Lower Bounds on the Query Times of Hub Labeling, Contraction Hierarchies, and Transit Node Routing

Colin White, Carnegie Mellon University, crwhite@cs.cmu.edu

Abstract

In the last decade, there has been a substantial amount of research in finding routing algorithms designed specifically to run on real-world graphs. In 2010, Abraham et al. showed upper bounds on the query time in terms of a graph’s highway dimension and diameter for some of the fastest algorithms, including CONTRACTION HIERARCHIES, TRANSIT NODE ROUTING, and HUB LABELING. In this paper, we show corresponding lower bounds for the same three algorithms. We also show how a known result that lower bounds the number of shortcuts added in the preprocessing stage for CONTRACTION HIERARCHIES, can be generalized to work for any preprocessing algorithm using our techniques.

1 Introduction

The problem of finding shortest paths in road networks has been well-studied in the last decade, motivated by the application of computing driving directions. Although Dijkstra’s algorithm runs in small polynomial time, for applications involving continental-sized road networks, Dijkstra’s algorithm is simply not fast enough. There have been many different approaches to find algorithms that specifically run fast on real-world graphs.

Most recent innovations involve a two-stage algorithm: a preprocessing stage and a query stage. The preprocessing stage runs once and can spend hours calculating data. Then the query stage uses this data to find shortest paths very fast, often several orders of magnitude faster than Dijkstra’s algorithm for a continental query. Once the preprocessing stage is completed, the users can run as many queries as they want. For a query between two nodes s and t (an s–t query), the algorithm returns dist(s, t), the cost of the shortest path between s and t. Most algorithms can also return the vertices on the shortest path using an extra data structure. For a comprehensive overview of the best routing algorithms, see [5].

Recently, Abraham et al. defined the notion of highway dimension [1], intuitively, the extent to which all shortest paths are hit by at least one of a small set of access nodes. There is evidence that real-world routing networks always have low highway dimension. Abraham et al. then showed strong upper bounds on the query times in terms of highway dimension, diameter, and number of vertices for four of the fastest routing algorithms: HUB LABELING, CONTRACTION HIERARCHIES, TRANSIT NODE ROUTING, and REACH. In this paper, we show corresponding lower bounds for the fastest three algorithms: HUB LABELING, CONTRACTION HIERARCHIES, and TRANSIT NODE ROUTING (the reach algorithm cannot be lower bounded using our approach). This makes the bound for HUB LABELING tight. For CONTRACTION HIERARCHIES and TRANSIT NODE ROUTING, the definition of highway dimension in the lower bound versus upper bound is slightly different, so we cannot quite say the bounds are tight.
The lower bounds use a family of graphs that were first devised to show that the number of shortcut edges in the preprocessing stage of CONTRACTION HIERARCHIES is tight for a wide range of parameters \(|V|\), highway dimension, and diameter \([12]\). However, this result assumes a specific (optimal) preprocessing algorithm. In the appendix, we will generalize this result to be independent of the preprocessing stage.

The paper will proceed as follows. Section 2 will provide background information on the two different definitions of highway dimension. In Section 3, we will review the definition and properties of the family of graphs used in our main theorems. Sections 3, 4, and 5, will explain HUB LABELING, CONTRACTION HIERARCHIES, and TRANSIT NODE ROUTING respectively, and show their corresponding lower bounds. Section 6 summarizes and discusses future work. The appendix contains a generalization of the lower bound for shortcut edges in \([12]\).

2 Highway Dimension

In general, we will assume graphs are undirected with nonnegative integral edge lengths and unique shortest paths. Relaxing these assumptions is discussed in Section 6.

\(B_r(v)\) represents all nodes \(u\) such that \(\text{dist}(u,v) < r\). We say a set of nodes covers a set of paths if each path has at least one of its vertices in the set of nodes. Now we will formally define the notion of highway dimension.

The highway dimension of a graph \(G = (V, E)\) is the smallest \(h\) such that for all \(r > 0\) and for all \(B_{4r}(v)\), there exists a set \(H \subseteq V\), such that \(|H| \leq h\) and \(H\) covers all shortest paths of length \(\geq r\) in \(B_{4r}(v)\).

Highway dimension was specifically designed to explain why the best routing algorithms perform well on real-world graphs but do not perform well on arbitrary graphs. Although it is too computationally intensive to calculate the exact highway dimension of a network as big as the USA, there is evidence that the highway dimension of real-world graphs is at most polylogarithmic in the number of vertices \([1]\).

Abraham et al. introduced a slightly refined version of the normal highway dimension in 2013 \([1]\).

The difference in the new definition versus the old one is that instead of having to hit all local shortest paths of length \(\geq r\), we have to hit all paths \(P\) where there is a shortest path \(P'\) with endpoints \(s\) and \(t\) such that \(l(P') > r\), \(P \subseteq P'\), and \(P' \setminus P \in \{\emptyset, \{s\}, \{t\}, \{s,t\}\}\). That is, we have to hit all paths that can be obtained by removing zero, one, or both endpoints of a shortest path with length \(> r\). We will refer to a graph’s highway dimension as \(h\) for the first definition, and \(\hat{h}\) for the second definition.

The two definitions of highway dimension are very similar but have a few key differences. Most notably, the new definition bounds the degree of the graph, which was not true before. Given any vertex, take a ball of radius \(\frac{1}{2}\) around the vertex. Then every neighboring vertex must be hit according to the definition, so \(\hat{h} \geq \Delta\), where \(\Delta\) is the maximum degree of the graph. This allowed Abraham et al. to eliminate the use of \(\Delta\) in their upper bounds.

