In the comparisons for the different approaches, we will always have equal highway network sizes between the calculated and the real highways. Therefore, when we only mention $Precision$, we also mean $Recall$, as they are the same.

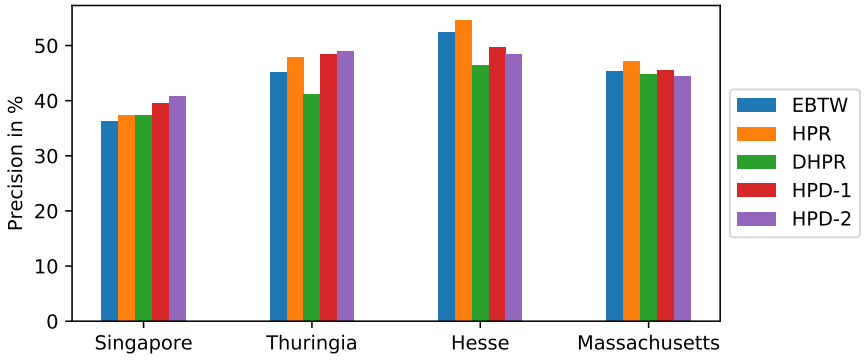
We will use a second measure for comparing the real highways and the computed highway network, which we call *coverage*. It is similar to recall, but instead of counting the number of correctly classified highway segments out of all highway segments, we measure the ratio of 'covered' distance with the correctly classified highway segments. We will indicate the distance by l_d, in contrast to l for the travel time.

Definition 5.2 (Coverage)**.** Let H be the set of classified highway edges and T be the set of the real highway edges we want to check against, then

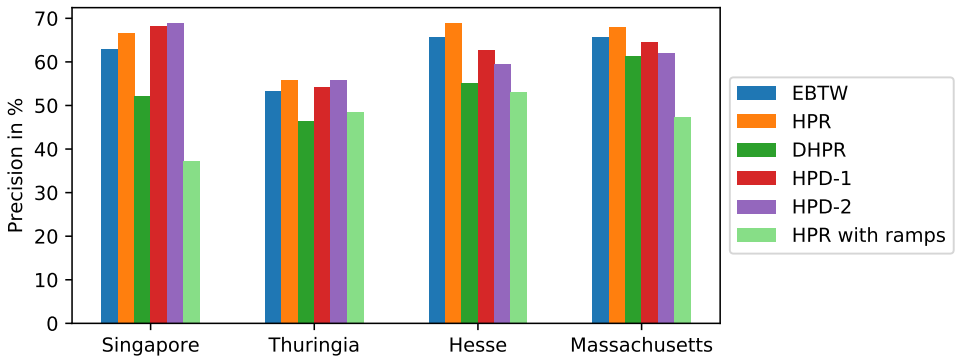
$$Coverage = \frac{l_d(H \cap T)}{l_d(T)}$$

Note that in a graph with uniform edge lengths, i.e. for $l_(e) = 1$ for all $e \in E$, $Coverage = Recall$.

Results First, we will take a look at the precision of identifying the real highways directly in our four road networks Thuringia, Hesse, Massachusetts and Singapore. We compare all four approaches, Edge Betweenness, Highway-Paths Ratio, Distance-Based Highway-Paths Ratio and Highwayness-Power-Distance ($q = 1, 2$). In Figure 7a, we see the results for $k = h$. This includes all ramps and connecting road segments as part of the officially labelled highway edges. Figure 7b shows the results for the choice $k = h'$, which excludes the connecting components, mainly on and off ramps.



(a) *$k = h$, including ramps*



(b) *$k = h'$, excluding ramps.*
HPR with ramps as a guidance for comparison with ramps.

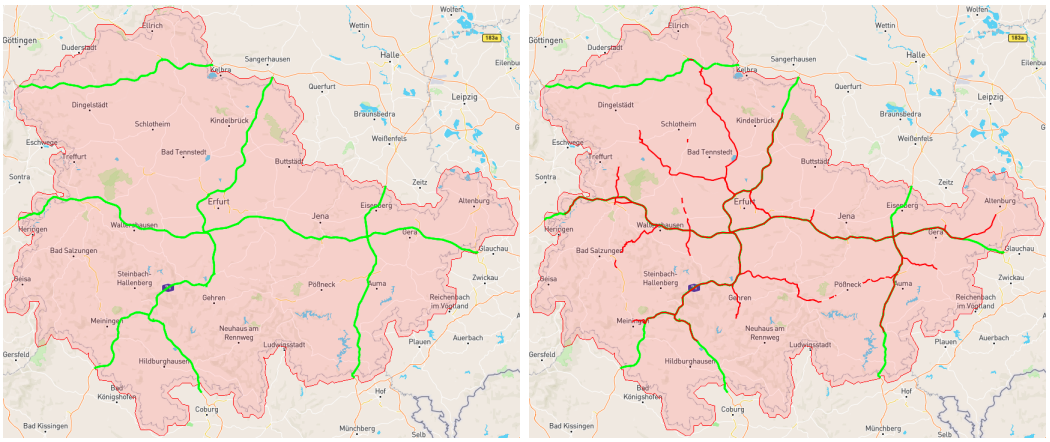
Figure 7: *Comparison of the precision in percentage for the different approaches Edge Betweenness (EBTW), Highway-Paths Ratio (HPR), Distance-Based Highway-Paths Ratio (DHPR), Highway Power-Distance $q = 1$ (HPD-1) and Highway Power-Distance $q = 2$ (HPD-2) compared to the real highways of the four different states. States ordered by their number of edges.*

We observe that all of our approaches perform similarly. However, in contrast to the outer approaches, Highway-Paths Ratio is the only one which outperforms our baseline Edge Betweenness in all four networks. Highway-Paths Ratio benefits from a specific property of highways. As highways usually have restricted access, there are only few intersecting points, on and off ramps. That means, a highway only needs few road segments to cover a large distance. Hence, shortest paths over highways have a low total edge count. Highway-Paths Ratio boosts those shortest paths with low edge counts.

In contrast, we observe that the similar approach Distance-Based Highway-Paths Ratio almost always performs worse in terms of precision than the other approaches. There, we maximise the distance of a shortest path within the highway network instead of the number of edges. We will take a closer look at Distance-Based Highway-

Paths Ratio at a later point.

We see from Figure 11b that all approaches perform better without including the on and off ramps. This can be easily explained: If we include on and off ramps as part of our official highway labels, the number of highway edges is larger, i.e. $h > h'$. On the one hand, that means we need to identify a larger number of highway edges with our algorithms. On the other hand, those additional edges we need to find are on and off ramps. On and off ramps are only part of shortest paths which enter or leave the highway at that particular point. The number of shortest paths entering or leaving the highway on that ramp is naturally expected to be lower than those shortest paths which stay on the highway to connect to even more areas. Instead, our approaches *fill* the remaining edges to be classified as highways with other roads which are almost as important to the road networks as the real highways instead of selecting the on and off ramps. The edges selected instead are usually subordinate road types such as primary roads.



(a) *Real highways of Thuringia* ***(b)*** *Overlay of* H_{HPR}

Figure 8: *Real highways labelled by OpenStreetMap for Thuringia, Germany marked in green.* $H_{HPR}(h)$ *marked in red. The Highway-Paths Ratio covers most of the central highways, but selects many subordinate roads instead of highways close the borders of the state. Visualised with Mapbox.*

We visualise the road network of Thuringia and highlight the real highways and the ones which have been selected by Highway-Paths Ratio. With that, we want to get a better overview of how well our algorithms, specifically Highway-Paths Ratio, perform in selecting real highways. It helps us to get a better understanding why we get a precision of only 65-70%, even with excluding ramps. Representative for the other networks, we visualise the real highways and the detected highways on Thuringia with Highway-Paths Ratio. In Figure 8a, you can see a map with all officially labeled highways of Thuringia in green. In Figure 8b, we overlapped the road segments labelled to be highways by Highway-Paths Ratio in red.

As we can see, highway edges have been selected reliably mainly in the centre of Thuringia. Highway edges closer to the border of Thuringia have not been selected.

Instead, subordinate roads closer to the centre were prioritised. This can be easily explained: Highways close to the border of a state are important to connecting the state to the outside, or connecting other states where the state in the middle functions as a bridge connecting the others. These roads play an important role in connecting different regions beyond state borders. The algorithms we proposed do not consider them because they are less important from an isolated perspective only looking at the state itself, without the surrounding road network. Note that when we look at an isolated network, we are not interested in the highways close to the border. Those highways usually serve little to no purpose if we cannot use them to leave the state network. Instead, our goal with our approaches is to find the highways with a structural importance.

However, we also aim to detect as many official highways in the selected road networks as possible. We introduced Highway Power-Distance with $q = 1, 2$ to compensate for the few connections of highway segments close to the border. Nevertheless, it could not out-perform the other approaches with that amplification. Only in the smaller networks of Singapore and Thuringia, Highway Power-Distance manages to have better results than the other approaches.

If the goal is to find the actual highways of a road network beyond finding the important roads of the isolated state network, the algorithms need to be executed on a larger, surrounding network. Later, the edges can be filtered to only select edges within the state borders. Figure 9 shows the identified highways based on the network of Thuringia including all road segments within a 75 kilometre radius around the state borders (within Germany). That map verifies our assumption that the highways close to the border are important to the overall road network of a superordinate entity. However they play a less important role if we look at a single isolated state not embedded in a larger road network.

