Scalability of Route Planning Techniques

Johannes Blum, Sabine Storandt
Julius-Maximilians-Universität Würzburg
97072 Würzburg, Germany
{blum,storandt}@informatik.uni-wuerzburg.de

Abstract

In this paper, we thoroughly analyze the scaling behavior of several state-of-the-art route planning techniques for road networks, all of which rely on preprocessing. One goal is to determine which technique is most suitable to be used on huge networks. To be able to conduct scalability studies in a clean way, we first describe a new kind of road network generator that allows to produce road networks even larger than that of our planet with similar properties as real networks. We then carefully implement several preprocessing-based route planning techniques, as contraction hierarchies, hub labels and transit nodes, to study their space consumption as well as their search spaces in different sized networks. This allows to derive functions that describe their empirical scaling behavior for the first time. We also compare our functions to existing theoretical bounds. We show that several of our results can not be sufficiently explained by the theoretical investigations conducted so far. Hence our results encourage a further look for road network models that allow for better predictions.

Introduction

In the last decade, several new acceleration techniques for shortest path planning in road networks have been proposed. The most successful ones rely on preprocessing. Here, in an initial phase, auxiliary data is created which later on can be used to reduce the search spaces of planning techniques without compromising the optimality of the found route.

Incarnations of this scheme are, for example, contraction hierarchies [CH] (Geisberger et al. 2012), hub labels [HL] (Abraham et al. 2011b) and transit nodes [TN] (Bast et al. 2007). All of these techniques are used as building blocks in plenty of applications and research projects, for route planning and location-based services as well as in navigation systems. Indeed, all these techniques have their merits as they offer different trade-offs between space consumption and query time as shown in (Bast et al. 2016): On the Western Europe network (18 million nodes, 42.5 million directed edges), the space consumption of CH is about 0.4 GB; for TN it's 6 times higher, for HL 47 times higher. Regarding query times, HL needs on average 0.56 μs to compute the shortest path; query times for TN are higher by a factor of 4, and by a factor of 196 for CH. The explicit numbers are listed in Table 1. So the empirical observations indicate that wrt query time, we have CH $>$ TN $>$ HL and wrt space consumption we have CH $<$ TN $<$ HL. A natural question is, whether these relations are generally true (invariant of the scale of the network) – or, if there is some network size n for which the current observations no longer hold. Of course, comparisons on a single instance can not provide conclusive results here.

Theoretical investigations were conducted as an attempt to capture and predict the scaling behavior of shortest path planning techniques. For example, a parametrized analysis depending on a graph parameter called highway dimension h was performed in (Abraham et al. 2010). The analysis, however, reveals similar space consumption values and search space sizes for all three, CH, HL and TN. So far, it remains unclear if this difference between theory and practice can be explained by asymptotic bounds versus finite networks, or if the notion of highway dimension is not fine-grained enough to differentiate between the scaling behavior of these techniques sufficiently. The goal of this paper is to shed light on these questions.

Related Work

The empirical justification of route planning techniques usually stems from the investigation of country or continental sized networks. Typical benchmarks besides the Western Europe graph are the U.S. road network (TIGER data, 24 million nodes, 58 million edges) or networks extracted from OpenStreetMap as Germany (20 million nodes, 42 million edges) or Europe (175 million nodes, 348 million edges), see e.g. (Bast et al. 2007; Delling et al. 2011; Abraham et al. 2011b; Geisberger et al. 2012; Bast et al. 2016). However, none of these papers features a scalability study conducted on different sized networks. Therefore, the only functions that describe the behavior of route planning techniques in dependency of the network size n are based on theoretical investigations.

Such investigations rely on certain assumptions about the structure of the road network. Besides the highway dimension h (Abraham et al. 2010), also the so called skeleton dimension k (Kosowski and Viennot 2017) and the tree width t (Bauer et al. 2013) could be instrumented as parameters to derive theoretical search space size and space consump-

	CH		TN		HL	
	query	space	query	space	query	space
graph	practical results					
Western Europe	110 μs	0.4 GiB	2.09 μs	2.5 GiB	0.56 μs	18.8 GiB
model	theoretical results					
highway dimension h	$\mathcal{O}(h \log D)$	$\mathcal{O}(nh \log D)$	$\mathcal{O}(h^2)$	$\mathcal{O}(hn + m)$	$\mathcal{O}(h \log D)$	$\mathcal{O}(nh \log D)$
skeleton dimension k	-	-	-	-	$\mathcal{O}(k \log D)$	$\mathcal{O}(nk \log D)$
tree width t	$\mathcal{O}(t \log n)$	$\mathcal{O}(nt \log n)$	-	-	-	-
minor closed	$\mathcal{O}(\sqrt{n})$	$\mathcal{O}(n \log n)$	-	-	-	-
bounded growth	$\mathcal{O}(\sqrt{n} \log n)$	$\mathcal{O}(n \log D)$	$\mathcal{O}(\sqrt{n} \log^4 n)$	$\mathcal{O}(n\sqrt{n} \log n)$	$\mathcal{O}(\sqrt{n})$	$\mathcal{O}(n\sqrt{n})$

Table 1: Overview of practical and theoretical results for different shortest path speed-up techniques, with n, m and $D \leq n$ denoting the number of nodes, the number of edges, and the diameter of the road network.

tions bounds. These bounds are listed in Table 1. However, the values for h, k and t are unknown for any large road network, which makes it difficult to judge the quality of the obtained bounds. Other models used are minor-closed graphs with $\mathcal{O}(\sqrt{n})$ separators (Bauer et al. 2013) and bounded growth of the graph metric (Blum, Funke, and Storandt 2018), see also Table 1.