Furthermore, the new definition allowed Abraham et al. to prove a strong bound for TRANSIT NODE ROUTING that was not possible under the old definition (regardless of \(\Delta\)).
3 Graph Construction

3.1 Definition of $G_{t,k,q}$

The family of graphs $G_{t,k,q}$ was designed to show a lower bound on the number of shortcuts created during the preprocessing stage of contraction hierarchies \[12\].

Intuitively, imagine an set of $q$ identical, complete $t$-ary trees of height $k$ with two types of additional edges. Inside every tree, there is an edge between any node and each of its ancestors. Among the trees, each node has an edge to every one of its 'copies' in other trees.

The rigorous definition of $G_{t,k,q}$ is as follows. Consider a complete $t$-ary tree of height $k$ for integers $t, k \geq 2$. Let $\lambda(v)$ denote the height of node $v$, and let $\lambda(u,v)$ denote the height of the lowest common ancestor between two nodes $u$ and $v$.

Now define the edges as follows: for all nodes $v$ and $w$ such that $w$ is a proper ancestor of $v$, there is an edge between $v$ and $w$ with length $16^{\lambda(v)-1}$. This means the edge length from a node $v$ to one of its descendants $v$ is independent of $\lambda(v)$. Furthermore, edge lengths increase for nodes higher up in the tree.

Denote this graph by $G_{t,k} = (V_{t,k}, E_{t,k})$. For convenience, we will still refer to this graph as a tree, even though the extra types of edges create cycles.

Now we will define $G_{t,k,q} = (V_{t,k,q}, E_{t,k,q})$ by taking $q$ copies of $G_{t,k}$, and naming them $G_{t,k}^{(a)} = (V_{t,k}^{(a)}, E_{t,k}^{(a)})$ for $a = 1, 2,..., q$. The copy of a node $v \in G_{t,k}$ is denoted $v^{(a)}$ when it is in $G_{t,k}^{(a)}$.

For all nodes $v \in G_{t,k}$, we add additional edges $v^{(a)} - v^{(b)}$ to $E_{t,k,q}$ for all $a \neq b$. The length of $v^{(a)} - v^{(b)}$ is $2^{\lambda(v) - k - 1}$. This ensures that switching copies has a low penalty ($2^{\lambda(v) - k - 1}$ is always less than 1), and it is always cheaper to switch among copies lower down in the tree.

3.2 Properties of $G_{t,k,q}$

We will now discuss the properties of $G_{t,k,q}$. The following lemmas are proven in \[12\].

Lemma 1. Given $s, t \in V_{t,k}$ with lowest common ancestor $w$, the unique shortest $s$–$t$ path is $s$–$w$–$t$.

Lemma 2. Given $s^{(a)}$ and $t^{(b)}$ in $G_{t,k,q}$, let $w$ be the lowest common ancestor between $s$ and $t$. Then the shortest $s^{(a)}$–$t^{(b)}$ paths are:

$s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)}$, if $\lambda(s) \leq \lambda(t)$, and/or
$s^{(a)} - w^{(a)} - t^{(a)} - t^{(b)}$, if $\lambda(t) \leq \lambda(s)$.

Lemma 3. The highway dimension $h$ of $G_{t,k,q}$ is equal to $q$, the diameter $D$ is $\Theta(16^k)$, and $|V_{t,k,q}| = \Theta(qt^k)$.

It is worth noting that at the start we assumed graphs have unique shortest paths, but now many shortest paths in our main family of graphs are not unique. However, this is a common assumption in routing algorithm proofs because it is not hard to perturb the input to make all shortest paths unique while maintaining the validity of the proofs.

Additionally, integrality of edge lengths is violated. Since the smallest edge is $2^{-k}$ (and all edge lengths are multiples of this), all of the edge weights should technically be multiplied by $2^k$. This will increase log $D$ by $k$, doubling it, which will not affect our results.

4 Hub Labeling

The hub labeling algorithm was first devised in 2004 by Gavoille et al. \[10\], and further studied by Cohen et al. \[7\]. However, the algorithm was not practical for continental routing queries until
2011, when Abraham et al. came up with an efficient way to perform the preprocessing and query phases, which made it the fastest routing algorithm to date [2].

**Hub Labeling** relies on the concept of labeling. Each node stores information about its shortest paths that allows us to reconstruct the shortest path during a query. This idea is used in a clever way to make queries run exceptionally fast.

In the **Hub Labeling** algorithm, we give each node \( v \in V \) a label consisting of other nodes (the **hubs** of \( v \)), and we store the shortest distances to the hubs from \( v \). We define a **labeling** \( L \) as the set of labels \( L(v) \) for all \( v \in V \).

We construct the labeling in such a way that for any pair of nodes \( s \) and \( t \), \( L(s) \cap L(t) \) contains at least one node on the shortest path from \( s \) to \( t \). When satisfied, this is called the **cover property**. Then in order to perform an \( s-t \) query, we only need to find the \( v \in L(s) \cap L(t) \) that minimizes \( \text{dist}(s,v) + \text{dist}(v,t) \). This takes \( O(|L(s)| + |L(t)|) \) time because we can choose some node ordering and make it consistent among all labels. Then to find the common node, we merely traverse the two labels simultaneously. This process returns \( \text{dist}(s,t) \). To return the nodes on this shortest path, we need to add another data structure in the preprocessing stage, which does not increase the space complexity by more than a constant factor. Then the shortest path can be found in time linear in the number of nodes on the path [2].

