Provable Efficiency of Contraction Hierarchies with Randomized Preprocessing

Stefan Funke[1] and Sabine Storandt[2]

[1] FMI, University of Stuttgart (Germany)
funke@fmi.uni-stuttgart.de
[2] Department of Computer Science, University of Freiburg (Germany)
storandt@informatik.uni-freiburg.de

Abstract. We present a new way of analyzing Contraction Hierarchies (CH), a widely used speed-up technique for shortest path computations in road networks. In previous work, preprocessing and query times of deterministically constructed CH on road networks with n nodes were shown to be polynomial in n as well as the highway dimension h of the network and its diameter D. While h is conjectured to be polylogarithmic for road networks, a tight bound remains an open problem. We rely on the empirically justifiable assumption of the road network exhibiting small growth. We introduce a method to construct randomized Contraction Hierarchies on road networks as well as a probabilistic query routine. Our analysis reveals that randomized CH lead to sublinear search space sizes in the order of $\sqrt{n}\log\sqrt{n}$, auxiliary data in the order of $n\log^2\sqrt{n}$, and correct query results with high probability after a polynomial time preprocessing phase.

1 Introduction

Contraction Hierarchies (CH) [1] are a preprocessing based technique to accelerate shortest path computations in road networks. The basic idea behind CH is to augment the network with shortcut edges which allow to settle less nodes in a Dijkstra run without compromising correctness of the result. CH are widely used on real-world instances, as they provide an excellent trade-off between the amount of auxiliary data (only doubling the network size) and speed-up (about three orders of magnitude compared to a plain Dijkstra). But these values are solely empirical, based on experiments on real-world networks [2]. Theoretical explanations for this good empirical behaviour are still not fully satisfying.

1.1 Related Work

In [3], the notion of the *highway dimension* h of a network was introduced to explain the practical performance of CH and other speed-up techniques (on undirected networks). A small highway dimension indicates that shortest paths in the road network longer than a parameter r can be hit by a set S of nodes with S being locally sparse. Here, locally sparse means that the intersection of a ball of radius r with S contains at most h elements. Assuming optimal preprocessing, it was shown that $\mathcal{O}(nh \log D)$ shortcut edges are added to the original network, with n being the number of nodes in the network and $D \leq n$ the network diameter. The number of nodes settled in a CH-Dijkstra run was shown to be in $\mathcal{O}(h \log D)$. As optimal CH preprocessing is NP-hard (using Hitting Set computations as a subroutine), they also study a polynomial time approximation version. This adds another factor of $\log n$ to the auxiliary data size and the query time. While h is conjectured to be polylogarithmic for road networks, the problem of proving h to be small is still open (for grids it is known that $h \in \Theta(\sqrt{n})$). Moreover, h-values for real-world networks are unknown (as its computation is NP-hard as well) and the preprocessing methods introduced to study CH theoretically are too slow to be practical for large road networks [4]. Hence validating whether the theoretical results reflect real-world behavior is difficult.

In [5], CH were studied based on the topology of the network. It was shown that for planar graphs, CH preprocessing based on nested dissection leads to auxiliary data in the order of $\mathcal{O}(n \log n)$. For minor-closed graphs with balanced $\mathcal{O}(\sqrt{n})$ separators, search spaces are shown to be in $\mathcal{O}(\sqrt{n})$, for graphs with treewidth k in $\mathcal{O}(k \log n)$. For graphs with highway dimension h, results matching those in [3] were reported assuming edge costs that maximize h. An implementation of CH based on nested dissection [6] showed that it leads to good performance in practice. Nevertheless a real comparison to the theoretical results again is hardly possible due to h being unknown. Moreover all results so far heavily use Big-O-Notation, making it difficult to tell whether the observed behaviour in practice is due to asymptotics or due to hidden constants.