Contribution

In the first part of the paper, we present a new road network generator which cuts tiles from real networks and fits them together in order to produce arbitrary large road networks of compact shape. The resulting networks are the basis for our scalability studies. We then discuss different preprocessing schemes for CH, TN and HL and provide implementation details of the methods used in our study. Afterwards we conduct experiments on (generated) road networks with the number of nodes ranging from 2,000 to almost a billion. We use curve fitting methods to turn these experimental results into functions that describe the scaling behavior of the considered techniques. We then compare our functions to the functions derived from theoretical investigations. This allows to judge the quality of the used road network models. The main insights gained by our study are:

- CH and HL are both useful in huge networks. CH exhibits the lowest space consumption among the three considered techniques and HL the smallest query times independent of the scale of the network. Surprisingly, TN is dominated by HL in huge networks, as it has a higher space consumption as well as higher query times. Hence TN should not be applied to huge networks.
- The road network model based on the highway dimension h is not suitable to explain the different scaling behavior of CH and HL that was observed in our experiments. Furthermore, the bounds for space consumption obtained from the highway dimension h or the tree width t are too loose for CH.
- Theoretical bounds based on minor closed graphs, graphs with bounded growth or graphs with skeleton dimension k (partially) match our empirical results pretty well. Still, we observe some gaps between theory and practice which should be investigated in future work.

A New Paradigm for Road Network Generation

To enable sound scalability studies, we need access to road networks of different size. Usually, cut outs of real road networks are used for this purpose. But we argue, that this is not sufficient here. First of all, the planet road network as available in OSM[1] contains only about 600 million nodes, and these nodes are distributed over many unconnected or only sparsely connected components. Secondly, due to the silhouette of real road networks, one runs quickly into border affects – which leads to an artificial reduction of search space sizes of route planning techniques. To avoid such distortions and to be able to consider even larger networks than those available at the moment, we want to generate road networks of compact shape and arbitrary size.

Existing Road Network Generators

Simple models for generating road networks are e.g. grid graphs or unit disk graphs. But they lack the hierarchy of slow and fast roads usually present in large road networks. In (Eppstein and Goodrich 2008), road networks were characterized as multi-scale dispersed graphs and modeled as subgraphs of disk intersection graphs. It was proven that in such non-planar geometric graphs, shortest paths and Voronoi diagrams can be computed in linear expected time. In (Eisenstat 2011), nested quad trees were proposed to model road networks but with the primary goal of analyzing maximum flows. The graph generator presented in (Abraham et al. 2010) is custom-tailored to produce networks with constant highway dimension. There, the road network is constructed in an online fashion, always connecting a new node to the so far existing network following a specific protocol. To model the hierarchy, a speedup parameter is defined that makes travel times on longer edges proportionally quicker.

In (Bauer et al. 2010), most of these generators were implemented and tested against a newly designed generator based on recursive Voronoi cell computations. Properties as node degree distribution, distance distributions and speedups for selected route planning techniques were measured with the help of a small scalability study (up to half a million nodes). It was experimentally shown that the sophisticated generators perform well for most considered aspects.

[1] openstreetmap.org

But the highway dimension related generator produced too dense networks and the scalability studies revealed too large speed-ups for CH when using the Voronoi-based generator. Furthermore, most of these generators require to fix a large number of parameters manually. We will follow a completely different approach, where we do not generate road networks from scratch. Instead, we bootstrap our generator with tiles cut from real road networks and recombine them to derive road networks larger than that of our planet – with a compact shape, and with similar properties as the original networks.

Road Networks as Jigsaw Puzzles

Our network generator requires a road network as input where for every node its coordinates and for every edge the corresponding road type (e.g. motorway or living street) and traversal speed (or distance) are given. All of this information is available for road networks extracted from OSM. We distinguish between critical and non-critical road types, where critical roads such as motorways are the most important and fastest ones in the network.

From the input network we cut out square shaped tiles at random positions and combine them on a grid that is filled bottom-up from left to right. In every tile we insert *portals* at all positions where an edge was cut by the tile boundary, these portals are the places where we connect the whole network by inserting edges between neighboring tiles on the grid. To smooth the transitions between tiles, they are not placed seamlessly next to each other, but with a small gap (see Figure 1). To create sensible networks, we place only similar tiles next to each other. For that, we keep a supply of already cut out tiles. When we place a new tile into the grid, we consider the portals along the neighboring tiles that have already been placed and select a tile from the supply according to the following criteria. First of all, we want every portal corresponding to an edge of a critical road type to be connected to a portal of the same road type. The remaining portals are connected whenever possible according to their relative positions (avoiding crossings). To quantify the similarity of two tiles, we introduce a cost for connecting them: Connecting two portals of the same type comes with no cost, connecting two portals of different types and turning a portal into a dead-end has cost 1. The total cost of connecting two tiles can be computed by means of dynamic programming as in the common algorithm for the computation of the Levenshtein distance of two strings. So when a new tile is to be placed into the grid, we choose that tile from our supply that exhibits the least total cost to its neighbors. If necessary, we break ties by the total offset along the boundary (in terms of euclidean distance) that all portals to be connected exhibit.

Reasonable road types and traversal speed of the inserted inter-tile edges are derived from the portals that they connect. Previous road generators did not produce road types, and travel times were derived from very simple models. With our approach we can interfere sensible values for all edges. Moreover, only two parameters need to be set for our generator, namely the tile size and the size of the network to generate.