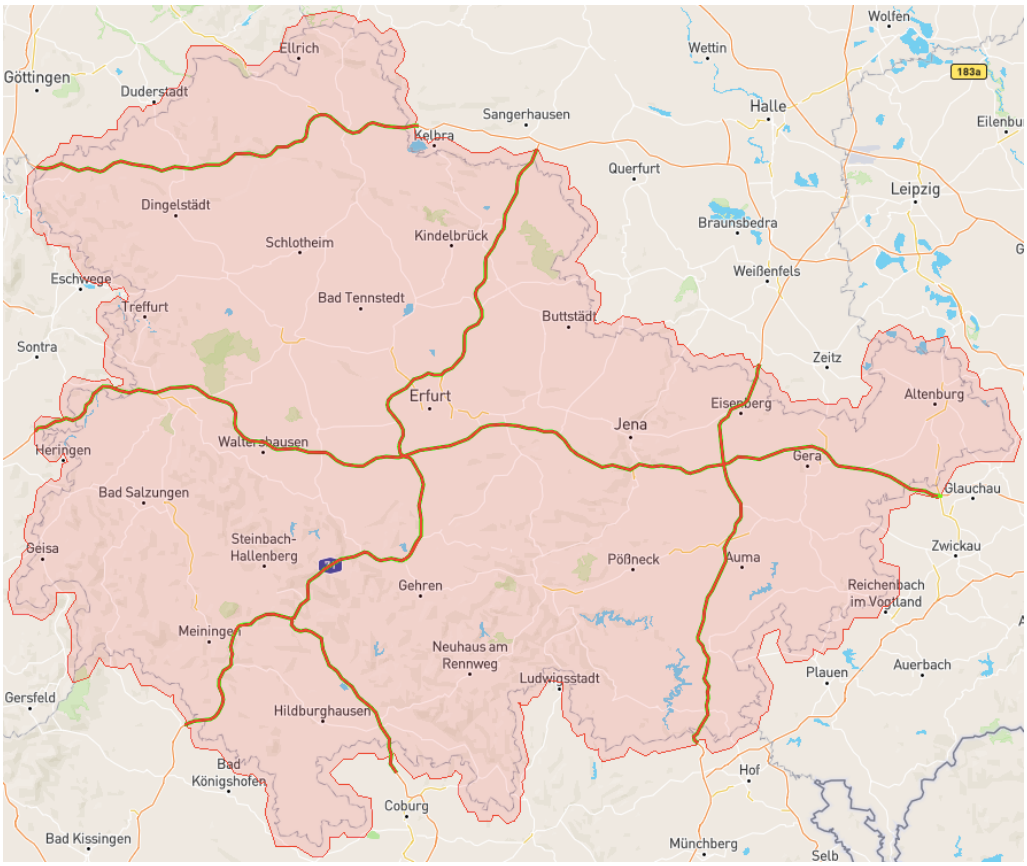


Figure 9: *Real highways labelled by OpenStreetMap for Thuringia, Germany marked in green. H_{HPR} selected from Thuringia including all roads in a 75km radius around the state border and then reduced to the top h inside the border in red. With this, Highway-Paths Ratio covers virtually all real highways. Visualised with Mapbox.*

Coming back to Distance-Based Highway-Paths Ratio, it is performing noticeably worse than most approaches. Figure 10 gives us a hint why that is the case. We can see that the classified edges are more scattered across the entire network compared to HPR in the second map of Figure 8, which selects bigger portions of road segments. That makes sense for these other approaches because the number of shortest do not vary a lot between neighbouring edges on a path. Contrarily, for Distance-Based Highway-Paths Ratio, the length of the edge itself plays an important role into its selection. That results in cases where longer edges on subordinate roads with a decent amount of shortest paths are preferred over short edges on some highways. That makes DHPR less useful in determining and finding bigger portions of highways. The scattering makes Distance-Based Highway-Paths Ratio not suitable for our use case of finding highways.

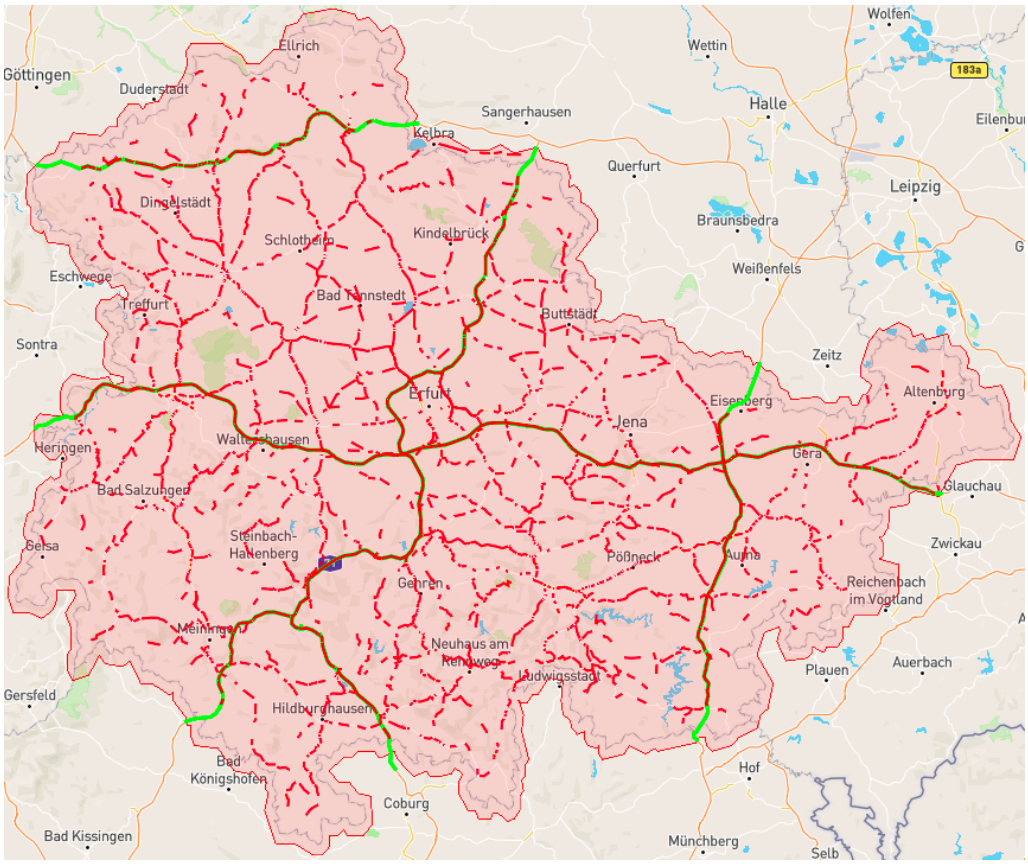


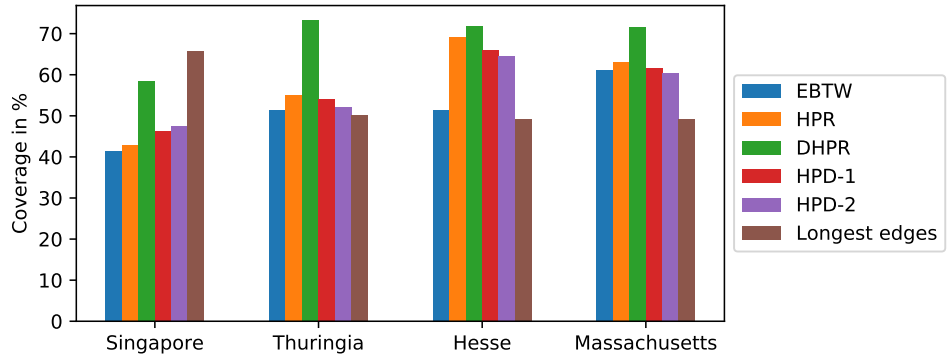
Figure 10: *DHPR selects scattered edges, as the length of an edge plays a role in its selection. Therefore, long non-highway segments are chosen over some short highway segments. Visualised with Mapbox.*

However, in contrast to precision, Distance-Based Highway-Paths Ratio performs very well on coverage according to the results in Figure 11. Coverage measures the correctly classified real highway segments, but not in number of edges but in correctly classified length. Overall, all approaches perform better without including the ramps, similarly to precision.

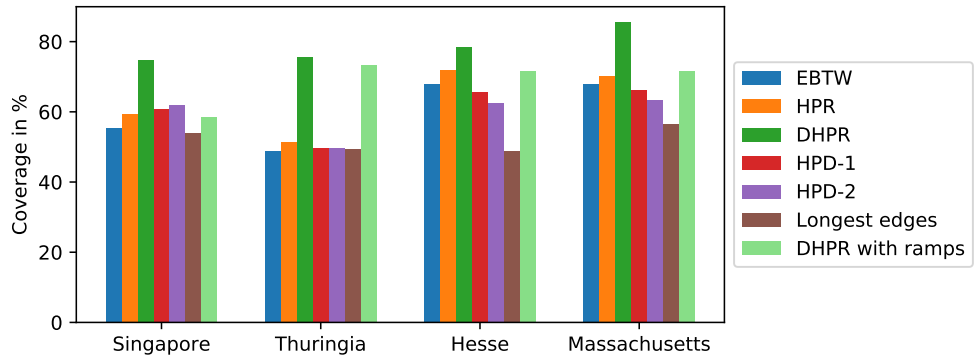
Distance-Based Highway-Paths Ratio prefers long edges over short ones. With that, even with a scattered selection of other non-highway edges which can be seen in Figure 10, it manages to outperform the other approaches in terms of coverage. When we look at the resulting map of Figure 10, we can see that there are large road segments selected which are not part of the official highway network. This raises the question how so many segments can be selected if k is limited to h. If we compare the map with the selection of Highway-Paths Ratio, we would get the impression that more edges have been selected in Figure 10 than in Figure 8. That can be explained with the fact that while Distance-Based Highway-Paths Ratio prefers long edges, the

other approaches consider other criteria than the local length of an edge. With that, many edges selected for the other approaches can be short and are not as visible on a rendered map than the edges selected by Distance-Based Highway-Paths Ratio. Those are generally longer by definition. Even with selecting those long non-highway edges, Distance-Based Highway-Paths Ratio still manages to select enough highway edges to have such a high coverage, and a slightly lower but non-vanishing precision.

Overall, it is questionable if the selection of Distance-Based Highway-Paths Ratio is reasonable. As mentioned before, due to the scattering, it is a bad choice for finding highway segments. Note that we could simply select the $k = h$ longest edges in the network and already reach a coverage of 50% in Thuringia, with a precision of only about 8%. We conclude that coverage overall has to be taken with caution, as its results on its own can be misleading.



(a) *$k = h$, including ramps*



(b) *$k = h'$, excluding ramps.*
DHPR with ramps as a guidance for comparison with ramps.

Figure 11: *Comparison of the coverage for Edge Betweenness (EBTW), Highway-Paths Ratio (HPR), Distance-Based Highway-Paths Ratio (DHPR), Highway Power-Distance $q = 1$ (HPD-1) and Highway Power-Distance $q = 2$ (HPD-2). Additionally, a simple selection with the h longest edge is added as a baseline approach. States ordered by their number of edges.*

After doing the basic evaluation with our two measurements precision and coverage, we will now look at some tuning parameters.