As we have said, the query time of this algorithm depends linearly on the size of the labels. It is very hard (suspected to be NP-hard) to find the labeling that minimizes total label sizes. Therefore, in practice we must rely on heuristics in the preprocessing stage.

Abraham et al. showed that the query time of **Hub Labeling** is \( O(h \log D) \), using a specific labeling that they created. The proof requires defining a new concept, multiscale shortest-path covers, so we refer the reader to [1]. The proof did not use any properties of \( \hat{h} \) that are different from \( h \), so we can also say that the query time is \( O(h \log D) \).

It is not known how to construct the labeling used in their proof in polynomial time, so they showed a corollary that uses a polynomial time preprocessing algorithm and permits queries to be handled in \( O(h \log h \log D) \) time.

### 4.1 Lower Bound

We cannot prove a lower bound on the minimum query time, since labelings can be constructed to make any one query run in constant time. Instead, we will prove a bound on the average query time by bounding the sum of all label sizes.

**Theorem 4.** For all \( h, D, n \), there is a graph \( G = (V,E) \) with highway dimension \( h \), diameter \( \Theta(D) \), and \( |V| \geq n \), such that for any choice of labeling \( L \), the average query requires \( \Omega(h \log D) \) time.

**Proof.** We will show that \( G_{t,k,q} \) satisfies the desired requirements, with \( t, k, \) and \( q \) to be defined at the end of the proof.

Consider different classes of shortest paths between pairs of leaves distinguished by the height of their lowest common ancestor as follows.

For \( 0 \leq i \leq k \), let \( P_i = \{ s-t \mid s \text{ and } t \text{ are leaves, and } \lambda(s,t) = i \} \).

Let \( \sum_{v \in V} |L(v)| = H \). Our goal is to show that a constant fraction of the \( k+1 \) sets \( P_0, P_1, ..., P_k \) each contribute \( \Omega(q^2t^k) \) distinct nodes to the sum \( H \).

We make the assumption that all the neighbors of a leaf \( v^{(a)} \), and the leaf itself, are in that leaf’s label. That is, \( L(v^{(a)}) \) contains \( v^{(b)} \) for all \( b \) (even when \( b = a \)), and contains \( w^{(a)} \) for all ancestors \( w \) of \( v \). These are \( k+q+1 \) nodes per leaf and \( t^k(k+q+1) \) total nodes, which is asymptotically...
less than $\Omega(t^k q^2 k)$, the desired result. Therefore, this assumption will not affect the validity of our proof.

Now consider an arbitrary path in $P_i$. Label the endpoints of the shortest path $P_i$ by $s^{(a)}$ and $t^{(b)}$. From Lemma 2, $P_i$ must equal $s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)}$, where $w$ is the lowest common ancestor of $s$ and $t$, and $\lambda(w) = i$.

$L(s^{(a)}) \cap L(t^{(b)})$ must contain at least one of $s^{(a)}$, $w^{(b)}$, $t^{(b)}$ in order to satisfy the cover property. By our assumption above, $s^{(a)}, w^{(b)}, t^{(b)} \in L(s^{(a)})$ and $w^{(b)}, t^{(b)} \in L(t^{(b)})$. Now there are four cases.

Case 1: $s^{(a)} \in L(t^{(b)})$. Note that $s^{(a)}$ is not on any other shortest path starting at $t^{(b)}$.

Case 2: $t^{(b)} \in L(s^{(a)})$. Again, $t^{(b)}$ is not on any other shortest path starting at $s^{(a)}$.

Case 3: $w^{(b)} \in L(s^{(a)})$. $w^{(b)}$ is on all leaf-leaf shortest paths (that end at $t^{(b)}$) of the form $s^{(c)} - s^{(b)} - w^{(b)} - t^{(b)}$ for $c \neq b$. There are $q - 1$ such paths in $P_i$.

Case 4: $w^{(b)} \in L(s^{(a)})$. $w^{(b)}$ is on all leaf-leaf shortest paths (that start at $s^{(a)}$) of the form $s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)}$ for $v$ such that $\lambda(s, v) = i$. There are $t^i - t^{i-1}$ such paths, since there are $t^i$ leaves with $w^{(b)}$ as an ancestor, and all but $t^{i-1}$ of those leaves have $w^{(b)}$ as the lowest height ancestor to get to $s^{(a)}$.

Furthermore, 

$$|P_i| = \binom{q}{2} t^k (t^i - t^{i-1}) = \frac{q(q-1)t^k(t^i - t^{i-1})}{2}$$

because there are $\binom{q}{2}$ ways to pick two copies of trees, $t^k$ choices for the first leaf, and $t^i - t^{i-1}$ choices for the second leaf (in order to guarantee that the leaves have a lowest common ancestor of height $i$).

So if we assume $t^i - t^{i-1} \geq q - 1$ (we will explain in the next paragraph why we can make this assumption), then we can achieve a lower bound on the number of labels needed for $P_i$ by exclusively using Case 4 for our choice of labels.

$$\frac{q(q-1)t^k(t^i - t^{i-1})}{2} \div (t^i - t^{i-1}) = \frac{q(q-1)t^k}{2}.$$

Therefore, the contribution of $P_i$ to the sum $H$ is at least $\frac{q(q-1)t^k}{2}$. For all $i$, the hubs that $P_i$ contributes to the sum $H$ have height $i$, ensuring that a node does not get double counted in $H$.