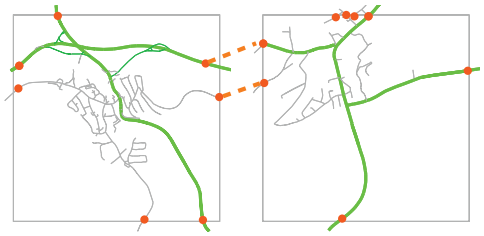

Figure 1: Two neighboring tiles that have been connected via some of their portals. Green roads are critical.

Performance of Route Planning Techniques

To conduct scalability studies, and to make fair comparisons, we need to implement the preprocessing-based route planning techniques carefully and then measure their space consumption and their search space sizes on a variety of input networks. But for each of the techniques (CH, TN, HL), different preprocessing schemes were proposed in previous work. As different preprocessing schemes might also result in different scaling behavior, we ideally would implement and evaluate all of them. Unfortunately, the preprocessing schemes used in the highway dimension dependent theoretical analysis require the extraction and storage of all shortest paths in the network, and the computation of hitting sets on these shortest paths. This is totally impractical in larger networks. While exact hitting set computation can be replaced by an approximate solution (at the cost of a factor of $\log n$ in the space consumption and the search space sizes), even the extraction of all shortest paths requires superquadratic time and uses quadratic space in n (Abraham et al. 2013).

Hence, in the following, we focus on preprocessing schemes which are applicable to real-world road networks of continental size and larger.

Contraction Hierarchies

The original CH construction scheme (Geisberger et al. 2012) relies on node contraction: A node v is deleted from the graph, and shortcut edges are inserted between neighbors of v if they are necessary to preserve the pairwise shortest path distances. The preprocessing phase of CH consists of contracting all nodes in some order. In the end, a new graph $G^+(V, E \cup E^+)$ is constructed, with E^+ being the set of shortcut edges that were inserted during the contraction process. So the space consumption of CH is determined by $|E^+|$. Let $rank(v)$ be the rank of a node in the contraction order. An edge (v, w) is called upward if $rank(v) < rank(w)$. The upward graph $G^\uparrow(s)$ of a node s is formed by all paths consisting of only upward edges emerging from s in G^+. In an undirected graph, an s-t-query is answered via a bidirectional Dijkstra run on $G^\uparrow(s)$ and $G^\uparrow(t)$, respectively. Both runs settle the node that was contracted last on the original shortest path from s to t in G. Hence identifying p such that $d_s(p) + d_t(p)$ is minimized leads to correct query answering. Any contraction order is valid, but the space consumption and the search space sizes depend on the chosen permutation. In practice, always contracting the node next that minimizes the graph size after the

contraction works well; more complicated selection functions were also tested.

A treewidth t based analysis of CH (Bauer et al. 2013) inspired a different CH construction scheme based on nested dissections, which also performs very well in practice (Dibbelt, Strasser, and Wagner 2014). There, the contraction order is determined top-down by identifying a small balanced cut in the network, let the cut vertices be the ones contracted last and then recurse on the two network parts.

Transit Nodes

TN differs from other speedup techniques as it guarantees correct shortest path distances only for 'long' queries where the source and the target are sufficiently far from each other. For 'short' queries a fall-back algorithm has to be used, e.g. Dijkstra. The idea behind TN is that all shortest paths from a node v to all 'far away' destinations pass through some small set of so-called access nodes $AN(v)$ (e.g. slip roads of nearby interstates). The union of all access nodes forms the transit node set T. For every pair of transit nodes, the shortest path distance is precomputed and stored in a look-up table. In addition, every node stores the distances to all its access nodes. So the total space consumption can be expressed as $|T|^2 + \sum_{v \in V} |AN(v)|$. A 'long' s-t-query reduces to check all access node distances of s and t and the respective distances between them in the look-up table, all of which is precomputed. Hence the query time is in $|AN(s)| + |AN(s)| \cdot |AN(t)| + |AN(t)|$.

In practice, the most efficient method for TN computation is based on CH (Arz, Luxen, and Sanders 2013). Here, the c nodes with the highest $rank$ in the contraction order are declared transit nodes. Access node computation then also makes use of the CH graph.

Hub Labels

In the HL approach, every node v gets assigned a set of labels $L(v)$. A label is a node w, together with the distance $d_v(w)$. The goal is to find concise label sets which fulfill the *cover property*, that is, for every $s, t \in V$ the label set intersection $L(s) \cap L(t)$ contains a node w on the shortest path from s to t. If this is the case, queries can be answered by simply summing up $d_s(w) + d_t(w)$ for all $w \in L(s) \cap L(t)$ and keeping track of the minimum. Computing $L(s) \cap L(t)$ can be done by a merging-like step assuming the label sets are presorted by node IDs. Hence the query time is in $\mathcal{O}(|L(s)| + |L(t)|)$, while the space consumption is in $\mathcal{O}(\sum_{v \in V} |L(v)|)$.

CH-based Hub Labels In (Abraham et al. 2011a), it was observed that hub labels can also be computed based on CH. More precisely, choosing the nodes in $G^{\uparrow}(v)$ as $L(v)$ for each $v \in V$ leads to a correct HL data structure. A label w of v is however useless if its shortest path distance from v in $G^{\uparrow}(v)$ exceeds the one in G. By performing a one-to-all shortest path computation from v in both $G^{\uparrow}(v)$ and G, such labels can be identified and subsequently pruned, which reduces the label sizes further in practice.