We defined the highway selection with Edge Betweenness in Definition 4.2 on page 22 with dividing the Edge Betweenness value by two if an edge is undirected. With that, we compensate for the undirected edges which can connect shortest paths in two directions instead of one. Further, we argued that most highways are directed, so ranking undirected edges with the same number of shortest paths lower should improve the precision on selecting highways. We demonstrate that for the four selected networks, dividing the undirected edges by two improves both precision and coverage of all approaches. We will specifically look at Edge Betweenness as our baseline approach and Highway-Paths Ratio, which performs better than Edge Betweenness on every road network tested. In Figure 12, we see that dividing undirected edges by two significantly improves precision and coverage.

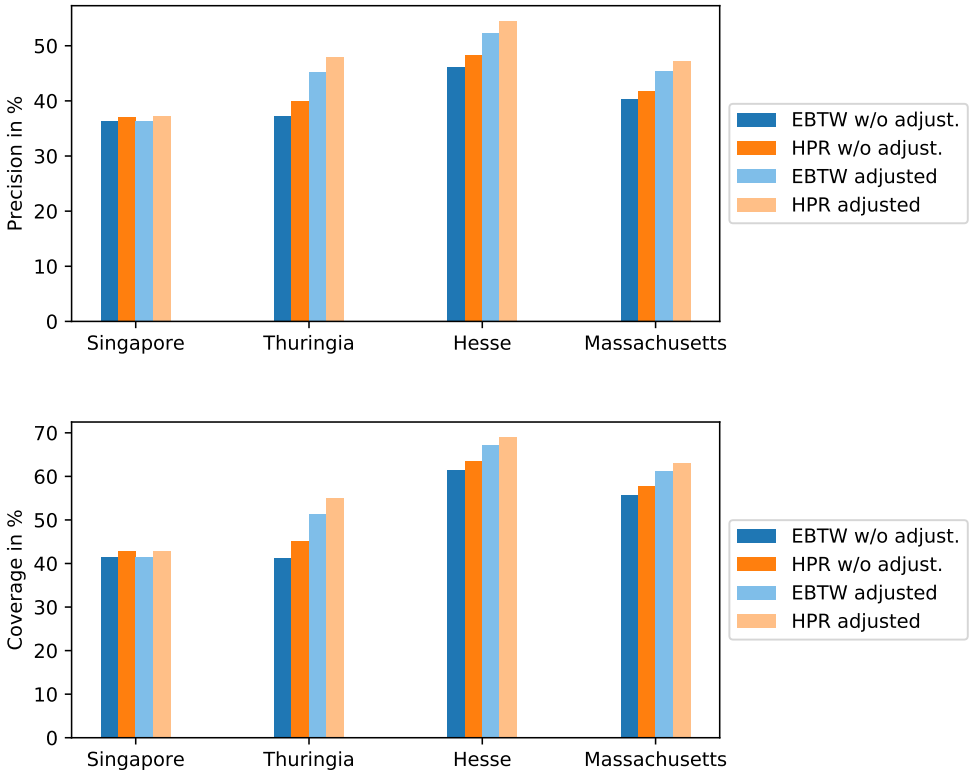


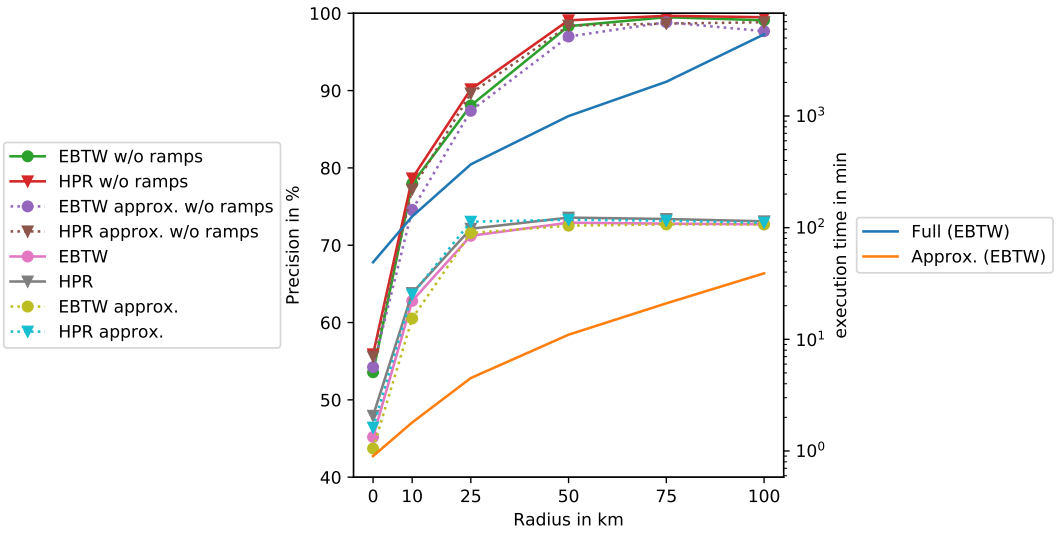
Figure 12: *Comparison of the precision (top chart) and coverage (bottom chart) for Edge Betweenness (EBTW) and Highway-Paths Ratio (HPR) between the raw value without adjustment (w/o adjust.) and adjusted by dividing the value by two for undirected edges. The networks are ordered by their number of edges.*

We remember from the maps of Figure 8 that highway segments closer to the border have not been selected. We could already see in the map of Figure 9 that there

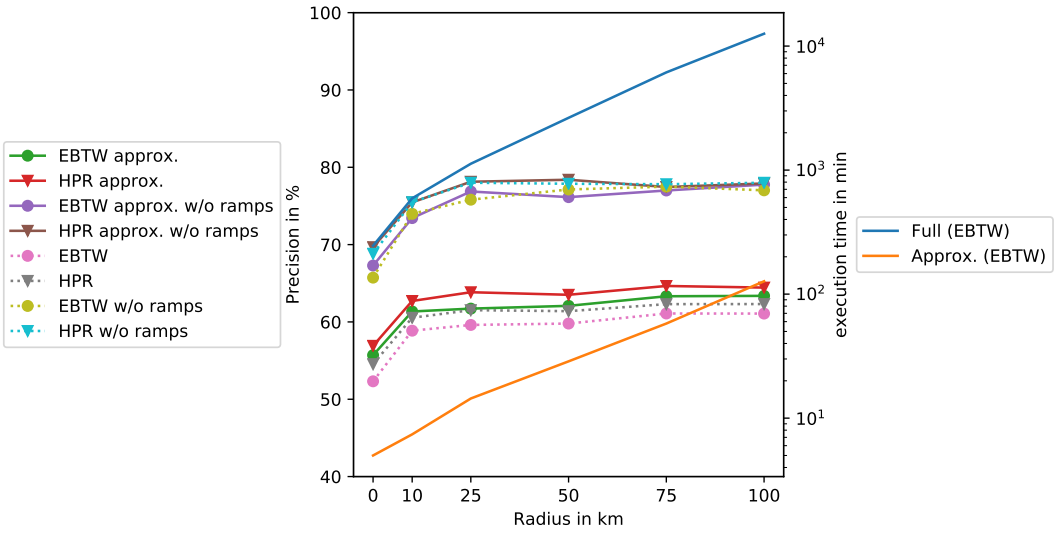
is an improvement with an increased radius of 75 kilometres. A bigger radius increases the network size drastically. Bounded Growth even bases their analysis of runtime of route planning algorithms on the observed quadratic growth of road networks [18]. Therefore, we want to analyse the improvements on precision of different radii. For each of those radii, we need to recompute our measurements. The larger the radius the longer the computation takes. Therefore, we limit our analysis to two networks, Thuringia and Hesse, and the radii 10, 25, 50, 75 and 100 kilometres. With increasing radii, the approximations of H become more important to reduce computational time. Therefore, we also include the results of our approximations based on the definition and algorithm of Section 4.3 on page 32 ff. Additionally, we want to analyse if the precision behaves differently with including or excluding ramps. Figure 13a shows a significant increase of precision with increasing radii on the road network of Thuringia. However, we witness that a point of saturation is reached at a radius of 50 kilometres. The increase is similar for both cases of including and excluding ramps.

As observed before, excluding the ramps overall gives better results as they are not included in many shortest paths. An extended network around the road network Hesse gives an improved precision as seen in Figure 13b. However, some differences can be seen here. The precision starts higher than in Thuringia. Nevertheless, it does not increase as much as in Thuringia. In Hesse, the additional computational effort of a bigger radius quickly outweighs the benefit of including a bigger radius. Many smaller highway segments in Hesse are not considered, while Thuringia has a more clear defined continuous highway network. That makes a big difference in the selection of highway segments with our algorithms, which prefers to select sub-ordinate roads, specifically in the north-west of the state of Hesse where there are no highways. Our approaches prefer to connect those cities with those sub-ordinate roads over smaller highway segments in other parts of the state.

Finally, we want to look at the runtime of the original approaches and the approximations. As we can see in Figure 13, the approximation approaches perform similarly well as the exact ones, even outperforming the original computations in some cases. Overall, we were interested how well the approximations' runtimes are in comparison to the exact approaches compared to the network dimensions, particularly in the number of edges. Therefore, we combined the runtimes of Edge Betweenness of all networks, including the different radii for Thuringia and Hesse, in one plot and compared it with the runtime of the approximated equivalents based on their number of edges. Figure 14 shows how the runtime of the approximation is significantly lower in all networks, with a difference in an increasing order of magnitude. This shows that the approximation is a large improvement in the computation of the highway network, which can be leveraged whenever the exact results are not needed. This becomes particularly interesting in practical applications like the computation of the highway-based contraction order.