Let $k = \lceil \log D \rceil$, $q = h$, and pick $t$ big enough such that $qt^{k+1} \geq n$ (ensuring that $|V| \geq n$) and $t^{k/2} \geq q$ (ensuring that at least half of the $P_i$'s satisfy $t^i - t^{i-1} \geq q - 1$).

Then the highway dimension of $G$ is $h$ and the diameter is $\Theta(D)$. Recall that $|V| \in \Theta(qt^k)$. Then for any given labeling $L$,

$$\sum_{v \in V} |L(v)| \geq \frac{k}{2} \cdot \frac{q(q-1)t^k}{2} \in \Omega(h|V| \log D).$$

This completes the proof since query times depend on the size of the labels. \qed

With this theorem, we see that the upper bound presented in [1] is tight; there is no better upper bound.

5 Contraction Hierarchies

CONTRACTION HIERARCHIES [11] is a shortcut-based algorithm, making it fundamentally different from HUB LABELING and TRANSIT NODE ROUTING. CONTRACTION HIERARCHIES is both simpler
and faster than highway hierarchies \cite{13}, its predecessor. It works by ordering the nodes based on importance with respect to shortest paths, and then pruning the queries using that order.

In the preprocessing stage for contraction hierarchies, we iteratively contract nodes using a predefined ordering, called a contraction ordering. The contraction operation called on \( v \) first deletes \( v \) from the graph, and then adds edges between its neighbors to preserve all shortest path lengths. For any two neighbors \( s \) and \( t \) of \( v \), we add a new edge \( s - t \) with weight \( \text{dist}(s, v) + \text{dist}(v, t) \) if \( s - v - t \) is the only shortest \( s - t \) path. We contract every node in the graph based on the ordering, and we are left with the set \( E^+ \) of shortcut edges.

To run an \( s - t \) query, run bidirectional search (two alternating Dijkstra searches starting at \( s \) and \( t \), and stopping when the searches intersect) on the graph \( G^+ = (V, E \cup E^+) \), except at node \( v \), only consider edges \( v - w \) in which \( w \) has a higher rank in the ordering than \( v \). When there are no more nodes to consider in either direction, find the node \( v \) that minimizes the sum of its distances to \( s \) and to \( t \).

In \cite{11}, it is proven that \( v \) is guaranteed to be on the shortest path between \( s \) and \( t \), which means that \( \text{dist}(s, t) = \text{dist}(s, v) + \text{dist}(v, t) \). The query returns the shortest \( s - t \) path in \( E \cup E^+ \); the shortest path in \( E \) can be found in time linear in the length of the path by keeping track of which nodes were contracted to make each shortcut.

When choosing a node ordering, we wish to minimize \( |E^+| \). Any ordering will give correct queries, but making \( |E^+| \) small will decrease the time and space requirements for the algorithm. Although finding the optimal ordering is NP-hard \cite{6}, there are fast heuristics that make \( |E^+| \) within \( \log h \) of optimal \cite{11}.

Abraham et al. showed that queries run in \( O((\hat{h}\log D)^2) \) time, which involves proving that \( |E^+| \in O(n\hat{h}\log D) \). The latter bound was proven tight by using \( G_{t,k,q} \) in \cite{12}. However, the proof assumes the contraction order from the algorithm in Abraham et al. which is thought to be NP-hard to compute. In the appendix, we show a new proof of the same lower bound generalized to any contraction order.

Using the older definition of highway dimension, Abraham et al. achieved a result that depends on \( \Delta \): \( O((\Delta + h\log D)(h\log D)) \) \cite{3}.

The upper bounds again depend on multiscale shortest-path covers which are hard to compute. If a polynomial time preprocessing algorithm is required, the bounds are modified to \( O((\hat{h}\log \hat{h}\log D)^2) \) and \( O((\Delta + h\log h\log D)(h\log h\log D)) \).

### 5.1 Lower Bound

If we order the nodes in \( G_{t,k,q} \) with Abraham et al.’s contraction ordering, then every node of height \( i \) will be contracted before all nodes of height \( j \) where \( i < j \). The ordering among nodes of the same height is arbitrary.

A criterion for exactly what shortest paths are shortcut for this ordering was presented in \cite{12}: the path \( s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)} \) is shortcut if and only if \( a \neq b \), \( w \) is a proper ancestor of \( s \), and \( s^{(b)} \) is contracted before \( s^{(a)} \).

Our strategy will be to find a lower bound for the height-based ordering using this criterion, and then generalize it to an arbitrary ordering.

**Theorem 5.** For all \( h, D, n \), there is a graph \( G = (V, E) \) with highway dimension \( h \), diameter \( \Theta(D) \), and \( |V| \geq n \) such that the average query time is \( \Omega((h\log D)^2) \) for contraction hierarchies.

**Proof.**
Again we will show $G_{t,k,q}$ satisfies the properties, defining $t, k, q$ at the end of the proof. Consider a query between two leaves $s^{(a)}$ and $t^{(b)}$ such that $\lambda(s, t) = k$ and $a \neq b$. This type of query makes up a constant fraction of all queries, so we will limit our analysis to this case. A regular Dijkstra search settles $s^{(a)}$ and all copies of $s$, and then it settles the parent of $s^{(a)}$ and all its copies, and continues to settle the successive ancestors of $s^{(a)}$ along with their copies. A total of $q(k + 1)$ nodes are settled in this way. The backwards search goes through a similar process starting at $t^{(b)}$. For contraction hierarchies, each node only needs to look at neighbors with a higher contraction order than itself. If we are using an adjacency list to represent the graph, this can be done by reordering the adjacency list based on contraction order.