Skeleton-based Hub Labels In (Kosowski and Viennot 2017), a randomized preprocessing algorithm for HL was introduced, which in expectation uses $\mathcal{O}(k \log D)$ labels per node and thus has a total space consumption of $\mathcal{O}(nk \log D)$. It assigns a random value to every edge and chooses for every pair of nodes (v, u) a hub $\eta_v(u)$ as the edge of minimal random value among the middle $d_v(u)/6$ edges of the shortest v-u-path. The label of a node v is then constructed as the set of all hubs $\eta_v(u)$ – note that this approach uses edges as labels instead of nodes.

To compute the label set of a node v, the authors of (Kosowski and Viennot 2017) propose to iterate over the shortest path tree of v by non-decreasing distance, while keeping a sliding window for the middle $r/6$ edges of every active branch at distance r. In every step, the window belonging to the current branch is moved away from v and the minimum edge is chosen. By storing every sliding window in a balanced binary search tree, this operations can be performed in $\mathcal{O}(\log r)$ time on a window of size $r/6$. Note however that a window needs to be split when it is moved over a node where the shortest path tree branches. If this is handled by cloning the whole window in $\Omega(r)$ time, the algorithm might take $\Omega(Dn)$ time where D denotes the diameter.

Therefore we suggest to traverse the shortest path tree in a depth-first search instead. Again we keep a sliding window for the current branch, which moves forwards and backwards when descending and ascending, respectively. During this process, only one sliding window is required that can be accessed and modified in $\mathcal{O}(\log n)$ time, so the total runtime is $\mathcal{O}(n \log n)$.

Experiments

We implemented the road network generator in C++, as well as the route planning techniques CH, TN and HL using a node-based graph representation and the following preprocessing and query answering schemes. For CH we use the greedy preprocessing scheme described in (Geisberger et al. 2012) that always contracts a node minimizing the number of shortcuts to be inserted. For TN we select the $5\sqrt{n}$ nodes of highest rank in the contraction order as transit nodes. For HL we use both the CH-based approach from (Abraham et al. 2011a) with subsequent pruning and the skeleton-based approach from (Kosowski and Viennot 2017).

Experiments were conducted on a AMD Opteron 6272 CPU (32 cores clocked at 2.1 GHz) with 264 GB main memory, running Ubuntu 16.04.2 (kernel 4.4.0). We used the GNU C++ compiler 5.4.0 with optimization level 3. All experiments were based on the OSM road network of Germany (in the following just "Germany") consisting of about 23.9 million nodes and 24.6 million undirected edges that are organized in 15 road categories. Shortest paths were computed wrt travel time.

Validation of the Road Network Generator

To evaluate our generator we created different sized networks and also varied the tile size. For determining a good tile size, we first created 10 different networks of dimension

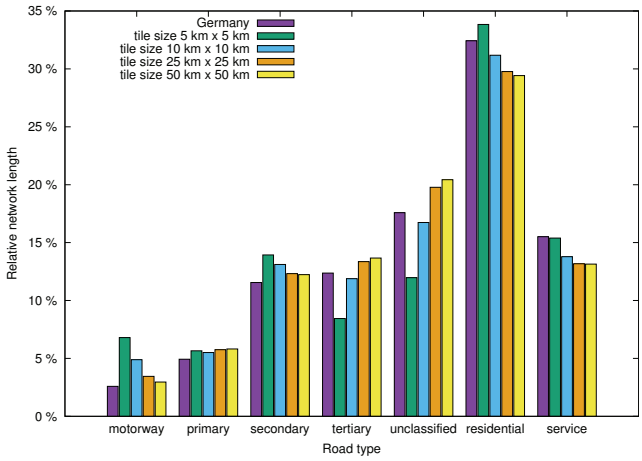

Figure 2: Relative network length of road types.

	degree					
network	0	1	2	3	4	≥ 5
Germany	0.0%	6.5%	75.5%	15.6%	2.2%	0.3%
Generated	0.0%	6.2%	76.1%	15.3%	2.2%	0.3%

Table 2: Relative frequency of node degrees in a network generated with a tile size of 25 km $\times$ 25 km

1000 km$\times$1000 km for each tile size of 5 km$\times$5 km, 10 km$\times$ 10 km, 25 km $\times$ 25 km and 50 km $\times$ 50 km. It turned out that tiles with a size about 25km $\times$ 25 km produced the best results. Larger tiles were more difficult to fit together and smaller tiles did not preserve the road network structure well. With 25km $\times$ 25 km tiles, important aspects of the original road network, as e.g. the distributions of node degrees (see Table 2) and road types (see Figure 2 for a comparison between different tile sizes) are well preserved.

The largest network we generated contains about 778 million nodes and 834 million edges. Its construction required 110 minutes and reached a peak memory consumption of 7.5 GB. This is clearly more efficient than both sophisticated approaches evaluated in (Bauer et al. 2010), where the authors reported a runtime of 150 minutes and 63 minutes and a peak memory consumption of 21 GB and 8.4 GB for the generation of networks with less than 50 million nodes using the Voronoi-based generator and the approach proposed in (Abraham et al. 2010), respectively.

An example of a network generated with our approach can be found in Figure 3.

Performance Study

To study the performance of route planning techniques on networks of different sizes, we used our largest generated network with roughly 800 million nodes and recursively cut out squares, reducing the number of nodes in every step by a factor of four until we stopped at about 2 000 nodes. To show that the results are truthful (and to further validate our generator), we also include results on cutouts of Germany. There, the largest square we could cut contained about 11 million nodes. For networks with up to $7 \cdot 10^5$ nodes, we report the average over 4 instances (for both Germany and the generated network), to avoid strong distortions by 'unlucky' selection.