1.2 Contribution

We exhibit a so far unexplored connection of CH to Skip Lists [7], a data structure for fast search within ordered sets of elements. Based on the model of randomized Skip List construction, we describe a CH variant with randomized preprocessing for some probability parameter $p \in]0, 1[$. We prove the expected number of shortcuts to be at most $n(1-p)(0.5 \cdot \log^2_{1/p} \sqrt{n} + (1-p^2)^{-1})$, and the expected search space size to be $(6 \ln {}^1\!/\!_p \log \sqrt{n} + 2)\sqrt{n}$ (no O-notation here!). Preprocessing is in polynomial time. For our results to be valid, we rely on a simple and intuitive bound on the growth rate of the underlying metric. Moreover, we prove our theoretical bounds to be meaningful by comparing them to experimental results on real-world road networks. Surprisingly, the randomized construction shares certain characteristics with a common heuristic CH construction scheme. For this simple heuristic construction, no theoretical guarantees of any kind are known. While heuristically constructed CH naturally outperform CH with randomized preprocessing, our studies are the first to give some insight in the theoretical auxiliary data size and search space parameters for this heuristic construction.

2 Preliminaries

In the following, we describe preprocessing and query answering for conventional CH carefully. We then briefly review randomized Skip Lists to show their potential to serve as model for randomized CH construction. Finally, we provide some details on the graph model that is used in our analysis.

2.1 Contraction Hierarchies

Given a road network $G(V, E)$ and edge costs $c : E \to \mathbb{R}^+$, the preprocessing phase of CH works as follows: Every node $v \in V$ gets assigned a level $l : V \to \mathbb{N}$ inducing a (not necessarily total) order on the nodes. Then a CH-graph $G'(V, E \cup E^+)$ is constructed upon this order where E^+ denotes the set of shortcut edges. For a pair of nodes v, w a shortcut edge $e = (v, w)$ is added to E^+ if all nodes on the shortest path between v and w exhibit a level smaller than $\min\{l(v), l(w)\}$. The cost of e is set to the shortest path distance between v and w. Determining node levels that minimizes $|E^+|$ is APX-hard [8],[9].

In practice, though, there exist heuristics which construct CH-graphs with small sets of shortcut edges very efficiently. The most common heuristic is based on the *node contraction* operation [1]. Here, a node v and all its adjacent edges are removed from the current graph, and shortcut edges are inserted between any pair of neighbors u, w of v if u, v, w was the shortest path from u to w. Nodes are contracted one-by-one and their rank in the contraction order is used as node level. If the goal is keeping $|E^+|$ small, a good candidate for the next node to contract is the one with minimal *edge difference* (ED), which denotes the number of added shortcuts if v is contracted minus the number of edges that are currently adjacent to v (some heuristics also consider a linear combination of the ED and other values). So after contraction of v, the current graph has one node less and $ED(v)$ more edges (note that $ED(v)$ can be negative). The ED-values need to be continuously updated, as contracting a node influences the ED-values of its neighbors. It was noted, though, that independent sets of nodes can be contracted at once without violating correctness. So nodes in the current graph are first sorted increasingly by their ED-value. Then an independent set of nodes is chosen greedily considering the nodes in the ED-order (with all nodes in the set receiving the same level). This approach allows to construct the CH-graph in very few contraction rounds and leads to few added shortcuts in practice.

The final CH-graph (original graph plus all shortcuts) has the following nice property (no matter how l was chosen): Between any pair of nodes s, t there exists a shortest path on which the node levels at first monotonously increase and then monotonously decrease (so the path is unimodal wrt. l). Therefore queries can be answered via a bi-directional Dijkstra computation that only relaxes upward edges (v, w) with $l(v) \leq (w)$ in the forward run and accordingly downward edges in the backwards run. By construction, the forward and the backward run both settle the node(s) with the highest level on the shortest path from s to t. Hence a node that minimizes the forward plus the backwards distance yields the optimal shortest path distance.