(a) Thuringia



(b) Hesse

Figure 13: *Trend of precision with increasing radii with the two approaches Edge Betweenness (EBTW) and Highway-Paths Ratio (HPR). Includes both results of the exact algorithm and their approximations (approx.). For each, we include the results including ramps and excluding them (w/o ramps). On a logarithmic scale, the right axis indicates the execution time of both exact algorithm and approximation in minutes.*

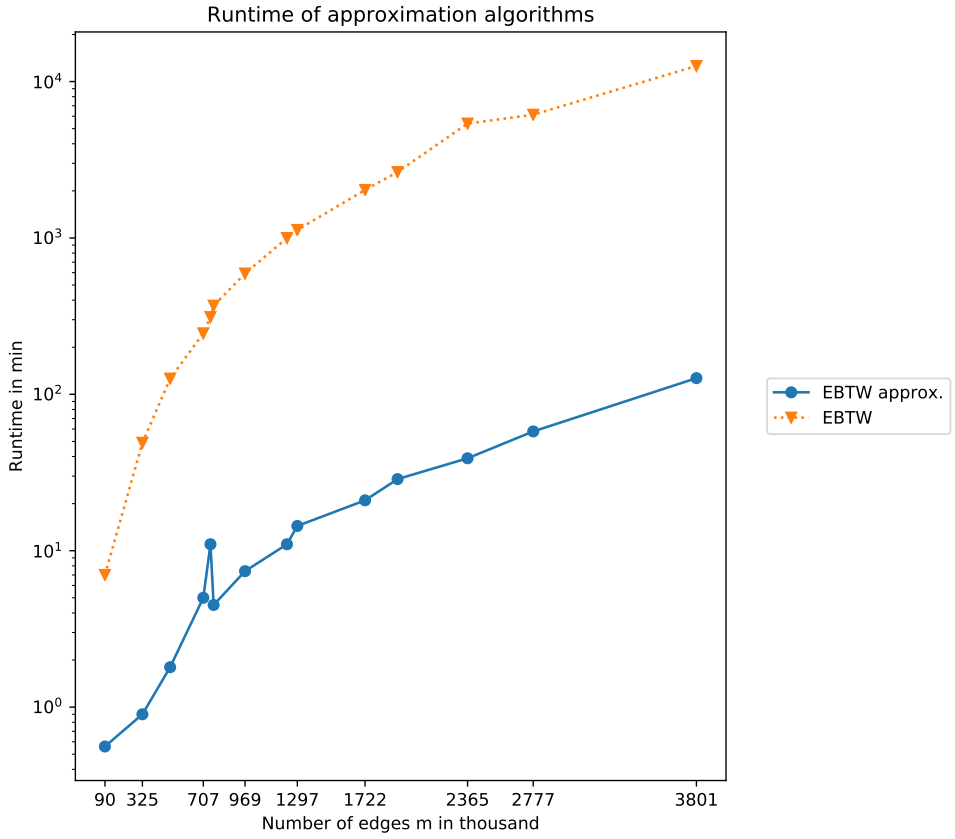


Figure 14: *Runtime of exact Edge Betweenness on different networks with various number of edges compared with the approximated equivalent from Section 4.3.*

5.1.3 Highway-Induced Contraction Order in Contraction Hierarchies

We mentioned before, our approach aims at finding highways which are important to the road network. Therefore, some highway segments close to the border have not been selected as they serve little to no purpose from an isolated perspective. We verify that Highway-Paths Ratio is able to detect an intrinsic hierarchical structure in road networks. This verification is independent of the labels of official highways, therefore it can also be used where we either have no or there is no information of real highways.

For this verification, we use Contraction Hierarchies [20], a fast route planning algorithm which uses shortcuts and traverses them in a hierarchical order. The shortcuts are created by vertex contraction to maintain the shortest paths between the neighbours of a contracted vertex. That means, whenever a contracted vertex v lies on the shortest path between its neighbours, a shortcut edge between those neighbours is inserted. An overlaying graph is created which consists of all original vertices and edges and the additional shortcut edges which have been created by the contraction of all vertices in the graph in a particular order, called the contraction order. The overlaying graph is then queried to retrieve shortest paths, where only edges are allowed to be traversed whose head has been contracted after the tail.

The intuition behind Contraction Hierarchies is similar to the hierarchical structure we expect in a road network. For long distance queries, there are only few local connections needed. The shortcuts of Contraction Hierarchies try to skip many possible connections and directly go to a vertex which is further away, for example a ramp. The longer we traverse the graph, the more selective we are. That means, from our starting point we might want to jump to the next main road, to a road leaving the city, to a highway. Once we reached a high level, we reverse the process with an inverse search from the target to climb the hierarchy in both directions. Therefore, having the knowledge of a highway structure might help us building such a contraction hierarchy.

For evaluating our definition of a highway, we create a contraction order based on our highway network and compare it to the original contraction order *Edge Difference* [20]. We do this with all previously introduced road networks Thuringia, Hesse, Massachusetts and Singapore. Furthermore, we add an additional network to our evaluation which has not been extracted by us via OpenStreetMaps, the road network of Colorado from the 9th DIMACS Implementation Challenge [15]. By using that existing and publicly available dataset, we want to make our results comparable to other research about Contraction Hierarchies.

The original contraction order Edge Difference [20] measures the number of expected difference in the number of edges in the graph after contracting a particular vertex. For that, we take the edge degree of the vertex v and compute which shortest paths from the neighbours of v go through v. For that, we can use a local query starting at all incoming neighbours of v and limiting the exploration to all outgoing neighbours of v. For those neighbours where the shortest path goes through v, shortcuts have to be created to maintain the shortest paths between all remaining vertices in the graph

after the contraction of v. Therefore, the Edge Difference [20] is defined as

$$ED(v) = (\#\text{shortcuts after contracting } v) - deg(v).$$

The vertices are contracted greedily in ascending order of their Edge Difference. After each contraction of a vertex v, the Edge Difference of the neighbours of v have to be updated. To improve runtime, the re-computation and re-ordering of the contraction order can be postponed and performed in batches after a number of contractions [20]. Of course, that has an influence on the contraction order. Hence, this can lead to a different overlaying graph and a different number of total shortcuts. However, it is preferred for the sake of reducing preprocessing time. After contracting all vertices, the result is an overlaying graph of the original graph with all the additional shortcuts which have been added in the process. The additional space consumption of Contraction Hierarchies comes from the number of shortcuts.

For our evaluation, we propose a different contraction order than Edge Difference. Our contraction order uses the highway structure defined by Highway-Paths Ratio in Definition 4.3 on page 23. We show empirically that the highway structure can be used to create an overlaying graph for Contraction Hierarchies through vertex contraction with similarly good results as Edge Difference. With that, we verify that Highway-Paths Ratio is not only able to detect the official highways of a road network, but that it is also able to detect an intrinsic hierarchical structure. This structure can be leveraged for other use cases, such as route planning.

While the computation of Highway-Paths Ratio takes additional time, an contraction order based on Highway-Paths Ratio has the benefit of being created prior to contracting the vertices. Therefore, in contrast to Edge Difference, the values do not have to be recomputed and updated during the contraction. That makes the HPR-induced contraction order faster than Edge Difference for most networks, once the Highway-Paths Ratio values for each edge are already computed. Only in networks where the number of shortcuts is significantly larger with the highway-based contraction order, the overall contraction time is longer. This can be explained by the number of additional shortcuts which make the entire graph denser, hence the execution time for the local queries of each contraction step higher.

Contraction Order Our contraction order is based on a simple observation. On and off ramps are the connecting points between highways and the remaining road networks. Contracting vertices on those ramps splits the network into two parts: the non-highway network and the highway network. In our first experiments, we decided to take the Edge Difference as a base ordering. Then, we chose $k = h$ and selected the highway network of size k from $H_{HPR}(k)$. A vertex was considered a ramp vertex if it was incident to an edge in $H_{HPR}(k)$ and an edge not in $H_{HPR}(k)$. Those ramp vertices were pushed to the end of the contraction order and were therefore contracted last. This contraction order yielded a small reduction in the number of shortcuts. However, the number of those ramp vertices are low, as the entire highway network has only size k and only few of them connect highway and non-highway edges. Therefore, the improvements were minor. Nevertheless, we can use this observation and

generalise the transition from non-highway to highway network from a fixed size of k to a more *continuous* transition from non-highway to highway network.

We designed this so that every vertex has its own scoring of how much involved it is in a highway network. This creates a layered highway network where a vertex with a lower ranking would be considered later and part of a larger, more brought highway network. On the other hand, a vertex with a high ranking would be part of a smaller core highway network. We achieve that by summing up $h_{HPR}(e)$ for all edges e incident to a vertex v. This sum represents its score.

In addition to the highway ranking, we want to contract vertices of degree ≤ 2 first. Those are no ramps, because a ramp is always an intersection connecting at least 3 road segments. Instead, a vertex with degree ≤ 2 is always part of a simple path. If we do not contract those vertices first, the vertices of degree 2 might propagate the creation of shortcuts from the higher degree endpoints of the simple path between all vertices of degree 2 in each contraction step. That would yield a high number of unnecessary additional shortcuts. Figure 15 illustrates why vertices of degree 2 should be contracted before vertices of a higher degree.

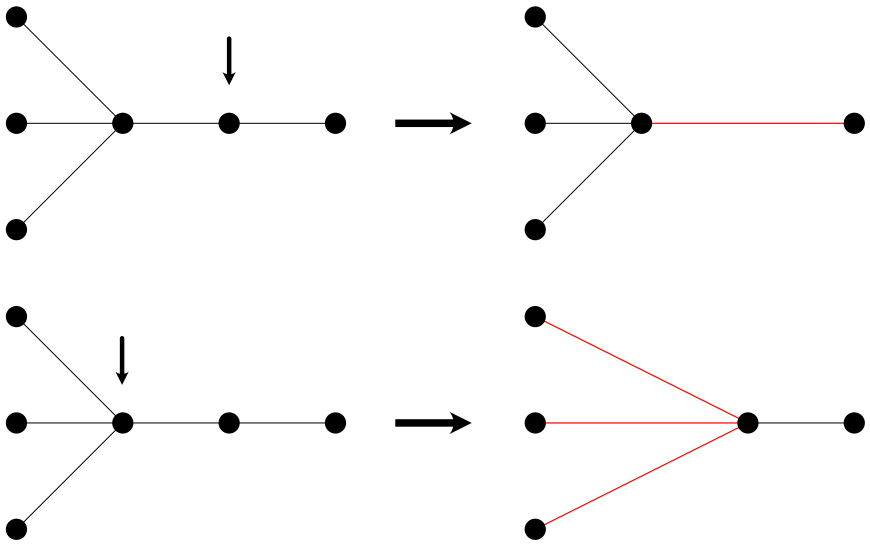


Figure 15: *Contracting a vertex of degree* 2 *first compared to contracting a vertex of higher degree first. The arrow indicates the vertex to be contracted. Red lines indicate the added shortcuts. Contracting the vertex of degree* 2 *first yields* 1 *shortcut, contracting the vertex of higher degree* k *first yields* k *shortcuts, in this example* 3 *shortcuts.*

With all those considerations, we present a contraction order where vertices are contracted based on their highway score in ascending order, contracting vertices with degree ≤ 2 before any other vertex.

Definition 5.3. The highway-based contraction order for two vertices $v, w \in V$ is defined as

$$v \leq w \iff (deg(v) \leq 2 \wedge deg(w) > 2) \vee \left(\sum_{v \in e \in E} h_{HPR}(e) \leq \sum_{w \in e \in E} h_{HPR}(e) \right)$$

However, the definition of the highway-based contraction order can be interpreted in two ways. It can describe the degree of a vertex in the original graph before contraction, or the current degree in the contraction process. We call the two different approaches highway-based without and with updates respectively when we compare their results.