Assume initially that we are using Abraham et al.’s contraction ordering, which orders nodes by height from the bottom up (we will remove this assumption shortly). So in the forward search, the only nodes we may visit are ancestors of $s$ in the tree. Among the nodes in $S$ with height $i$, let $T_i$ contain the $\frac{q}{2}^i$ nodes with lower contraction order than the other $\frac{q}{2}^i$ nodes in that layer. Let

$$T = \bigcup_{i=0}^{k/2} T_i.$$  

Suppose $v^{(c)}$ is one of the $\frac{q}{4}^k$ nodes in $T$. Recall that the shortcut criterion for Abraham et al.’s ordering says the path $v^{(d)} \rightarrow v^{(c)} \rightarrow u^{(c)}$ (where $u$ is an ancestor of $v$) will be shortcut if $v^{(c)}$ is contracted before $v^{(d)}$. Then contracting $v^{(c)}$ will create at least $\frac{qk}{2}$ shortcuts, since $v^{(c)}$ is in the bottom half of the tree and has a lower contraction order than half of the nodes in $v$ (in other copies). Therefore, the forward search will need to look through at least $\frac{q^2k^2}{16}$ nodes, making the average query take $\Omega(q^2k^2)$ time.

Now we will consider a general ordering by examining the effects of contracting an arbitrary node $v^{(a)}$ on edges in $G^{(a)}_{t,k}$.

If $v^{(a)}$ is contracted before a descendant $u^{(a)}$, shortcuts from $u$ in any copy to $v^{(a)}$ will never be created. The number of queries this affects is based on the height of $u$. If $u$ is a leaf, it only affects queries starting from $u$, but if $u$ is higher up in the tree, it will affect all queries starting at leaves with $u$ as an ancestor. In effect, we need to weight the nodes based on their importance. We do this using $\sum_{u^{(a)}} t^{\lambda(u)}$, where $u^{(a)}$ is a descendant of $v^{(a)}$ with contraction order higher than $v^{(a)}$. The value of this sum is proportional to the loss in total query time when contracting $v^{(a)}$ compared to Abraham et al.’s ordering. Let $\lambda(v) = i$. If all of $v^{(a)}$’s descendants were contracted before $v^{(a)}$, the sum would be $\sum_{j=0}^{i-1} t^{i-j} t^j = it^i$ because each layer can contribute at most $t^i$ to the sum. There are two cases to consider.

Case 1: $\sum_{u^{(a)}} t^{\lambda(u)} \leq \frac{1}{2}it^i$. In this case, the average query time decreases by at most a factor of two, which doesn’t affect our big-Omega bound.

Case 2: $\sum_{u^{(a)}} t^{\lambda(u)} > \frac{1}{2}it^i$. The number of edges in $G^{(a)}_{t,k}$ that are lost from $v^{(a)}$’s contraction is $\leq t^i$, the number of $v^{(a)}$’s descendants. However, contracting $v^{(a)}$ before many of its descendants will create many leaf-leaf shortcuts.

The smallest possible set of contracted descendants would contain the $\geq t^{i/2}$ nodes in the top $\frac{i}{2}$ layers below $v^{(a)}$.

Given two of these nodes $x^{(a)}$ and $y^{(a)}$ with $\lambda(x, y) = \lambda(v)$, a shortcut will be created between $x^{(a)}$ and $y^{(a)}$. Half of the subtrees rooted at $v^{(a)}$’s children will have half of their nodes with contraction order higher than $v^{(a)}$, so we will gain at least $\frac{(t/2)^2}{2^i} = \frac{t^{i-1}(t/2-1)}{16} \in \Omega(t^i)$ extra shortcuts this way.
Therefore, the number of edges decreases by at most a constant factor, which does not affect our big-Omega bound.

In both cases, we maintain the $\Omega(q^2k^2)$ bound even with an arbitrary ordering.

Now let $k = \lceil \log D \rceil$ and $q = h$, and we pick $t$ big enough such that $qt^k \geq n$. Then the average query for contraction hierarchies is $\Omega((h \log D)^2)$.

\section{Transit Node Routing}

Transit node routing \cite{4} was devised in 2007 by Bast et al., and it (and variants) remain the second-fastest family of routing algorithms, behind hub labeling \cite{5}. However, transit node routing requires about an order of magnitude less space than hub labeling.

The algorithm works by picking a set $T \subset V$ of transit nodes that hit as many long-distance shortest paths as possible. $|T|$ is often chosen to be in $\Theta(\sqrt{|V|})$, which makes the algorithm run fastest while maintaining that additional memory requirements are bounded by the input graph size. Usually, the contraction order is used to pick $T$ (since contraction order essentially seeks to measure a node’s importance with respect to shortest paths), which works very well in practice.

Next, given any node $v$, $A(v) \subset T$ is the set of that node’s access nodes, which are chosen to hit the long-distance queries stemming from $v$. This usually means that we want to pick nodes in $T$ that are close to $v$.