2.2 Randomized Skip Lists

Skip Lists are a data structure for efficient search and maintenance of an ordered set of elements. Skip Lists consist of layers of linked lists. The bottom layer is a linked list that contains all n elements in sorted order. Each element gets assigned a height h. To determine the height values randomly, first a probability $p \in]0,1[$ is chosen. Then for every element a coin with probability p for $HEAD$ is flipped until $TAIL$ shows up. The number of times the coin showed $HEAD$ marks the height. The maximum height among all elements determines the number of additional layers in the Skip List data structure. Each list i contains only links between elements with a height $\geq i$ and 'skips' over the others. In expectation, the maximum height is $\log_{1/p} n$ and therefore the total space consumption is in $\mathcal{O}(n \log_{1/p} n)$. Searching for an element using a suitable query algorithm that works its way from the topmost layer down demands ${}^1\!/\!{}_p \log_{1/p} n$. Choosing different values of p allows to trade search costs against storage costs. We will use randomized Skip List construction as model for randomized CH construction by interpreting the road network as the bottom layer.

2.3 Our Model: Graph Metrics with Bounded Growth

Like [3] we need to make some assumptions about the structure of our road networks for an analysis to succeed. In typical representations of a road network (as for example derived from data of the OpenStreetMap project) as graphs $G(V,E,c)$ with edge costs $c: E \rightarrow \mathbb{N}$, edges represent road segments of rather uniform length, so we can replace edge costs by respective sequences of unit-cost edges withouth blowing up the size of the graph by more than a constant factor. So from now on we will focus on graphs with unit edge costs. Our crucial assumption on the structure of G can be stated as follows: For any node $v \in V$, the number of nodes w at distance k is bounded by $g \cdot k$ for some constant $g \geq 1$, that is

$$|\{w \in V : d(v,w) = k\}| \leq g \cdot k$$

We have verified this condition to hold for small values of g for several real-world networks. It also implies $|\{w \in V : d(v,w) \leq k\}| \leq gk(k+1)/2$ – which mimics the area growth in R^2 when increasing the radius of a circle. To keep the presentation simpler, we assume in the following $g = 1$, but it is easy to see that the parameter g could be carried along all following calculations.

Our condition has some connection to already existing characterizations of graph metrics. For example, demanding that the number of nodes at distance k is *exactly* $g \cdot k$ implies an *expansion rate* of 4 according to the definition of [10] as well as constant doubling dimension [11]. On the other hand there are metrics with an unbounded expansion rate yet satisfying our condition.

3 Randomized Contraction Hierarchies

3.1 Preprocessing

We start the preprocessing phase by assigning levels $l : V \rightarrow \mathbb{N}$ by coin tosses in the same way as for Skip Lists (with a probability p for *HEAD*). So $l(v)$ is an integer greater than zero with $P(l(v) \geq L) = p^{L-1}$.

To complete the preprocessing, we need to compute the set of shortcuts resulting from our randomized choice of node levels. To that end, we run a Dijkstra computation from each node v until on every active path in the search tree there is a node with a level $\geq l(v)$. For every first node w on a shortest path from v with $l(w) \geq l(v)$ we insert the shortcut $e = (v, w)$ with $c(e) = d_v(w)$ in the CH-graph (avoiding multi edges).

The preprocessing obviously demands only polynomial time. We expect a maximum node level of $\mathcal{O}(\log n)$, so assigning levels to n nodes can be done in expected $\mathcal{O}(n \log n)$ time. The n Dijkstra runs in the second phase require $\mathcal{O}(n^2 \log n + nm)$ time and dominate the overall runtime.

3.2 Analysis

Let us now analyze our CH construction. The two key performance indicators are the *total number of shortcuts* and the *number of settled nodes* in a query. Ideally, both of these values should be small in order to guarantee a space-efficient CH-graph and a good speed-up compared to plain Dijkstra's algorithm.

Throughout the analysis, log always refers to $\log_{1/p}$.

Total Number of Shortcuts To bound the total number of shortcuts we first bound the number of upward edges emanating from some node v. We provide two such bounds, one being stronger for nodes v with small levels, the other being stronger for nodes v with large levels.