To get a better understanding how well our contraction order performs, we will add another baseline approach. We call it *Simple Ramps*. This contraction order contracts vertices with degree ≤ 2 before vertices of degree > 2. That means, it is a simplified version of definition of the highway-based contraction order. With this, we want to verify that the highway score of Definition 5.3 is needed to achieve good results and that the distinction between vertices with degree ≤ 2 and > 2 is not sufficient.

Definition 5.4. The Simple Ramps Contraction Order for two vertices $v, w \in V$ is defined as

$$v \leq w \iff deg(v) \leq 2 \vee deg(w) > 2$$

Out of simplicity, we only consider the updating variant of the Simple Ramps Contraction Order, which should generally perform better than the non-updating variant.

Finally, we want to compare our results with a random contraction order. With that, we want to show that both Edge Difference and the highway-based contraction order are good contraction orders. This comparison beyond Simple Ramps is important because the Simple Ramps order could just give extra-ordinarily bad results.

Results Figure 16 shows that both highway-based approaches show similar results as the original contraction order Edge Difference, except for Massachusetts. Note that a lower number of shortcuts is better because it reduces memory requirements, preprocessing time and query time for shortest paths queries afterwards. We can see that both baseline approaches Simple Ramps and the random contraction are significantly worse than Edge Difference and the highway-based contraction order. However, we only have results for Singapore and Thuringia. The other networks were too big for those two contraction strategies and had to be aborted after over 590 hours. The reason is that a bad contraction strategy creates many more shortcuts than there are original edges. This makes the network denser during the contraction process, which again means there are more neighbours for some vertices. With more neighbours, there are more pairs of vertices candidates for shortcuts, which can lead to even more shortcuts. Also, with more neighbours, more local queries have to be computed to determine which shortcuts have to be created and the local Dijkstra takes longer for

each of those candidate pairs. Therefore, we had to abort those computations and infer that those two strategies create many more shortcuts than the highway-based approaches and Edge Difference. With this, the purpose of this baseline comparison is still fulfilled, showing that the highway-based approaches perform significantly better than our baseline contraction orders. We demonstrate that the hierarchical highway structure is indeed the reason behind the good results of the highway-based approaches.

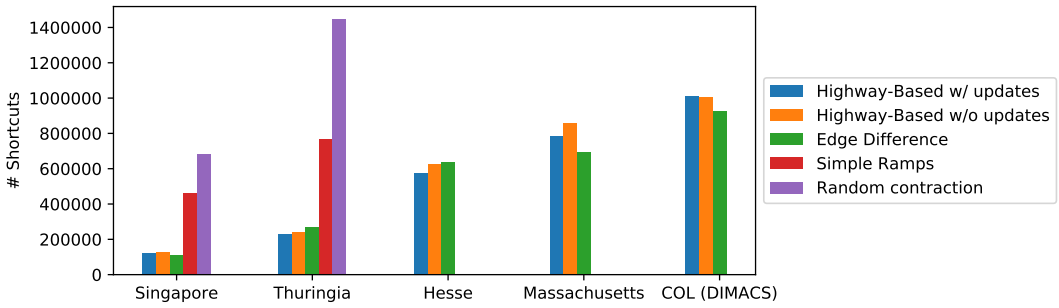


Figure 16: *Number of shortcuts created by the contraction process in all previously introduced networks, including the DIMACS Colorado network. The networks are ordered by their number of edges.*

The highway-based contraction order also has some computational advantages, as seen in Figure 17. The highway-based approaches do not need to compute local shortest paths in each contraction step. This also explains why the highway-based contraction orders are generally faster than Edge Difference, given h_{HPR} has already been computed. Only for Massachusetts, the runtime is similar or even slower for the updating approach. This can be explained by a significantly larger number of shortcuts produced by the highway-based approaches compared to Edge Difference.

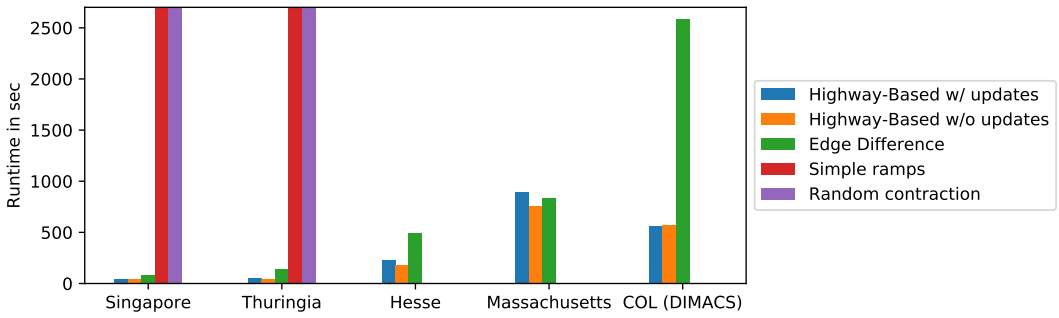


Figure 17: *The runtime of the contraction process in all previously introduced networks, including the DIMACS Colorado network. Simple ramps and random contraction are cut off because they are larger by 5 orders of magnitude. The networks are ordered by their number of edges.*

5.1.4 Comparison of Highway Networks with Random Networks

As a last step, we compare the distribution of h_{HPR} of real road networks with random networks with the same degree sequence and same edge weights. To generate those random network with the same degree sequence, we use the widely used *switching method* which takes the original network and keeps on switching heads and tails of randomly selected edges [25, 28, 32]. Each switch maintains the in-degree and out-degree of the edge's endpoints. Therefore, also the degree sequence is maintained. Such a null model has been used to find unique features and behaviours in various networks [25].

The original method takes two random edges (u, v) and (x, y), removes them from the graph and adds new edges (u, y) and (x, v). This procedure is repeated $k \times m$ times, where $k \in \mathbb{N}^{>0}$. We modify the switching method slightly to adapt it to our needs. First of all, we turn our originally mixed graph into a directed graph by replacing an undirected edge with two directed ones. Afterwards, we simplify the graph again. Instead of taking two edges at random, we iterate through all edges and perform a switch with each edge exactly once with a second edge which is picked randomly. Because we want a simple graph, we can neither allow multiple edges nor loops. Therefore, a switch is only performed if $u \neq y$ and $x \neq v$ and if no edge (u, y) or (x, v) already existing. The first criterium prevents loops, the second one multiple edges. Note that with selecting each edge as a candidate for switching and only the second edge by random, we only need one iteration to shuffle the graph. The edge (u, y) keeps the edge weight of the original edge (u, v), and the edge (x, v) keeps the edge weight of the original edge (x, y).

While the original switching method performs multiple iterations at random to achieve a high degree of randomness, the results of our method already demonstrate significant differences between the original graph and the randomly generated ones. We argue that by selecting the second edge at random, each edge has the chance of performing a random switch. Note that when the second edge selected is the same edge as the first one, or a loop or multiple edges would be created, a switch is not performed. In other words, it is really likely that a random switch is performed, with a small chance to stay in place. With that, a nearly randomly permutated graph is generated.

We do this comparison with all of our selected original road network. For each of those original networks, we generate 10 random counterparts with the same degree distributions and weights. Afterwards, we compare the value distributions of h_{HPR} between an original network and its 10 randomly generated ones.

Results A representative example for all networks is Thuringia, where we can see in Figure 18 that the distribution of h_{HPR} is highly zero-centred. In other words, most edges have a low h_{HPR} value, while only few have a high value. This is an indication that we might have a logarithmic distribution. That means, we compare

the distribution of the logarithm of the h_{HPR} value with each other.

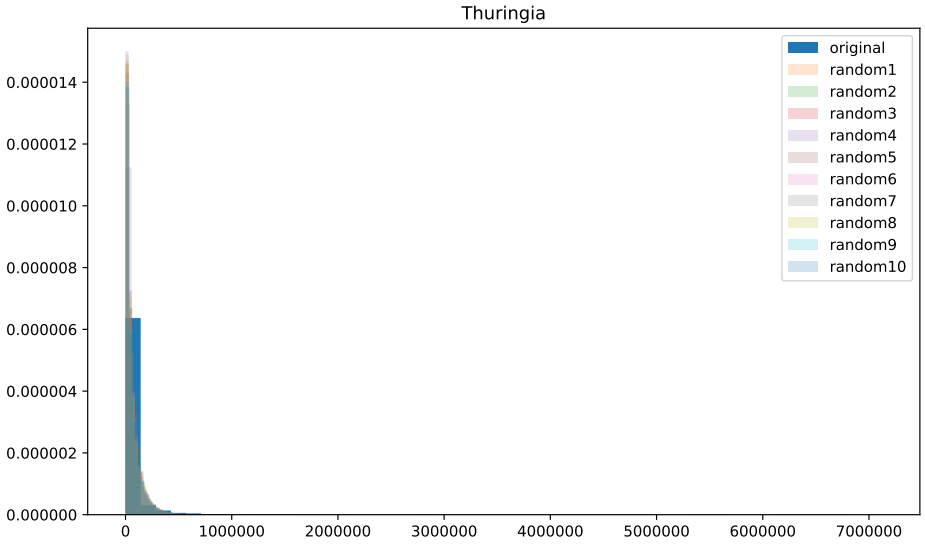
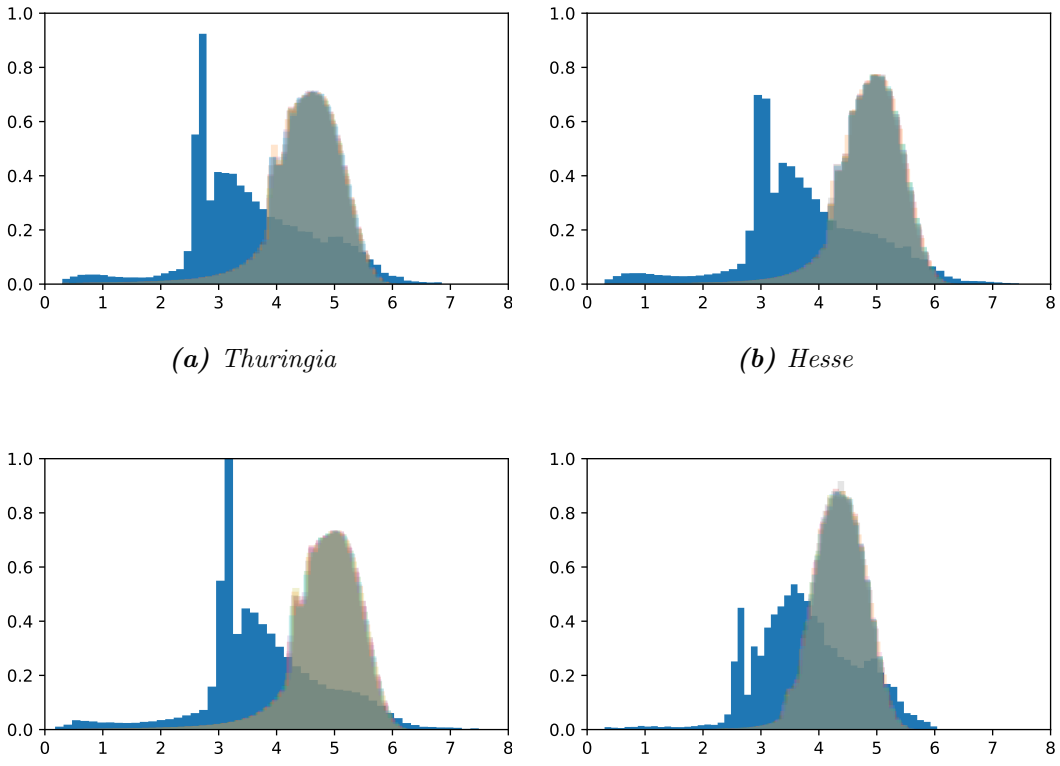