The distances between all pairs of transit nodes are computed and stored, as well as the distances between a node $v$ and each of its access nodes. A query is called a global query if $\min(\text{dist}(s,u)+\text{dist}(u,v)+\text{dist}(v,t) \mid u \in A(s), v \in A(t)) = \text{dist}(s,t)$. Otherwise, it is a local query. To run an $s$–$t$ query, first run a quick locality filter that determines whether the query is local or global. Locality filters are historically based on the coordinates of the vertices. If it is a global query, calculate the minimum $\text{dist}(s,u)+\text{dist}(u,v)+\text{dist}(v,t)$ by trying all combinations of access nodes from $A(s)$ and $A(t)$. Local queries are handled by a fast local search such as contraction hierarchies.

Abraham et al. use a choice of $T$ based on multiscale shortest-path covers to prove that access nodes are bounded in size by $O(\hat{h})$, from which it follows that global queries can be handled in $O(\hat{h}^2)$ time. Local queries done using contraction hierarchies can be handled in $O((\hat{h} \log \hat{h})^2)$ time as we saw in the previous section (however, local queries tend to be small, making the queries run much faster than the average contraction hierarchies query).

This bound is not possible without the new definition of highway dimension. Again, if we want polynomial time preprocessing, the query time bound for global queries increases to $O(\hat{h} \log \hat{h})^2)$.

\subsection{Lower Bound}

While the upper bound for transit node routing was for global queries only, our lower bound will include both local and global searches. We will use contraction hierarchies for local queries.

Call a leaf-leaf shortest path regular if the shortest path is global and neither endpoint is a transit node. We would like to exclude irregular shortest paths from our analysis.

First, we show that queries with a transit node as an endpoint do not make up a constant fraction of all queries. Since $|T| \leq \sqrt{|V|}$, the number of shortest paths in which at least one endpoint is a transit node is $O(|V|\sqrt{|V|}) \in o(|V|^2)$.

Next, we consider the case in which local queries make up at least a $\frac{1}{10}$ fraction of total queries. In the previous section, we showed in Theorem 3 that the average query for contraction hierarchies requires $\Omega((h \log D)^2)$ time. The proof showed a constant fraction of all queries required
We added another factor of $\frac{1}{16}$ of total queries, a constant fraction of those queries require $\Omega((h \log D)^2)$ time (thus a constant fraction of all queries require that amount of time). This Omega bound is higher than the one we seek to prove for global queries, so for the rest of our analysis we can assume that a $< \frac{1}{16}$ fraction of total queries are irregular.

**Lemma 6.** If the number of local leaf-leaf queries is $a < \frac{1}{16}$ fraction of total queries, then there is a set of $\frac{q}{2}$ copies in which there are $\frac{k}{2}$ leaves that are each an endpoint of $\frac{k}{2}$ regular shortest paths going to at least $\frac{q}{2}$ different copies.

**Proof.** Assume the number of local leaf-leaf queries is $o(|V|^2)$, but assume the lemma is false. Then there must be $\geq \frac{q}{2}$ copies with the following property: $\geq \frac{k}{2}$ leaves are each endpoints of $\leq \frac{k}{2}$ regular shortest paths going to $\geq \frac{q}{2}$ copies.

Now consider the maximum number of regular leaf-leaf shortest paths possible in $G_{t,k,q}$ under that assumption. Making all four inequalities tight, we have $\frac{q}{2}$ copies with $\frac{k}{2}$ leaves each as endpoints of $\frac{k}{2}$ regular shortest paths going to $\frac{q}{2}$ copies each. In other words, in half of the copies, half of the leaves each have the property that in half of the copies, half of the shortest paths going from that leaf to the copy are regular. This means that at the very least $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{16}$ of all leaf-leaf shortest paths must not be regular.

This violates one of our assumptions, so we have a contradiction.

Now we have the machinery necessary to prove a lower bound on the query time of transit node routing. We will use $G_{t,k,q}$, and our strategy will be to only consider regular shortest paths, and show the maximum number of shortest paths an access node can hit is bounded below. This will lower bound the average query time.

**Theorem 7.** For all $h$, $D$, $n$, there is a graph $G = (V,E)$ with highway dimension $h$, diameter $\Theta(D)$, and $|V| \geq n$ such that for any choice of transit nodes $T$ and access nodes $A$, the average query time is $\Omega(h^2)$.

**Proof.** We will show that $G_{t,k,q}$ has the desired properties, with the values of $t$, $k$, and $q$ to be defined at the end of the proof.

From our previous argument at the start of this subsection, we need only consider the case where $< \frac{1}{16}$ of all queries are irregular.

We use Lemma 6 to define a set $S$ of regular shortest paths such that there are exactly $\frac{q}{2}$ copies that have exactly $\frac{k}{2}$ leaves with exactly $\frac{k}{2}$ regular shortest paths in $S$ going to $\frac{q}{2}$ copies.

Then,

$$|S| \geq \frac{1}{2} \cdot \frac{q}{2} \cdot \frac{k}{2} \cdot \frac{k}{2} = \frac{q^2k^2}{32}.$$  

We added another factor of $\frac{1}{2}$ because these shortest paths can be double counted.

Given a path $P \in S$, $P$’s endpoints are two leaves $s^{(a)}$ and $t^{(b)}$ in different copies and must be of the form $s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)}$ or $s^{(a)} - w^{(a)} - t^{(a)} - t^{(b)}$ by Lemma 2. Without loss of generality, assume that $P$ is $s^{(a)} - s^{(b)} - w^{(b)} - t^{(b)}$. Since the path is global, $s^{(a)}$ must have an access node on $P$. The access node can’t be $s^{(a)}$ itself since $P$ is regular. Therefore, the access node must be in $G^{(b)}_{t,k}$.