Lemma 1. *The expected number of upwards edges (original or shortcut) emanating from a node v with level L is bounded by p^{1-L}.*

Proof. A shortcut (v, w) from v to a node w with a shortest path $v \rightsquigarrow w$ of length k exists if and only if the level of w is at least L while the level of all $k-1$ nodes inbetween on the shortest path from v to w is less than L. So the probability for the shortcut (v, w) to exist can be expressed as $P(l(w) \geq L) \cdot P(l < L)^{k-1}$. Due to our condition we have at most k nodes at distance k, hence the total number of upward edges can be bounded as:

$$E(X) \leq \sum_{k=1}^{D} k \cdot P(l \geq L) \cdot P(l < L)^{k-1} = \sum_{k=1}^{D} kp^{L-1}(1 - p^{L-1})^{k-1}$$

Here D is the diameter of the graph, $D \leq n$. We then substitute $1 - p^{L-1}$ with q and end up with:

$$E(X) \leq \frac{1-q}{q} \sum_{k=1}^{D} kq^k < \frac{1-q}{q} \sum_{k=0}^{\infty} kq^k = \frac{1-q}{q} \cdot \frac{q}{(1-q)^2} = \frac{1}{p^{L-1}} \qquad \square$$

Lemma 2. *The expected number $E(X)$ of upwards edges (original or shortcut) emerging from a node v with level L is bounded by np^{L-1}.*

Proof. Upwards shortcuts demand the target node to have a level $\geq L$. Therefore the total number of shortcuts emerging from a node with level L is bounded by the expected number of nodes with a level $\geq L$ in the network. As the expected number of nodes with level L equals $n(1-p)p^{L-1}$, the expected number of nodes with a level at least L is np^{L-1}. $\square$

We observe that the bound by Lemma 1 is tighter for $L \leq \log \sqrt{n}$ and the bound by Lemma 2 for $L > \log \sqrt{n}$.

Theorem 1. *The expected number of upwards edges in the CH-graph is bounded by* $n(1-p)(0.5 \cdot \log^2 \sqrt{n} + (1-p^2)^{-1})$.

Proof. Using Lemma 1 and the fact that we expect $np^{L-1}(1-p)$ nodes at level L we bound the number of outgoing edges from nodes with level $\leq \log \sqrt{n}$ by

$$\sum_{L=1}^{\log \sqrt{n}} n(1-p)p^{L-1}p^{1-L} \leq n(1-p) \cdot 0.5 \cdot \log^2 \sqrt{n}$$

and the number of edges from nodes with higher level using Lemma 2 by

$$\sum_{L=\log \sqrt{n}+1}^{\infty} n(1-p)p^{L-1}np^{L-1} = \frac{n^2(1-p)}{p^2} \cdot \sum_{L=\log \sqrt{n}+1}^{\infty} p^{2L}$$

$$= \frac{n^2(1-p)}{p^2} \cdot \frac{p^2(p^{\log n} - p^{2n})}{1-p^2} = \frac{n^2(1-p)(1/n - p^{2n})}{1-p^2} \leq \frac{n(1-p)}{1-p^2} \quad \square$$

The analysis for the number of downwards edges can be done analogously. So the final number of expected edges in the CH-graph is $n(1-p)(0.5 \cdot \log^2 \sqrt{n} + (1-p^2)^{-1})$ for undirected networks (summing up the two bounds in the Theorem) and twice this number, i.e., $n(1-p)(\log^2 \sqrt{n} + 2(1-p^2)^{-1})$ for directed networks.