Figure 18: *h_{HPR} value distribution on Thuringia with 50 bins. Comparing the original Thuringia graph with 10 graphs generated from the original one with the described switching method. A linear value distribution is unsuitable for evaluating the h_{HPR} value distribution.*

The plots of the logarithmic distribution from the four networks Thuringia, Hesse, Massachusetts and Singapore can be seen in Figure 19. All four networks have remarkable differences between the original network and the random ones. The random ones seem to follow a nearly log-normal distribution. The normal distribution in the random networks gives the impression that there is no structure and hierarchy of importance. It appears to be random if a road is useful for shortest paths. Contrarily, the values of h_{HPR} in the real road network have a distribution that is completely different to the random ones. Interestingly, they seem to follow the same structure in all road networks. In the original networks, there are more edges with a low edge value, and fewer with high value, reaching higher values than the random networks.

We explain this distribution with an overall lower h_{HPR} value, which allows few edges to have a higher value than in the distributions of the random networks. The edges with low values are most likely road segments which do not function as part of many shortest paths. The sole purpose of these minor roads is to connect residences and other facilities built next to those roads with the remaining road network. As a result, we have many minor roads which quickly connect to roads of higher order, such as main roads. With that, it is overall less likely that shortest paths go through such minor roads, because a main road is usually nearby with higher speed limits and higher capacity. Those main roads can then be connected to even fewer high level



(a) *Thuringia* ***(b)*** *Hesse*

(c) *Massachusetts* ***(d)*** *Singapore*

Figure 19: $\log(h_{HPR} + 1)$ *value distribution of each road network with 50 bins. Comparing the original graph (blue) with 10 graphs generated ones (semi-translucent colours).*

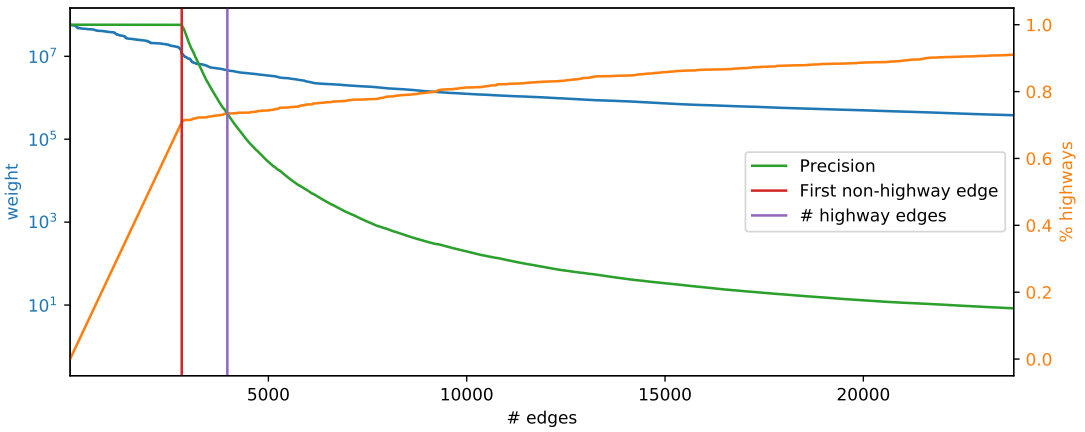


Figure 20: *Distribution of Highway-Paths Ratio and the highway edges (including ramps) in the network of Thuringia. The edges are sorted from highest to lowest h_{HPR} value. The plot is only a slice of the highest h_{HPR} values.*

roads, e.g. highways. This results in this large number of roads with a relatively low h_{HPR} value, with a decreasing number of increasingly important roads. In those original road networks, there is a highway core with high h_{HPR} values, unreachable by the random networks. We argue that those random networks cannot have edges with so many shortest paths as the lack of hierarchical structure does not channel the long-distance paths through those few edges. Note that most shortest paths in a road network are long-distance paths, which accounts for the high value of those highway edges.

Figure 20 backs our assumption that those edges with a very high h_{HPR} value are the most important for the highway network. The blue line with its scale on the left side represents the distribution of h_{HPR} on a logarithmic scale. The orange and green lines represent the fraction of highway edges and precision, with their scale on the right side. It measures the fraction of highway edges and precision respectively out of the x edges with the highest h_{HPR} value according to the x axis. The red horizontal line marks the first edge out of the top h_{HPR} edges which is no highway edge. The purple horizontal line marks the number of highway edges h. That means the intersection of the purple line with the precision marks the precision of $H_{HPR}(h)$. The plot indicates that most of the highway edges are selected among the edges with highest h_{HPR} value. Indicated by the red horizontal marker, almost 80% of highway edges can be selected from the edges with the highest h_{HPR} value without a single edge in between which is not a highway. From that point onwards, there are only few highway edges, distributed over the remaining edges of the network. It supports the idea that the edges with the highest h_{HPR} value are all highway edges. With a decreasing h_{HPR} value, the density of highway edges among all edges also decreases.

In our h_{HPR} value distribution from Figure 19, we further observe that there is a

gap after the first peak in all original road networks. The most plausible explanation is that the distinction between a minor road and a main road is significant. The number of shortest paths leading through them is so much higher compared to the minor roads that there is a big jump of magnitude between those two. Because of that, there are few roads in between the local and the more important inter-connecting roads.

Additionally, note that the described structure is also present in the network of the city state of Singapore. However, it is not as emphasised as in the other networks of territorial states. It is possible that highways in city states such as Singapore have a slightly different function, not attempting to connect remote places but mainly provide high-capacity roads within the city.

To finish our analysis with random networks, we wanted to compare how well the random networks can be contracted with our highway-based contraction order. For that, we took one of the random networks for each road network, computed the h_{HPR} values and started the contraction. However, only Singapore finished the contraction after 922,796 seconds with 1,396,465 shortcuts, over 10 times as many as in the original network. We decided to stop the computation after 1,700,000 seconds. At this time, the other bigger networks have not completed their contraction yet. This demonstrates that the structure of the network has a significant influence on the efficiency of the contraction. This supports the observation from the h_{HPR} distribution that there is a significant structural difference between the original and the random networks.

5.2 General Networks

In the previous section, we verified the definitions of highways in road networks by comparing them to the officially labelled highways. We found a unique distribution of the highway edge scores h_{HPR} compared to random networks. Furthermore, we introduced a contraction order based on Highway-Paths ratio for Contraction Hierarchies with a similar number of shortcuts as the original Edge Difference. This demonstrates that our definition of highways with h_{HPR} is able to describe an intrinsic network structure. In this particular case, it can be leveraged for fast route planning.

As we verified that this definition works well on road networks, we can use it as a foundation for a broader exploration in other types of networks. If other networks behave similarly to road networks, they might show similar structures.

We want to perform an exploration on other types of networks as an outlook of what could be done with our highway definition. Our verification by the comparison with random networks and the performance of Contraction Hierarchies can be used as a toolset for evaluating any type of network. We are open-minded about potential results, however our ultimate goal would be to detect highway-like structures in some of those networks. With this, we hope to be able to draw conclusions from the highway structure of other networks, e.g. regarding the efficiency of Contraction Hierarchies.

Our exploration of general networks specifically focuses on our mentioned toolset. We compare the h_{HPR} distributions within a network and its corresponding random networks, equally to the road networks. Furthermore, we use Contraction Hierarchies with both Edge Betweenness and our highway-based contraction and compare both runtime and number and shortcuts together with the distribution of h_{HPR} values. With this explorative analysis, we set the foundation and hope to motivate a deeper analysis of different types of networks, their structure and implications, e.g. on the efficiency of Contraction Hierarchies, based on their highway structure.

5.2.1 Evaluation setup

In general networks, we are faced with a number of challenges. First of all, as they are no road networks, we have no truth about whether an edge belongs to a highway. Because of that, we cannot evaluate them with precision or coverage. However, we can still generate random networks and compare the distributions of h_{HPR} between them and the original network. Furthermore, we can also contract those networks and compare the number of shortcuts between Edge Difference and our proposed contraction order from Section 5.1.3 using h_{HPR}.

Another problem is that those networks have a different structure. They might not necessarily be directed or weighted. For our experiments, we will treat undirected edges the same as in road networks. They can be traversed in either direction. Further, we treat unweighted networks as uniform networks, i.e. each edge has an edge weight of 1.

Equivalent to our analysis on road networks, we generate 10 random networks based on the original network with the switching method described in Section 5.1.4.

Network selection For our evaluation, we take various networks from the Stanford Network Analysis Project [24].