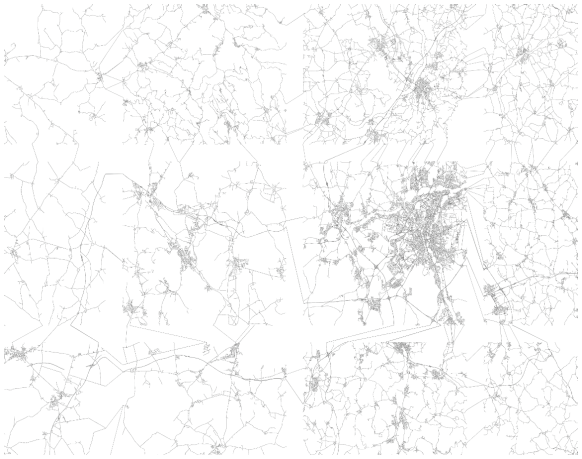

Figure 3: Cut-out of a road network generated with our approach.

On the resulting networks we measured important performance aspects of CH, TN and HL. In particular, for CH, we provide the ratio of the number of inserted shortcuts and the number of original edges $|E^+|/|E|$, as well as the average number of nodes $|V^\uparrow|$ and edges $|E^\uparrow|$ in the search space. For TN, we measured the average number of access nodes $|AN|$, and the radius r that determines the minimum distance between the source and target vertex such that the TN approach will return the shortest path distance for sure. For HL, we measure the average label size $|HL|$ per node subdivided by the used preprocessing technique (CH-based or skeleton-based). All results are collected in Table 3.

We observe that the results on the generated instances are very close to the ones obtained on Germany. In particular, there is no overestimation on larger instances and the results differ by less than a factor of 2.5. Hence we deem the generated networks suitable as basis for our scalability analysis.

Curve Fitting

To evaluate the scalability of the route planning techniques, we tried to fit the data points obtained from our performance study to $f(x) = a \cdot \log_2(x)^b$ (polylogarithmic growth) and $g(x) = c \cdot x^d$ (polynomial growth)[2]. These functions cover the bounds from the theoretical investigations. For example, if h, t or k would be constant, then search spaces for CH and HL would be in $\mathcal{O}(\log n)$ and the space consumption in $\mathcal{O}(n \log n)$. If, on the other hand $h, k, t \in \mathcal{O}(\sqrt{n})$ (which is their known value in uniformly weighted grid graphs), then search spaces would be in $\mathcal{O}(\sqrt{n} \log n)$ and the space consumption in $\mathcal{O}(n\sqrt{n} \log n)$.

We use the gnuplot implementation of the Marquardt-

[2]We also tried more complicated models with additional terms but obtained very similar results.

	Germany							Generated						
	CH			TN		HL		CH			TN		HL	
n	$\frac{\lvert E^+\rvert}{\lvert E\rvert}$	$\lvert V^\uparrow\rvert$	$\lvert E^\uparrow\rvert$	$\lvert AN\rvert$	r	$\lvert HL_1\rvert$	$\lvert HL_2\rvert$	$\frac{\lvert E^+\rvert}{\lvert E\rvert}$	$\lvert V^\uparrow\rvert$	$\lvert E^\uparrow\rvert$	$\lvert AN\rvert$	r	$\lvert HL_1\rvert$	$\lvert HL_2\rvert$
2×10^3	0.76	14	26	2	18	12	100	0.74	15	28	2	19	13	105
1×10^4	0.79	26	57	2	37	18	167	0.78	23	59	2	42	17	163
4×10^4	0.81	43	154	3	75	26	265	0.79	41	147	3	81	26	257
2×10^5	0.82	82	479	4	135	40	387	0.79	74	379	4	158	37	349
7×10^5	0.83	175	1521	5	271	58	514	0.79	149	1106	5	333	54	482
2×10^6	0.83	393	5032	8	582	90	675	0.80	300	3010	8	536	76	630
1×10^7	0.82	929	15611	20	1054	121	904	0.80	562	7335	12	1081	103	780
5×10^7	-	-	-	-	-	-	-	0.79	1185	21300	21	2253	149	953
2×10^8	-	-	-	-	-	-	-	0.79	2547	63223	45	4057	213	1243
8×10^8	-	-	-	-	-	-	-	0.79	5188	160193	88	7459	295	1405

Table 3: Average search space sizes (over 1,000 random queries) and space consumption for different route planning techniques: ratio $|E^+|/|E|$, number of nodes in upward graph ($|V^\uparrow|$), number of edges in upward graph ($|E^\uparrow|$), number of access nodes ($|AN|$), radius to furthest access node in seconds (r), number of hub labels for CH-based preprocessing ($|HL_1|$) and skeleton-based preprocessing ($|HL_2|$)

Levenberg algorithm[3] for fitting. The algorithms works in an iterative fashion, always judging the current choice of parameters by the sum of squared residuals between the input data point and the function values, and then modifying the parameters slightly in a direction that might improve the fitting quality. After the procedure converged, the following quality measures are provided:

- *Final sum of squares of residuals (SSR)*, that is, the remaining squared gaps between the data points and the fitted function (the smaller the SSR the better the fitting quality).
- *Asymptotic standard errors (ASE) for each parameter* which are approximations to the standard deviations that can be derived directly from the variance-covariance matrix. Intuitively, the larger the standard deviation the larger the possible fluctuations of the parameter around the outputted mean. The ASE is prone to be overly optimistic but still provides a sound way to compare different parameter choices (the smaller the ASE the better the confidence in the parameter choice).