Search Space Analysis We define the search space $SS(v)$ for a node $v \in V$ as the number of nodes that are pushed into the priority queue (PQ) during a CH-Dijkstra run from v (relaxing only upwards edges). We will first analyze the *direct search space (DSS)* of v. A node w is in $DSS(v)$ if on the shortest path from v to w all nodes have levels $\leq l(w)$. Therefore, w will be settled with the correct distance $d(v, w)$ in the CH-Dijkstra run. Unfortunately, $SS(v)$ is typically a superset of $DSS(v)$ as also nodes on monotonously increasing but non-shortest paths are considered. We will modify the query algorithm to bound the number of such nodes.

Lemma 3. *The expected size of $DSS(v)$ is bounded by $(1+p)\sqrt{n}$.*

Proof. We can assume that all nodes with a level $l > \log\sqrt{n}$ are always in $DSS(v)$. In expectation, there are $\sum_{L=\log\sqrt{n}+1}^{\infty} np^L(1-p) = \sqrt{n}$ such nodes in the network. A node w is in $DSS(v)$ if on the shortest path from v to w all nodes have a level of at most $l(w)$. The expected number of such nodes with $l(w) \leq \log\sqrt{n}$ can be bounded by

$$\sum_{L=1}^{\log\sqrt{n}} P(l=L) \sum_{k=1}^{D} kP(l \leq L)^{k-1} = \sum_{L=1}^{\log\sqrt{n}} p^{L-1}(1-p) \sum_{k=1}^{D} k(1-p^{L-1})^{k-1}.$$

As the last sum can be bounded by p^{-2L+2}, we get:

$$\sum_{L=1}^{\log\sqrt{n}} p^{-L+1}(1-p) = p(\sqrt{n}-1)$$

Together with the at most $\sqrt{n}$ nodes in $DSS(v)$ with a level greater than $\log\sqrt{n}$, the size of $DSS(v)$ is bounded by $(1+p)\sqrt{n}$. □

To characterize and reduce the number of nodes in $SS(v) \setminus DSS(v)$, we need the following properties about nodes in $DSS(v)$.

Lemma 4. *The probability for a node w at distance k from v to be in $DSS(v)$ but exhibiting a level $l(w) < \log k - \log(c\ln(1/p)\log k)$ is bounded by k^{-c}.*

Proof. If $w \in DSS(v)$, all k nodes on the shortest path from v to w have a level of at most $l(w)$. As $l(w) < \log k - \log(c\ln(1/p)\log k)$ the same needs to hold for all nodes on this shortest path. The probability for that can be expressed as:

$$\left(1 - p^{\log k - \log(c\ln(1/p)\log k)}\right)^k = \left(1 - \frac{c\ln(1/p)\log k}{k}\right)^k$$

Using $(1+x) \leq e^x$ with $x = -c\ln(1/p)\log k \cdot k^{-1}$, we can upper bound the above formula by

$$\left(e^{-c\ln(1/p)\log k \cdot k^{-1}}\right)^k = e^{-c\ln(1/p)\log k} = e^{-c\ln(1/p)\ln(k)/ln(1/p)} = k^{-c}. \quad □$$

Applying the above Lemma we show that with high probability a node in $DSS(v)$ whose shortest path from v is at least $n^{1/4}$ long does not have too small a level.

Lemma 5. *A node w at shortest path distance $k > n^{1/4}$ from v is in $DSS(v)$ and exhibits a level $l(w) \geq \log k - \log(c\ln(1/p)\log k)$ with probability $\geq 1 - n^{-c/4}$.*

Proof. The probability $P = P(l(w) < \log k - \log(c\ln(1/p)\log k))$ is bounded by k^{-c} (according to Lemma 4). So the larger k the smaller the probability. For $k > n^{1/4}$ we get $P < n^{-c/4}$. Hence we can lower bound the probability of the counter-event by $1 - n^{-c/4}$. □

Armed with this insight, we modify our query algorithm such that nodes with too small a level relative to their distance are discarded during the exploration. That is, during the run of CH-Dijkstra, we discard a node w from further consideration (not pushing it into the PQ) if $d(w) > n^{1/4}$ and $l(w) < \min(\log\sqrt{n}, \log d(w) - \log(c\ln(1/p)\log d(w)))$, where $d(w)$ denotes the current distance label of w in the CH-Dijkstra run. The following theorem shows that for appropriate choice of c, this leads to small search spaces and with high probability to the correct result.