This access node hits at most $\frac{k}{2}$ paths in $S$ stemming from $s^{(a)}$ because that is the total number of shortest paths in $S$ from $s^{(a)}$ to a leaf in $G^{(b)}_{t,k}$.
So given an arbitrary path in \( S \), we have shown that an access node for some node \( v^{(a)} \) must exist that can hit at most \( t^k \) other shortest paths in \( S \). Then the total number of access nodes needed in \( S \) is at the very least

\[
\frac{q^2 t^{2k}}{32} \div \frac{t^k}{2} = \frac{q^2 t^k}{16} \in \Omega(q|V|).
\]

As in earlier proofs, we let \( k = \lceil \log D \rceil / 4 \) and \( q = h \), and we pick \( t \) such that \( qt^{k+1} \geq n \). Then \( G \) has highway dimension \( h \), diameter \( \Theta(D) \), and has \( |V| \geq n \).

Queries in which both endpoints’ access node sets are \( \Omega(h) \) will take \( \Omega(h^2) \) time, and these make up a constant fraction of all global queries.

\[\square\]

7 Conclusions and Future Work

We have proven lower bounds on the query time of hub labeling, contraction hierarchies, and transit node routing. The proofs are all quite different, despite using the same family of graphs for each proof.

Although we have theoretically proven lower bounds for these algorithms, they will likely not be realized on real-world graphs. Real inputs empirically have other properties that make them well-suited to perform better with these algorithms than the theory would predict. One example is that real-world graphs have small separators \[8\]. A future direction could be to incorporate this fact to try to prove better upper and lower bounds on recent routing algorithms (or to come up with a new routing algorithm that works better).

Another way to work with more realistic road networks is to use the idea of multiscale dispersed graphs, defined in \[9\], as a new model for graphs that simulate real-world graphs. One may be able to obtain better bounds on the query time with this model.

Throughout this paper, we assumed undirected graphs, so future work could extend these results to the directed case. Furthermore, apart from hub labeling, the upper and lower bounds are not tight because of the different definitions of highway dimension. Ideally, we would find a way to prove the lower bounds using the more recent definition of highway dimension. However, we cannot use \( G_{t,k,q} \) for this task. Under the new definition, \( G_{t,k,q} \) has highway dimension at least \( q + k \), since the new definition guarantees a graph’s degree is bounded by its highway dimension.

Acknowledgements

All of the results in this paper are from the senior honors thesis of the author, written under the direction of Prof. Lyle McGeoch, at Amherst College. We would like to thank Lyle McGeoch for helpful discussions throughout the writing process. We are grateful for the Post-Baccalaureate Summer Research Fellowship program at Amherst College, which supported the writing of this paper.

References

[1] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V Goldberg, and Renato F Werneck. Highway dimension and provably efficient shortest path algorithms. Technical Report MSR-TR-2013-91, Microsoft Research, 2013.
[2] Ittai Abraham, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. A hub-based labeling algorithm for shortest paths on road networks. In Proceedings of the 10th International Symposium on Experimental Algorithms (SEA ’11), volume 6630 of Lecture notes in Computer Science, pages 230–241. Springer, 2011.

[3] Ittai Abraham, Amos Fiat, Andrew V Goldberg, and Renato F Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA ’10), pages 782–793. SIAM, 2010.

[4] Julian Arz, Dennis Luxen, and Peter Sanders. Transit node routing reconsidered. In Proceedings of the 12th International Symposium on Experimental Algorithms (SEA ’13), volume 7933 of Lecture Notes in Computer Science, pages 55–66. Springer, 2013.

[5] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato Werneck. Route planning in transportation networks. Technical report, Microsoft Research, 2014.

[6] Reinhard Bauer, Tobias Columbus, Bastian Katz, Marcus Krug, and Dorothea Wagner. Preprocessing speed-up techniques is hard. In Proceedings of the 7th Conference on Algorithms and Complexity (CIAC’10), volume 6078 of Lecture notes in Computer Science, pages 359–370. Springer, 2010.

[7] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. SIAM Journal on Computing, 32(5):1338–1355, 2003.

[8] Daniel Delling, Andrew V Goldberg, Ilya Razenshteyn, and Renato F Werneck. Graph partitioning with natural cuts. In 25th International Parallel & Distributed Processing Symposium (IPDPS’11), pages 1135–1146. IEEE Computer Society, 2011.

[9] David Eppstein and Michael T Goodrich. Studying (non-planar) road networks through an algorithmic lens. In Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS’08), pages 1–10. ACM Press, 2008.

[10] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA ’01, pages 210–219, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[11] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. Transportation Science, 46(3):388–404, 2012.

[12] Nikola Milosavljević. On optimal preprocessing for contraction hierarchies. In Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science, pages 33–38. ACM Press, 2012.

[13] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In Algorithms-ESA 2006, pages 804–816. Springer, 2006.
Appendix

The following theorem gives a lower bound on $|E^+|$, independent of the node ordering. This generalizes the result in [12], which proved this theorem for the ordering based on multiscale shortest-path covers.

**Theorem 8.** For all $h$, $D$, $n$, there is a graph $G = (V, E)$ with highway dimension $h$, diameter $\Theta(D)$, and $|V| \geq n$ such that for any ordering, $|E^+| \in \Omega(h|V| \log D)$.

**Proof.** We will show $G_{t,k,q}$ satisfies the desired requirements, setting the values of $t$, $k$, $q$ at the end of the proof.