We also attempt to simplify the fitting results. For example, if a parameter is returned to be 0.489666754 we rerun the fitting procedure after fixing the parameter to be 0.5, and compare the quality of the two runs. If the differences turns out to be negligible (that is, the SSR value got worse by at most a factor of 4), we will also report the results for the simpler (rounded) parameter. We deem this to be a natural approach, as the experiments to compute the input data points as well as the fitting procedure might induce small fluctuations anyway, and the ASE implies that there is a (small) range of parameter choices and not only a single value that works.

Fitting Results

For every column in Table 3, we performed curve fitting to the polylogarithmic function $f(x)$ and the polynomial function $g(x)$. In the following, we discuss the results for each of the route planning techniques individually.

[3] http://gnuplot.sourceforge.net/docs_4.2/node82.html

CH					
shortcut to original edge ratio $\lvert E^+\rvert/\lvert E\rvert$					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	0.669	0.022	2.3%	33.6%	0.001
refit b=0	0.785		0.3%		0.002
g(n)	c	d	ASE(c)	ASE(d)	SSR
	0.748	0.001	0.9%	42.5%	0.002
refit d=0	0.785		0.3%		0.002
number of nodes in search space $\lvert V^\uparrow\rvert$					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	$\cdot 10^{-11}$	9.770	59.5%	1.8%	18255
g(n)	c	d	ASE(c)	ASE(d)	SSR
	0.134	0.516	7.2%	0.7%	2847
refit d=0.5	0.180		0.6%		10245
number of edges in search space $\lvert E^\uparrow\rvert$					
f($\lvert V^\uparrow\rvert$)	a	b	ASE(a)	ASE(b)	SSR
	10^{-7}	10.565	19.0%	0.7%	$2.2 \cdot 10^6$
g($\lvert V^\uparrow\rvert$)	c	d	ASE(c)	ASE(d)	SSR
	1.828	1.331	8.9%	0.8%	$2.8 \cdot 10^6$

Table 4: Fitting results for CH.

Contraction Hierarchies The fitting results for CH are summarized in Table 4. For the ratio of shortcut edges and original edges, we already observe by a glance on Table 3 that the value does not seem to depend on the number of nodes n in the network, but rather appears to be a global constant. This is reflected in our fitting experiments: While the initial curve fitting resulted in slightly positive exponents b and d when using $f(n)$ or $g(n)$, respectively, refitting after fixing $b = 0$ or $d = 0$ results in almost the same SSR value. Hence we conclude that the function describing the ratio is most likely the constant function 0.785.

For the number of nodes in the search space, the best fit in the polylogarithmic model is $f(n) = 10^{-11} \log_2(n)^{9.77}$. Note that $a = 10^{-11}$ can also be interpreted as a change of the log base. Nevertheless this function does not look natural at all. This observation is strengthened by the high ASE value for a and the large SSR term. In contrast, the polynomial model initially return the function $g(n) = 0.134 \cdot n^{0.516}$ with much smaller ASE values for both parameters (only

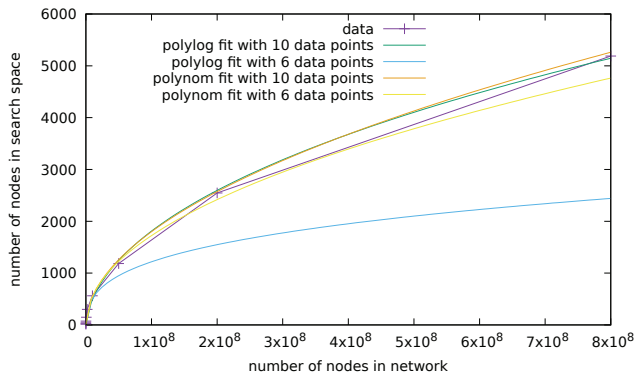

Figure 4: Average CH search space size in dependency of n. The plot shows the data points as well as the fitted functions for the different models and input sizes.

7.2% for c compared to 59.5% for a) and an SSR value which is more than 5 times smaller than the one obtained in the polylogarithmic model. Refitting after setting d to 0.5 results in a larger SSR value but still leads to a better quality than the polylogarithmic model in all aspects. Hence we conclude that the growth of the number of nodes in the search space of CH is most likely well described by $0.18\sqrt{n}$. To further support the evidence that the polynomial model is the correct one, we also performed the fitting using only the first six data points, that is, only up to $n = 2.7 \cdot 10^6$. The resulting functions are plotted in Figure 4. We observe that the function derived in the polynomial model for 6 data points is still a good fit for all 10 data points. In fact, the functions for 6 and 10 data points differ only slightly: $g_6(n) = 0.211 \cdot n^{0.489}$ and $g_{10}(n) = 0.134n^{0.516}$ and both produce small SSR values after refitting with $d = 0.5$ being fixed. In contrast, the function $f_6(n)$ in the polylogarithmic model (lowest plot line in Figure 4) is not a good fit for all 10 data points at all. The exponent b was only 6.499 for f_6 while it is 9.77 for f_{10}. This shows that the polylogarithmic model is not stable and most likely the exponent would chnge again if we would investigate even larger networks. In summary, the polynomial model clearly seems to fit best.