Theorem 2. *Our modified query algorithm has an expected search space size of at most $\sqrt{n}(2 + c\ln(1/p)\sqrt{2}\log\sqrt{n})$ and computes the correct result with probability $\geq 1 - 2n^{\frac{-c+4}{4}}$.*

Proof. We know that always $d(v,w) \leq d(w)$ has to be true, where $d(v,w)$ is the true distance from v to w. Therefore, the number of nodes with $d(v,w) \leq d(w) \leq n^{1/4}$ can be bounded by $\sum_{i=1}^{n^{1/4}} k \leq \sqrt{n}$. The number of nodes with $l(w) \geq \log d(w) - \log(c\ln(1/p)\log d(w))$ can be bounded by

$$\sum_{k=1}^{D} x_k P(l \geq \log k - \log(c\ln(1/p)\log k)$$

with $x_k \leq k$, $\forall k = 1, \ldots, D$ and $\sum x_k = n$. As $P(l \geq \log k - \log(c\ln(1/p)\log k)$ decreases with growing k, this sum can be upper bounded by:

$$\sum_{k=1}^{\sqrt{2n}} kP(l \geq \log k - \log(c\ln(1/p)\log k) = \sum_{k=1}^{\sqrt{2n}} k\frac{c\ln(1/p)\log(k)}{k}$$

$$= c\ln(1/p)\sum_{k=1}^{\sqrt{2n}} \log k \leq c\ln(1/p)\sqrt{2n}\log\sqrt{n}$$

Together with the at most $\sqrt{n}$ nodes above level $\log\sqrt{n}$ in expectation, our search space size does not exceed $\sqrt{n}(2 + c(\ln(1/p)\sqrt{2}\log\sqrt{n})$ nodes.

It remains to show that queries are answered correctly with high probability. Queries are answered correctly for sure if $SS(v) \supseteq DSS(v)$. According to Lemma 5, a node $w \in DSS(v)$ at distance $k > n^{1/4}$ is not contained in our pruned search space with probability at most $n^{-c/4}$. We are interested in an upper bound for the probability that at least one of the nodes in $DSS(v)$ is not in our search space. We simply apply the union bound upper bounding the probability that one or more nodes of $DSS(v)$ do not have large enough level by $n \cdot n^{-c/4} = n^{(-c+4)/4}$. So with probability $\geq 1 - n^{\frac{-c+4}{4}}$, all nodes of $DSS(v)$ are actually in the search space of v and with the same argument holding for the (reverse) search space of the target, we arrive at the bound for the correctness of the query result. □

The above Theorem implies for $c = 4 + \alpha$, $\alpha > 0$ our query routine produces the correct result with probability $\geq 1 - 2n^{-\alpha/4}$. Choosing for example $c = 6$ we have a success probability of $1 - 2/\sqrt{n}$ and expected search space sizes for source and target of less than $\sqrt{n}(2 + 6\log\sqrt{n})$ for $p = 1/2$.

4 Experimental Results

We implemented randomized CH construction and the proposed query answering algorithm in C++. We also implemented the heuristic CH construction based on iterative contraction of independent sets as described in Section 2.1. Experiments were conducted on a single core of an Intel i5-4300U CPU with 1.90GHz and 12GB RAM. We used the OSM road network data of a cut-out of Germany with 2,275,793 nodes and 4,637,537 directed edges for evaluation.