We will be concerned only with shortcuts added when contracting leaves. We will first count the number of shortcuts added by contracting all of the leaves first, as in the preprocessing algorithm by Abraham et al. Recall the criterion for creating a shortcut in this ordering, which was stated in Section 5.1. A path $s^{(a)} \rightarrow s^{(b)} \rightarrow w^{(b)} \rightarrow t^{(b)}$ is shortcut if and only if $a \neq b$, $w$ is a proper ancestor of $s$, and $s^{(b)}$ is contracted before $s^{(a)}$. Then the number of shortcuts added when contracting all of the leaves is $S = t^k(k)\frac{q}{2} \in \Theta(qk|V|)$ since there are $t^k$ ways of picking a leaf, $k$ ways of picking a proper ancestor, and $\frac{q}{2}$ ways of picking two copies.

In general, the number of shortcuts created for leaf $v^{(a)}$ at the time of its contraction is the number of ancestors $v^{(a)}$ has in $G^{(a)}_{t,k}$ multiplied by the number of copies $v^{(b)}$, $b \neq a$, in other trees. We will now consider the effects of arbitrary contraction order on the number of edges a leaf has in its own copy at its time of contraction.

Given an arbitrary contraction order $\theta$ and a non-leaf $v$, let $c_i$, $1 \leq i \leq t$, be the number of leaves with contraction order higher than $v$ in the subtree with $v$’s $i$th child as a root. Then $0 \leq c_i \leq t^{\lambda(v)-1}$ for all $i$.

Contracting $v$ causes $\sum_{i=1}^{k} c_i$ leaf descendants of $v$ to lose one edge each. However, contracting $w$ also increases the number of leaf-leaf edges by $\sum_{i \neq j} c_i c_j$.

Then the net edge gain for contracting $v$ instead of all leaves first is

$$A_{v,\theta} = \sum_{i \neq j} c_i c_j - \sum_{i=1}^{t} c_i.$$  

In order to find the minimum value of $A_{v,\theta}$, we consider four cases.

Case 1: $\geq 3$ $c_i$’s are nonzero. Without loss of generality, let the $c_i$’s make a decreasing sequence. So $c_1 \geq c_2 \geq \cdots \geq c_t \geq 0$ and $c_3 \geq 1$. Then $c_1 c_3 \geq c_1$, $c_1 c_2 \geq c_2$, $c_2 c_3 \geq c_3$, ..., $c_{t-1} c_t \geq c_t$. It follows that

$$A_{v,\theta} = \sum_{i \neq j} c_i c_j - \sum_{i=1}^{t} c_i \geq 0.$$  

Case 2: Exactly two $c_i$’s are nonzero. So $c_1 \geq c_2 \geq 1$ and $c_3 = c_4 = \cdots = c_t = 0$. Then $A_{w,\theta} = c_1 c_2 - c_1 - c_2 = (c_1 - 1)(c_2 - 1) - 1$. If $c_2 > 1$, then $(c_1 - 1) \geq (c_2 - 1) \geq 1$, so $A_{v,\theta} \geq 0$. If $c_2 = 1$, then $c_1 - 1 = 0$, so $A_{v,\theta} = -1$.

Case 3: Exactly one $c_i$ is nonzero. So $c_1 \geq 1$ and $c_2 = c_3 = \cdots = c_t = 0$. Then $A_{v,\theta} = -c_i$, so the minimum value of $A_{v,\theta}$ in this case is $-t^{\lambda(v)-1}$.

Case 4: All $c_i$’s are zero. Then clearly $A_{v,\theta} = 0$.

Therefore, the minimum value of $A_v$ is $-t^{\lambda(v)-1}$ from case 3.

Note that the possible leaf-leaf edges we gain from contracting a non-leaf $v$ are independent of other leaf-leaf edges we gain from contracting another non-leaf $w$: if $\lambda(v) = \lambda(u)$, the leaves in the edges must be different since they cannot have both $v$ and $u$ as an ancestor. If $\lambda(v) \neq \lambda(u)$, the
edges must be different since the lowest common ancestors between the endpoints of each edge are at different heights.

So given an arbitrary contraction order, the number of leaf-edges within a copy $G_{t,k}^{(a)}$ (at the time of the leaf’s contraction) is

$$kt^k - \sum_{v \in G_{t,k}^{(a)}} A_v, \theta \geq kt^k - \sum_{i=1}^{k} t^{k-i} t^{i-1} = kt^k - \sum_{i=1}^{k} t^{k-1} = kt^k - kt^{k-1} = kt^{k-1}(t-1).$$

Then

$$|E^+| \geq \left( \frac{q}{2} \right) kt^{-1}(t-1) \in \Omega(kq|V|)$$

We let $k = \lceil \frac{\log D}{4} \rceil$ and $q = h$, and we pick $t$ such that $qt^{k+1} \geq n$. Then $G$ has highway dimension $h$, diameter $\Theta(D)$, and has $|V| \geq n$. Finally, given a contraction order $\theta$, $|E^+| \in \Omega(h|V| \log D)$. □

It turns out that the maximum number of lost edges (case 3) is achievable for every single node in the graph. This preprocessing order is surprisingly more optimal than the shortest path cover ordering.

Although it is related, this theorem does not overlap with Theorem 5. We used the leaves to compute a lower bound on $|E^+|$, but the leaves do not make up a constant fraction of nodes settled during a typical query.