- ca-CondMat
  This graph models scientific collaboration on papers about Condensed Matter. An author is represented by a vertex. There is an undirected, unweighted edge if two authors co-authored at least one paper.
  $n = 23,133$, $m = 93,497$

- cit-HepPh
  This graph models citation relationships between papers in the field of high energy physics phenomenology. A paper is represented by a vertex. There is a directed, unweighted edge (u, v) if paper u cites paper v.
  $n = 34,546$, $m = 421,578$

- ego-Facebook
  This graph models friendships between users of Facebook. A user is represented by a vertex. There is an undirected, unweighted edge between two vertices if the users are friends.
  $n = 4039$, $m = 88,234$

- p2p-Gnutella31
  This graph models a snapshot of the structure of the peer-to-peer network Gnutella.
  A host in the network is represented by a vertex. There is a directed, unweighted edge if there is a connection between hosts. However, it is not disclosed what the direction models.
  $n = 62,586$, $m = 147,892$

- soc-Epinions1
  This graph models the wo-trusts-whom relationship between users on the review website Epinions. A user is represented by a vertex. There is a directed, unweighted edge (u, v) if user u trusts user v.
  $n = 75,879$, $m = 508,837$

Additional to the listed networks, we want to have a weighted network in our analysis. Therefore, we decided to take the ego-Facebook network and generate some weights based on triangle counts. Formally, an edge $e = \{u, v\}$ has the triangle count $t(e) = |\{w | w \in V \wedge \{u, w\} \in E \wedge \{v, w\} \in E\}|$. In other words, the triangle count of an edge is the number of common friends of the two endpoints. We assume a high number of common friends is allows for a higher connectivity and want those connections to be preferred. As we look for shortest parts, we have to reverse the weights. That means, as a weight on an edge we do not take the raw triangle count but determine $t_{max} = \max_{e \in E} t(e)$ and define $w(e) := t_{max} - t(e)$. We call the original unweighted network *uniform* and the network based on triangle counts *triangle*.

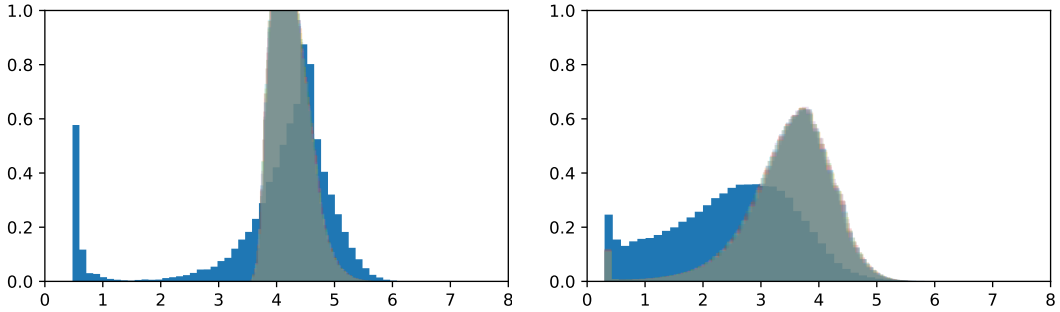
5.2.2 Results

The value distributions of the five different networks with their corresponding 10 random networks can be seen in Figure 21. The networks have different h_{HPR} distributions, none is a similar to the distribution of the road networks as the road networks are to each other. The network p2p-Gnutella31 is the most similar to the road networks in terms of overall shape.

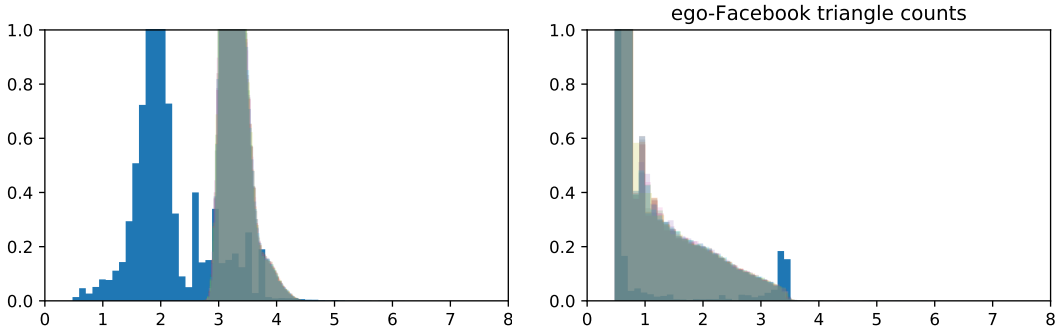
The random networks corresponding to ca-CondMat, cit-HepPh and ego-Facebook have a nearly log-normal distribution. Like in road networks, most h_{HPR} distributions of the original network are strongly dissimilar to their random ones. cit-HepPh is only slightly dissimilar and both networks p2p-Gnutella31 and soc-Epinions1 have almost no differences between the original network and their random counterparts. This is an interesting observation as this has not been seen in the road networks. The network p2p-Gnutella31 is particularly unique as the distribution of h_{HPR} values is quite similar between the original network and its random networks. While the random networks of road networks follow a nearly log-normal distribution, p2p-Gnutella31 maintains its overall distribution after shuffling. That means the p2p-Gnutella31 network has a property that is probably based on its weights or vertex degrees which allows the network to main its h_{HPR} distribution after shuffling. We explain this behaviour with what the network models. The connections in peer-to-peer networks tend follow a random pattern. The prioritisation of machines is mainly enforced by their interconnectivity. By swapping edges, the prioritisation of some vertices is not altered because the degree distribution is maintained. The network soc-Epinions1 also has a similar distribution of h_{HPR} values between the original network and its random equivalents. We observe that this who-trusts-whom network has a completely different structure than the ego-Facebook network, even though both model social networks. Part of the reason could be that Facebook friendships are undirected, while trust can be unilateral.

Furthermore, note the differences between the two Facebook networks in Figure 21c and Figure 21d. Both networks are based on the same vertices, the only difference is their edge weights. However, their h_{HPR} value distribution is significantly different on both networks, which applies to the original and the random networks. This makes sense, as the edge weights have a huge influence on the shortest in the network. The definition of h_{HPR} and highways overall is highly dependent on those shortest paths.

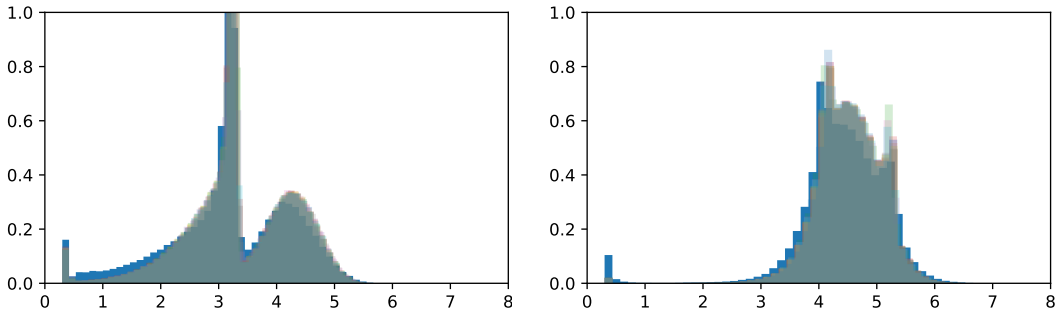
We take a look at the evaluation using Contraction Hierarchies of the selected general networks in Figure 22. We compare the number of shortcuts created by each approach in each network (denoted as SC) and the time needed to perform the entire contraction (denoted as *Contract. in sec*). With h_{HPR} *in sec*, we denote the additional time needed for the highway-based contraction order to compute the h_{HPR} values for all edges before the actual contraction. With $\lightning$, we indicate that the computation has not been finished, where the time $> t$ describes the time after which we decided to disregard the computation. We generally decided to disregard a computation if a long time has passed and it shows no promising progress from that point onwards.



(a) *ca-CondMat* ***(b)*** *cit-HepPh*



(c) *ego-Facebook (uniform)* ***(d)*** *ego-Facebook (triangle)*



(e) *p2p-Gnutella31* ***(f)*** *soc-Epinions1*

Figure 21: $\log(h_{HPR}+1)$ *value distribution of each general network with 50 bins. Comparing the original graph (blue) with 10 graphs generated ones (semi-translucent colours).*

		Edge Difference		Highway-Based Contraction		
Networks	m	SC	Contract. in sec	SC	Contract. in sec	h_{HPR} in sec
ca-CondMat	93,497	115,070	1,178,242	356,567	146,322	405
cit-HepPh	421,578	↯	>1,400,000	545,174	38,432	468
ego-Facebook (uni.)	88,234	17,657	181,750	38,460	6,588	137
ego-Facebook (tri.)	88,234	↯	>54,000	69	3,145	141
p2p-Gnutella31	147,892	↯	>1,400,000	633,102	164,123	225
soc-Epinions1	508,837	↯	>1,400,000	↯	>1,400,000	5881

Figure 22: *Comparison of Edge Difference and our highway-based contraction order on all introduced general networks.*

For some networks, Contraction Hierarchies seem to be challenging as Contraction Hierarchies is designed and suitable for road networks. We can observe this in the preprocessing of the network using Edge Difference. Except for ca-CondMat and ego-Facebook, none of the networks finished their contraction after a timeout of 1.4million seconds (>16 days). However, our highway-based contraction order using h_{HPR} manages to contract all except one graph in the given time. The contraction time of the highway-based approach is multiple orders of magnitude lower than that of Edge Betweenness. The comparingly low computational time for h_{HPR} is easily amortised by the much lower time of the highway-based contraction compared to the one of Edge Betweenness. For the two networks where we have a comparison with Edge Betweenness, the number of shortcuts for our highway-based contraction is higher with a factor of 2.

If we start with the assumption that the networks which have a similar looking structure to a road network are the ones which should be easiest to contract, p2p-Gnutella31 und ca-CondMat should be the two networks with the fastest and most-efficient contraction. However, we observe that the ratio of shortcuts to original edges is the lowest in both ego-Facebook networks (uniform, triangle). With uniform weights, we have a ratio of 0.43 with highway-based contraction and 0.20 with Edge Difference. With reverse triangle count weights, we have a ratio of $0.78 \cdot 10^{-3}$ with highway-based contraction. The network cit-Heph has the next-best contraction with a ratio of 1.29 with the highway-based contraction. We conclude that the characteristic similarity of the distribution to those of road networks alone does not give an indication of the efficiency of Contraction Hierarchies. Instead, the most similar network p2p-Gnutella31 has a shortcut to original edge ratio of 4.28 being out-performed by the other networks ca-condMat, cit-hepPh and ego-Facebook (both).