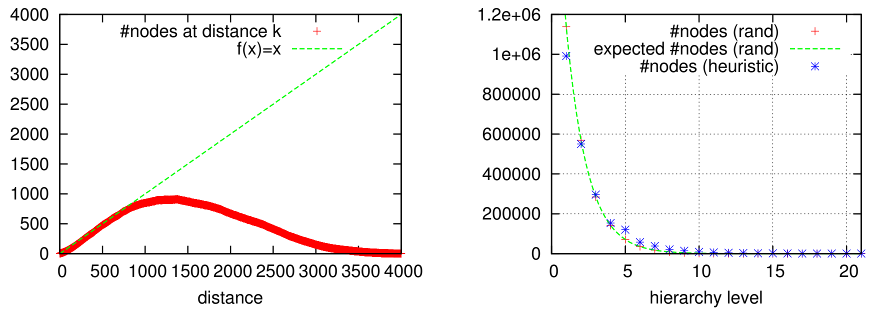

Fig. 1. Left: Our model predicts the number of nodes with distance k to be beneath the green line. The red boxes indicate the real distance dependent node distribution based on Dijkstra search trees from 1,000 randomly chosen source nodes. **Right:** Node level distribution resulting from heuristic and randomized CH construction.

We first validate our chosen model. In Figure 1, left, we compare the average number of nodes at distance k from a source in our real network (with euclidean distances as cost metric) to the prediction according to the model. We observe that up to distance about 1000 there are indeed almost exactly k nodes with distance k. Then the number declines.

Unless mentioned otherwise, the following experiments are conducted using $p = 1/2$ for randomized CH. We first want to evaluate the CH preprocessing. Figure 1, right, shows that the expected number of nodes per level reflects the real node levels quite perfectly. While this is not surprising for the randomized construction, it is indeed for the heuristic construction. So basically in every contraction round, about half of the remaining nodes form an independent set and get contracted at once, leading to the same node level distribution as for our Skip List based randomized levels. The maximum level in the heuristic construction was 99, though, and therefore about a factor of 5 higher than in the randomized CH, but this is due to the fact that towards the end of the contraction process the remaining nodes form clique-like structures which only allow for contracting a single element as independent set.

For randomized CH, we expect the number of shortcuts to be less than 41 million for $n = 2,275,793$. Averaged over three runs, our CH-graph contained $23,758,675$ shortcuts. In the heuristically constructed CH-graph, $8,678,644$ short-

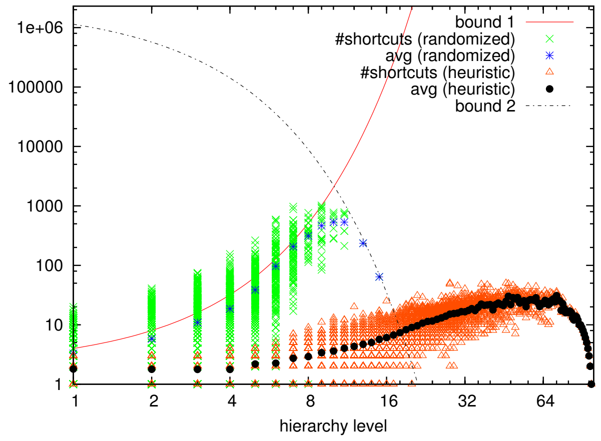

Fig. 2. Number of upwards shortcuts emerging from a node dependent on its level. Both axes are in logscale.

cuts are inserted, which is better by a factor of about 3. Figure 2 provides a detailed overview of the number of upwards shortcuts emerging from a node in dependency of its level. We observe that bound 1 and 2 resulting from Lemmas 1 and 2 are almost perfect predictions for the average value per level. For the heuristic construction the curve is stretched and exhibits a lower peak. This is a result of a wider range of node levels and a lower number of total shortcuts (due to the ED-related node contraction order).