The number of edges in the search space $|E^{\uparrow}|$ was not evaluated with respect to n but instead with respect to $|V^{\uparrow}|$, the number of nodes in the search space. We chose this different evaluation approach because it can be easily deduced that $|V^{\uparrow}| - 1 \leq |E^{\uparrow}| \leq |V^{\uparrow}|^2$ (the lower bound is true as the search space induces a connected graph, the upper bound is trivially true as there can be no more than quadratically many edges in the graph induced by the search space). Hence we already know that the polynomial model has to be the correct one. Nevertheless, the polylogarithmic model also produces a fit with small ASE and SSR values. However, again, the resulting function $f(|V^{\uparrow}|) = 10^{-7}\log_2(|V^{\uparrow}|)^{10.565}$ is not intuitive at all – and meaningless, given that we know that the polynomial model is valid here. The polynomial model suggest that the dependency of $|E^{\uparrow}|$ and $|V^{\uparrow}|$ can be roughly described by $|E^{\uparrow}| = 2|V^{\uparrow}|^{1.3}$, which is far from the quadratic upper bound (which often is used in theoretical analyses).

TN					
number of access nodes $\|AN\|$					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	10^{-12}	8.903	160.6%	5.3%	48
g(n)	c	d	ASE(c)	ASE(d)	SSR
	0.006	0.470	34.3%	3.6%	24
refit d=0.5	0.003		1.8%		32
radius r					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	10^{-9}	8.205	41.2%	1.5%	28575
g(n)	c	d	ASE(c)	ASE(d)	SSR
	0.900	0.441	10.7%	1.2%	20660

Table 5: Fitting results for TN.

HL					
number of hub labels (CH-based) $\|HL_1\|$					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	10^{-4}	4.157	47.3%	3.4%	325
g(n)	c	d	ASE(c)	ASE(d)	SSR
	2.00	0.244	3.9%	0.8%	22
refit d=0.25	1.79		0.5%		45
number of hub labels (skeleton-based) $\|HL_2\|$					
f(n)	a	b	ASE(a)	ASE(b)	SSR
	0.196	2.629	21.4%	2.5%	4918
refit b=2.5	0.298		1.2%		7393
g(n)	c	d	ASE(c)	ASE(d)	SSR
	49.73	0.166	16.9%	5.5%	27954

Table 6: Fitting results for HL.

Transit Nodes The fitting results for TN are summarized in Table 5. We first looked for a function that described the number of access nodes in dependency of the network size. The polylogarithmic model produces again a function with a very small coefficient but a large exponent, which are accompanied by a large ASE value for a. The polynomial model produces an SSR value that is half of the SSR value for $f(n)$, and the ASE values are smaller as well. As the initially produced value for d is close to 0.5 we rerun the fitting after rounding d. The resulting function $g(n) = 0.003\sqrt{n}$ still exhibits a low SSR value.

A closer look on Table 3 reveals, that the radius r is always larger than the number of nodes in the search space of CH, hence the growth function is likely to be polynomial here as well. And indeed, fitting to $g(n)$ leads to lower ASE and SSR values than $f(n)$.

Hub Labels The fitting results for HL are summarized in Table 6. We clearly observe from the results given in Table 3 that the CH-based hub label sets HL_1 are superior to the skeleton-based hub labels HL_2, as they are smaller on average in all considered networks.

For CH-based hub labels, the polynomial model is the clear winner. The SSR value is smaller by a factor of almost 15 and the ASE values are significantly smaller as well. Even after fixing $d = 0.25$, we only get an SSR value of 45, hence our function fits the data very well. We conclude that

	CH		TN		HL (CH-based)		HL (Skeleton-based)	
	query	space	query	space	query	space	query	space
fitted functions	$\mathcal{O}(\sqrt{n})$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n\sqrt{n})$	$\mathcal{O}(n^{0.25})$	$\mathcal{O}(n^{1.25})$	$\mathcal{O}(\log^{2.5} n)$	$\mathcal{O}(n \log^{2.5} n)$

Table 7: Search spaces and space consumption of the route planning techniques derived from our scalability study.

$g(n) = 1.8n^{0.25}$ appears to be a good description for the growth of the hub label sizes for HL_1.

For HL_2, although the label sizes are larger compared to HL_1, the polylogarithmic model produces a smaller SSR value than the polynomial model, even after fixing $b = 2.5$. The ASE value for a is rather large, though. Hence it is difficult to decide if the polylogarithmic model is really superior. When we fitted all data points except the one with largest n, the SSR for the polynomial model decreased by a factor of 3 whereas for the polylogarithmic model it remained nearly unchanged. This might give further evidence that the label sizes grow indeed polylogarithmicly. If this is the case, the skeleton-based preprocessing would outperform the CH-based approach in very large networks ($n > 6.7 \times 10^{13}$).

Comparison to Theoretical Bounds

From the functions we derived by curve fitting, we can deduce the search space sizes in a query and the space consumption of the CH, TN and HL data structures: For CH, the search space is defined as two times the number of nodes in $V^{\uparrow}$, and the space consumption as the number of shortcuts $|E^{+}|$. For TN, the search space equals $|AN|^2$ and the space consumption $n|AN|+\Theta(n)$. For HL, the query time is linear in $|HL|$ and the space consumption is $n|HL|$. The respective functions are provided in Table 7.

We observe that CH and HL are incomparable, as CH exhibits a lower space consumption but HL a better query time. TN however is dominated by CH and HL, as the query times and the space consumption turn out to be worse. We want to stress here, though, that we only tested one out of many different preprocessing schemes for TN here. Other preprocessing schemes might result in better running times and/or space consumption bounds, especially considering that the radius for long queries can be chosen very differently.