Nevertheless, what might play a role in the efficiency of Contraction Hierarchies, specifically for the highway-based contraction order, are highway-like structures in a network. We might be able to detect them by the dissimilarity of the original network to the random networks. The two networks with the most similar distributions

between the original and the random networks, p2p-Gnutella31 and soc-Epinions1, have the worst shortcut to edge count ratio. To observe the behaviour of the random networks more, we ran our highway-based contraction on one of the random networks of each of the networks. After over 1,900,000 seconds, only two networks managed to complete the contraction, ego-Facebook (only tested on uniform) and p2p-Gnutella31. The random network of ego-Facebook had two orders of magnitude higher runtime than the original network with 905,074 seconds and a worse shortcut to edge ratio of 4.54. The random network of p2p-Gnutella31 has a similar h_{HPR} distribution compared to the original network. The results vary less from its original network with 749,863 seconds and a shortcut to edge ratio of 6.00.

6 Conclusion

In our work, we took a look at the properties of real highways and how the term highway is used in network-related literature. We concluded that most definitions can be brewed down to road segments through which many shortest paths go, particularly long ones. We decided to take the edges with the highest Edge Betweenness value as a baseline approach and introduced a total of four definitions for highway networks. Under the assumption of unique shortest paths, we proved that our definitions can be rewritten such that we can compute a local measure for each edge independent of the choice of the highway network edge set. We further showed that this local measure can be computed with a modification of Brandes' algorithm. Afterwards, we could find the highway network edge set for each definition by selecting the edges with the highest value to maximise the highway network edge set criterion. With this, we introduced a fast way of receiving the highway network of our proposed definitions. To make this computation even faster, we described an approximation method which reduces the road network to a smaller skeleton network. This allowed us to run all the algorithms of our definitions directly on the reduced network, achieving similarly good results with a significantly lower computation time.

After we elaborated how we can compute the highway networks, we verified our definitions with experiments on four road networks, Thuringia, Hesse, Massachusetts and Singapore. To justify the term highway, we first wanted to evaluate how well our definitions manage to identify the real highway segments. For that, we used precision/recall and a measure we called coverage. In this analysis, we could see Highway-Paths Ratio as the clear winner. After some tweaking of the edge selection, we managed to correctly classify the highway segments in all road networks with over 50% correct labelling, mainly misclassifying highway segments close to the border of the network. With an extended network of 75 kilometres around Thuringia, we achieved a precision of 99% correctly classified highway segments.

Furthermore, we introduced two verification techniques independent of information about highway edges. Instead, they measure the overall structure detected in the network. We introduced a contraction order for Contraction Hierarchies, an efficient route planning algorithm. This contraction order is solely based on the local h_{HPR} values of each edge, therefore being called highway-based contraction order. We compared this contraction order with the original contraction Edge Difference, as well as random contraction and something we called Simple Ramps. In our experiments with real road networks, we managed to have a similar number of shortcuts as Edge Difference in the four road networks, both outperforming random contraction and Simple Ramps. While the computation of h_{HPR} takes additional time, the actual contraction of the highway-based contraction order is faster than Edge Difference in most networks. In contrast to our highway-based contraction order, the Edge Difference has to be recomputed in every contraction step.

As a second independent verification technique, we compared the h_{HPR} distribution in the original road networks with each 10 randomly generated networks. By comparing the distributions between the original and the random networks each, we

could see that the random networks follow a nearly log-normal h_{HPR} distribution, while the distributions in the original road networks have an emphasised structure similar to each other.

Our evaluation did not only demonstrate that our definition Highway-Paths Ratio is well-suitable for identifying real highways in road networks but for detecting an intrinsic structure of the network. This structure can be used for Contraction Hierarchies to do efficient route planning. With both our highway-based contraction order and the comparison of the h_{HPR} value distribution between original and random networks, we provided a toolkit for evaluating networks of various domains. As an introduction to highway network analysis in general networks, we chose 6 networks to compare the distributions of h_{HPR} and the performance of Contraction Hierarchies. We found out that there are different types of networks with an overall different value distribution. Some of the networks have a stable h_{HPR} distribution, even after shuffling their edges. Others, like road networks, have a clear difference between original and random networks. Further, we discovered that our highway-based contraction is faster than Edge Difference in all of the presented general networks. While in road networks highway-based contraction is outperformed by Edge Difference if we include the computation of h_{HPR}, highway-based contraction is significantly faster than Edge Difference in the selected general networks.

With these preliminary experiments on general networks, we want to motivate more experiments on highway network analysis on different kinds of networks. We want to find out if we if we can draw conclusions from the value distributions, e.g. if we can use them as markers for the efficiency of Contraction Hierarchies or other network properties. Furthermore, we are interested in seeing if based on the distribution and preprocessing time of Contraction Hierarchies, we manage to group different networks, e.g. highway-like networks and non-highway networks. The analysis of general networks should be expanded with a large set of other networks, specifically including more weighted networks. We particularly expect promising results in networks which model physically limited communications, as they have similar restrictions as road networks in terms of sparseness, capacity limits, etc. Such networks could include transport networks, communication networks or even networks of biological processes.

7 References

[1] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *Journal of the ACM*, 63(5):41:1–41:26, 2016. doi: 10.1145/2985473.

[2] Takuya Akiba, Yoichi Iwata, Kenichi Kawarabayashi, and Yuki Kawata. Fast shortest-path distance queries on road networks by pruned highway labeling. In *Proceedings of ALENEX*, pages 147–154, 2014. doi: 10.1137/1.9781611973198.14.

[3] Ziyad AlGhamdi, Fuad Jamour, Spiros Skiadopoulos, and Panos Kalnis. A benchmark for betweenness centrality approximation algorithms on large graphs. In *Proceedings of SSDBM*, pages 6:1–6:12, 2017. ISBN 978-1-4503-5282-6. doi: 10.1145/3085504.3085510.

[4] Jac M Anthonisse. The rush in a directed graph. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, 1971.

[5] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. *Route Planning in Transportation Networks*, pages 19–80. Springer, Cham, 2016. ISBN 978-3-319-49487-6. doi: 10.1007/978-3-319-49487-6_2.

[6] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566–566, 2007. ISSN 0036-8075. doi: 10.1126/science.1137521.

[7] Reinhard Bauer, Tobias Columbus, Ignaz Rutter, and Dorothea Wagner. Search-space size in contraction hierarchies. *Theoretical Computer Science*, 645:112–127, 2016. doi: 10.1016/j.tcs.2016.07.003.

[8] Alex Bavelas. A mathematical model for group structures. *Human organization*, 7(3):16, 1948.

[9] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001. doi: 10.1080/0022250X.2001.9990249.

[10] Ulrik Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, 2008. doi: 10.1016/j.socnet.2007.11.001.

[11] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007. doi: 10.1142/S0218127407018403.

[12] OpenStreetMap contributors. Region dumps retrieved from https://geofabrik.de, 2019. URL `https://download.geofabrik.de`. Accessed: 13th August, 2019.

[13] Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Customizable route planning. In *Experimental Algorithms*, pages 376–387. Springer, 2011. ISBN 978-3-642-20662-7. doi: 10.1007/978-3-642-20662-7_32.

[14] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0945-3245. doi: 10.1007/BF01386390.

[15] DIMACS. Implementation challenge - shortest paths - challenge benchmarks, 2006. URL `http://users.diag.uniroma1.it/challenge9/download.shtml`. Accessed: 12th September, 2019.

[16] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. ISSN 0004-5411. doi: 10.1145/28869.28874.

[17] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. ISSN 00380431. doi: 10.2307/3033543.

[18] Stefan Funke and Sabine Storandt. Provable efficiency of contraction hierarchies with randomized preprocessing. In *Algorithms and Computation*, pages 479–490. Springer, 2015. ISBN 978-3-662-48971-0. doi: 10.1007/978-3-662-48971-0_41.

[19] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004. ISSN 0196-6774. doi: 10.1016/j.jalgor.2004.05.002.

[20] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Experimental Algorithms*, pages 319–333. Springer, 2008. ISBN 978-3-540-68552-4. doi: 10.1007/978-3-540-68552-4_24.

[21] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. *Reach for A*: Shortest Path Algorithms with Preprocessing*, volume 74. American Mathematical Society, 2009. ISBN 978-0-8218-4383-3.

[22] David A Grossman and Ophir Frieder. *Information retrieval: Algorithms and heuristics*, volume 15. Springer, 2012. ISBN 978-1-4020-3005-5. doi: 10.1007/978-1-4020-3005-5.

[23] Luc Hogie. Grph - the high performance graph library for java, 2018. URL `http://www.i3s.unice.fr/~hogie/software`. Accessed: 13th August, 2019.

[24] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014. URL `http://snap.stanford.edu/data`. Accessed: 28th August, 2019.

[25] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002. ISSN 0036-8075. doi: 10.1126/science.298.5594.824.

[26] Carsten Möller. OSM2PO, 2017. URL http://osm2po.de. Accessed: 13th August, 2019.

[27] Jürgen Pfeffer and Kathleen M. Carley. k-centralities: Local approximations of global measures based on shortest paths. In *Proceedings of WWW*, pages 1043–1050, 2012. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188239.

[28] A. Ramachandra Rao, Rabindranath Jana, and Suraj Bandyopadhyay. A markov chain monte carlo method for generating random (0, 1)-matrices with given marginals. *Sankhya: The Indian Journal of Statistics*, 58(2):225–242, 1996. ISSN 0581572X. URL http://www.jstor.org/stable/25051102.

[29] Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, Mar 2016. ISSN 1573-756X. doi: 10.1007/s10618-015-0423-0.

[30] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In *Algorithms*, pages 568–579. Springer, 2005. ISBN 978-3-540-31951-1. doi: 10.1007/11561071_51.

[31] Dominik Schultes and Peter Sanders. Dynamic highway-node routing. In *Experimental Algorithms*, pages 66–79. Springer, 2007. ISBN 978-3-540-72845-0. doi: 10.1007/978-3-540-72845-0_6.

[32] Miguel E. P. Silva, Pedro Paredes, and Pedro Ribeiro. Network motifs detection using random networks with prescribed subgraph frequencies. In *Complex Networks VIII*, pages 17–29. Springer, 2017. ISBN 978-3-319-54241-6. doi: 10.1007/978-3-319-54241-6_2.

[33] OpenStreetMap Wiki. Key: Highway, 2019. URL https://wiki.openstreetmap.org/w/index.php?title=Key:highway&oldid=1787212. Accessed: 13th August, 2019.

[34] OpenStreetMap Wiki. Motorway, 2019. URL https://wiki.openstreetmap.org/w/index.php?title=Tag:highway%3Dmotorway&oldid=1875481. Accessed: 26th July, 2019.