Finally, we evaluated search spaces and query times for randomized and heuristic CH. In a query answered in the heuristically constructed CH-graph, 497 nodes were settled on average. Query times were in the order of a half microsecond which results in a speed-up of factor 500 compared to plain Dijkstra. With randomized CH, the predicted search space size for $c > 4$ is over 66,000. The number of actually settled nodes in our experiments was only 5,241, showing that some of our upper bound assumptions in the analysis are too pessimistic. All queries were answered correctly in our experiments. Query times were in the order of 20 ms, yielding only a speed-up of 10 compared to plain Dijkstra. Still, the fact that there is speed-up at all using a randomized construction shows that our results have some degree of practical justification. Moreover, it follows straight from the Skip List like construction that the expected number of nodes on the optimal CH-path from s to t with $d(s,t) = k$ is $2 \log k$. Evaluating 1,000 example queries, we observed that this result is matched accurately – indeed for both, randomized and heuristically constructed CH.

To study the influence of p, we ran the same experiments with $p = 1/4$ and $p = 3/4$. For $p = 1/4$, the expected maximum number of shortcuts is 32 million, the real number was $16,941,163$. For $p = 3/4$ our upper bound implies no more than 49 million expected edges in the CH-graph, and we ended up with $28,526,499$ in the experiments. The search space sizes and query times were slightly worse for both $p = 1/4$ and $p = 3/4$ compared to $p = 1/2$. If levels are more spread due to higher p they are also more connected due to more shortcuts. If levels are less spread there are large connected components of nodes with the same level which are explored until our level-distance-bound kicks in. Hence $p = 1/2$ seems to be a suitable choice, not only because it leads to a close relation with the heuristic CH construction but also since the auxiliary data size vs search space size trade-off is good.

5 Conclusions

It is rather surprising that it is possible to construct CH via a level assignment that does not take into account the structure of the graph. This is in stark contrast to common heuristic construction schemes as well as the approaches in [3] and [5] where the construction process is heavily guided by the graph structure. Furthermore, our randomized construction scheme shares some natural characteristics with the common heuristic construction scheme – both in theory as well as in empirical evaluations. Our results should be seen as a step towards a better understanding of the good performance of contraction hierarchies in practice.

References

1. Geisberger, R., Sanders, P., Schultes, D., Vetter, C.: Exact routing in large road networks using contraction hierarchies. Transportation Science **46**(3) (2012) 388–404
2. Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., Werneck, R.F.: Route planning in transportation networks. arXiv preprint arXiv:1504.05140 (2015)
3. Abraham, I., Fiat, A., Goldberg, A.V., Werneck, R.F.: Highway dimension, shortest paths, and provably efficient algorithms. In: Proc. 21st Ann. ACM-SIAM symposium on Discrete Algorithms. (2010) 782–793
4. Abraham, I., Delling, D., Fiat, A., Goldberg, A.V., Werneck, R.F.: Highway dimension and provably efficient shortest path algorithms. Microsoft Research, USA, Tech. Rep **9** (2013)
5. Bauer, R., Columbus, T., Rutter, I., Wagner, D.: Search-space size in contraction hierarchies. In: Automata, Languages, and Programming. Volume 7965 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2013) 93–104
6. Dibbelt, J., Strasser, B., Wagner, D.: Customizable contraction hierarchies. In: Experimental Algorithms. Springer (2014) 271–282
7. Pugh, W.: Skip lists: A probabilistic alternative to balanced trees. Commun. ACM **33**(6) (1990) 668–676
8. Bauer, R., Columbus, T., Katz, B., Krug, M., Wagner, D.: Preprocessing speed-up techniques is hard. In: 7th Int. Conf. on Algorithms and Complexity (CIAC'10),. Volume 6078 of Lecture Notes in Computer Science., Springer (2010) 359–370
9. Milosavljević, N.: On optimal preprocessing for contraction hierarchies. In: Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science, ACM (2012) 33–38
10. Karger, D.R., Ruhl, M.: Finding nearest neighbors in growth-restricted metrics. In: Proc 34th Annual ACM Symposium on Theory of Computing. STOC '02, New York, NY, USA, ACM (2002) 741–750
11. Gupta, A., Krauthgamer, R., Lee, J.R.: Bounded geometries, fractals, and low-distortion embeddings. In: Proc. 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14, IEEE Computer Society (2003) 534–543