If we now compare Table 7 to the theoretical bounds given in Table 1, we make the following observations: If our bound for the search space size of CH is correct, then the bounds depending on the highway dimension h and the tree width t would be valid for $h, t \in \mathcal{O}(\sqrt{n})$. But this would result in a predicted space consumption of $\mathcal{O}(n\sqrt{n}\log n)$ in both models which is huge compared to the linear space consumption function we derived. For TN search spaces it yields $\mathcal{O}(h^2) = \mathcal{O}(n)$, hence this would also imply $h \in \mathcal{O}(\sqrt{n})$. Assuming such an h value, the theoretical space consumption would be $\mathcal{O}(n\sqrt{n})$ which matches our fitted function perfectly. For HL_1, to match our bounds, the highway and the skeleton dimension only need to be in $\mathcal{O}(n^{0.25})$. Then the search space and the space consumption function would coincide. However, the analysis in dependency of h predicts the same growth behavior of the search spaces for both, CH and HL, while our analysis indicates that the search spaces induced by HL are significantly smaller – at least for the tested preprocessing schemes. If our bounds for HL_2 are correct, this implies that k grows at most polylogarthmicly in n as we applied the same preprocessing here that was developed for the theoretical analysis.

The bounds derived for minor closed graphs and graphs with bounded growth fit our functions for CH very well. In fact, they differ at most by a factor of $\log n$ from our results. For TN, the bounded growth model predicts better query times and for HL worse query times and space consumption bounds, but the preprocessing schemes analyzed there also differ from the ones we implemented here.

In summary, some of our derived functions are matched by the theoretical predictions (at least if the parameters h, k, and t assume suitable values) while others can hardly be explained by existing models for road networks. It would be helpful to know the values of h, t, k of large road networks in order to make the comparison of the models more conclusive. But so far, their true values have not been investigated and at least for h and t exact computation is NP-hard.

Conclusions and Future Work

We provided a scalability study for the three state-of-the-art route planning techniques in road networks, contraction hierarchies, transit nodes and hub labels, considering their most common practical preprocessing schemes. Based on experiments on (generated) road networks of different size, we derived for the first time empirical functions that describe the search space sizes and the space consumption of these techniques. Our results encourage the search for road networks models in which the preprocessing schemes used in practice become analyzable, which unfortunately is not the case for most of the road network models considered so far. In practice, additionally many engineering approaches are used to reduce the space consumption and the query times further. It would be interesting to see if these methods only shave off constant factors of the running time or whether they really change the scaling functions. In future work, also other road networks should be investigated (e.g. the US network with a lot of grid substructures, and networks from other sources than OpenStreetMap). Finally, to be able to study even larger networks (which might result in more truthful functions), new versions of the preprocessing schemes need to be developed that are more efficient in terms of space consumption and preprocessing time.

References

Abraham, I.; Fiat, A.; Goldberg, A. V.; and Werneck, R. F. 2010. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 782–793. SIAM.

Abraham, I.; Delling, D.; Fiat, A.; Goldberg, A. V.; and Werneck, R. F. 2011a. VC-dimension and shortest path algorithms. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 6755 of *Lecture Notes in Computer Science*, 690–699. Springer.

Abraham, I.; Delling, D.; Goldberg, A. V.; and Werneck, R. F. 2011b. A hub-based labeling algorithm for shortest paths in road networks. In *Proceedings of the 10th International Conference on Experimental Algorithms (SEA)*, volume 6630 of *Lecture Notes in Computer Science*, 230–241. Springer.

Abraham, I.; Delling, D.; Fiat, A.; Goldberg, A. V.; and Werneck, R. F. 2013. Highway dimension and provably efficient shortest path algorithms. Technical Report MSR-TR-2013-91, Microsoft Research.

Arz, J.; Luxen, D.; and Sanders, P. 2013. Transit node routing reconsidered. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA)*, volume 7933 of *Lecture Notes in Computer Science*, 55–66. Springer.

Bast, H.; Funke, S.; Sanders, P.; and Schultes, D. 2007. Fast routing in road networks with transit nodes. *Science* 316(5824):566–566.

Bast, H.; Delling, D.; Goldberg, A. V.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; and Werneck, R. F. 2016. Route planning in transportation networks. In *Algorithm Engineering*. Springer. 19–80.

Bauer, R.; Krug, M.; Meinert, S.; and Wagner, D. 2010. Synthetic road networks. In *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, volume 6124 of *Lecture Notes in Computer Science*, 46–57. Springer.

Bauer, R.; Columbus, T.; Rutter, I.; and Wagner, D. 2013. Search-space size in contraction hierarchies. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7965 of *Lecture Notes in Computer Science*, 93–104. Springer.

Blum, J.; Funke, S.; and Storandt, S. 2018. Sublinear search spaces for shortest path planning in grid and road networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press.

Delling, D.; Goldberg, A. V.; Nowatzyk, A.; and Werneck, R. F. 2011. PHAST: Hardware-accelerated shortest path trees. In *Proceedings of the 25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 921–931. IEEE.

Dibbelt, J.; Strasser, B.; and Wagner, D. 2014. Customizable contraction hierarchies. In *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA)*, volume 8504 of *Lecture Notes in Computer Science*, 271–282. Springer.

Eisenstat, D. 2011. Random road networks: The quadtree model. In *Proceedings of the 8th Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, 76–84. SIAM.

Eppstein, D., and Goodrich, M. T. 2008. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, 16. ACM.

Geisberger, R.; Sanders, P.; Schultes, D.; and Vetter, C. 2012. Exact routing in large road networks using contraction hierarchies. *Transportation Science* 46(3):388–404.

Kosowski, A., and Viennot, L. 2017. Beyond highway dimension: Small distance labels using tree skeletons. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1462–1478. SIAM.