Karlsruhe Institute of Technology

On the Complexity of Contraction Hierarchies

Student Thesis of

Tobias Columbus

At the faculty of Computer Science
Institute for Theoretical Informatics (ITI)

Reviewer: Prof. Dr. Dorothea Wagner
Advisor: Dipl.-Math. Reinhard Bauer

29. June 2009 – 29. December 2009

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und die vorgestellten Ergebnisse ohne die Hilfe Dritter erarbeitet habe. Ich habe auf keine anderen als die angegebenen Quellen und Hilfsmittel zurückgegriffen.

Tobias Columbus
Landau, den 29. Dezember 2009

Abstract

Contraction Hierarchies are an experimentally well studied method to efficiently handle point-to-point shortest-path queries in large graphs. In a preprocessing stage the vertices of the input graph are iteratively removed, while the shortest-path structure of the remaining graph is preserved. This results in a hierarchy of graphs and shortest-path queries can be answered by only considering edges that are leading to vertices of smaller graphs in the hierarchy. The order of removal of the vertices has great impact on performance, but by now only heuristics are known for determining such orders. In this thesis Contraction Hierarchies are studied from a theoretical point of view. It is shown that approximating contraction orders resulting in minimal preprocessing size is **APX**-hard and the corresponding decision problem is proven to **NP**-complete. Furthermore, lower bounds for the size of search space in Contraction Hierarchies of paths and trees are given. Finally an integer linear program is used to experimentally investigate optimal contraction hierarchies on small random graphs.

Contents

1. Introduction

The problem of finding a shortest path between two vertices in a graph is a well-known problem in computer science. The classic solution from 1959 for graphs with non-negative edge-lengths is Dijsktra's algorithm [Dij59]. In fact, Dijkstra's algorithm computes not only a single shortest path, but the shortest paths from one source vertex to all other vertices in the input graph. This renders Dijkstra's algorithm quite inefficient for applications, where one is interested in only a single shortest path between the given vertices, as there may be many vertices that are processed during Dijkstra's algorithm but are irrelevant for the problem at hand. This inefficiency is negligible, if the instances are rather small or if there are no tight requirements on the runtime, but it becomes a crucial factor when one deals with large inputs, such as road networks of Europe or North America.

In order to address this problem many algorithms, so called *speedup techniques* for Dijkstra's algorithm have been proposed in recent years. There are Reach [GKW06], Highway Hierarchies [SS05], Highway Node Routing [SS07], Arc-Flags [HKMS09], SHARC [BD09], ALT [GH05] and Contraction Hierarchies [GSSD08], the latter of which is the subject of this thesis. Furthermore some of these techniques may be combined to achieve even better results. An overview and experimental studies may be found for example in [WW07] and [HSW04]. Generally, these speedup techniques allow to answer shortest-path-queries in the aforementioned road-networks in mere milliseconds.

Most of these techniques use some preprocessing stage to augment the graph with additional information that can be used to speed up point-to-point shortest-path queries. This preprocessing often contains some degree of freedom that is currently filled heuristically. To date there are no non-trivial guarantees for these heuristics and no approximation or fixed-parameter tractable algorithms known. Moreover there is very little theoretical work on speedup techniques in general. There are no known bounds on the searchspace size and until recently the complexity status of optimal preprocessing for all these techniques was not known.

This thesis settles these questions at least partially for Contraction Hierarchies. In Chapter 3 finding contraction hierarchies with a minimal number of edges is shown to be **APX**-hard and the associated decision problem is proven to be **NP**-complete. We further show that it is **NP**-hard to approximate this problem within $\frac{7}{6} - \varepsilon$ for all $\varepsilon > 0$. Unless $\mathbf{P} = \mathbf{NP}$, these results render exact polynomial-time algorithms and polynomial-time approximation schemes for this problem impossible. In Chapter 4, contraction hierarchies with minimum average searchspace are considered. We prove tight lower bounds for the searchspace size of contraction hierarchies of paths and provide a simple algorithm to compute these hierarchies. These lower bounds are then generalized to trees. Moreover we give a characterization of average searchspace size in acyclic graphs, which seems to be an interesting problem on its own. Finally, in Chapter 5, an integer linear program for computing optimal contraction hierarchies is presented and some experimental results using that linear program are given.

2. Preliminaries

Notational Conventions

$\mathbf{Z}$	the set of integers
$\mathbf{N}$	the set non-negative integers
$\mathbf{R}$	the set of real numbers
$\mathbf{R}^+$	the set of positive, real numbers
$\log x$	logarithm of x to base 2
$\|A\|$	the cardinality of the set A
$A \subset B$	A is a subset of B, not necessarily proper
$A - B$	set difference of A and B
order on A	a reflexive, transitive, anti-symmetric and total binary relation on the set A

Graphs

A *graph* $G = (V, E)$ consists of a set V of *vertices* and a set $E \subset V \times V$ of *edges*. We say G is *undirected*, if $(u, v) \in E$ implies $(v, u) \in E$. In this case we identify each edge (u, v) with its inverse edge (v, u) and perceive the edges $e \in E$ as two-element subsets of V. The number of vertices and (undirected) edges is always denoted by n and m respectively. A subgraph of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subset V$ and $E' \subset E$. If $G = (V, E)$ is a graph and $V' \subset V$, then $G - V'$ is the maximal subgraph of G with vertices $V - V'$.

A *path* p between two vertices $u, v \in V$ is a finite sequence $(x_0, \ldots, x_r)$ of vertices, such that $x_0 = u, x_r = v$ and $(x_{i-1}, x_i) \in E$ for all $1 \leq i \leq r$. The *hoplength* $\mathrm{hop}(p)$ is the number of edges on p, i.e. $\mathrm{hop}(p) = r$. A path p from u to v is *hop-minimal* if there is no other path p' from u to v with $\mathrm{hop}(p') < \mathrm{hop}(p)$. If all vertices x_i on a path p are pairwise distinct, p is called *simple*. A *cycle* is a path $p = (u, \ldots, v)$ such that $u = v$ and $\mathrm{hop}(p) > 0$. A *simple cycle* is a cycle $p = (u, \ldots, u)$ where all vertices except u are pairwise distinct.

A directed graph G is *acyclic*, if there are no simple cycles in G. An undirected graph G is *acyclic* if there are no simple cycles of hoplength greater than 2 in G. If $G = (V, E)$ is a directed, acyclic graph, a *topological order* on V is an order $\prec$ on V such that for each edge $(u, v) \in E$ it is $u \prec v$. By the *diameter* Δ_G of a graph G we mean the maximum hoplength of hop-minimal paths in G, i.e. $\Delta_G = \max\{\mathrm{hop}(p) : p \text{ is a hop-minimal path}\}$. A graph $G = (V, E)$ is *connected* if for every two vertices $u, v \in V$ there is a path $p = (u, \ldots, v)$ or $p = (v, \ldots, u)$. A vertex is *separating* in a connected graph $G = (V, E)$ if $G - \{v\}$ is not connected. If a graph G is not connected the *components* of G are the maximal connected subgraphs of G.

A *weighted graph* $G = (V, E, \text{len})$ is a graph $G = (V, E)$ together with *edge-lengths* $\text{len}\colon E \to \mathbf{R}^+$. If G is undirected, we further require that $\text{len}(\{u, v\})$ is well-defined, i.e. $\text{len}((u, v)) = \text{len}((v, u))$. For the sake of brevity we mostly write $\text{len}(u, v)$ instead of $\text{len}((u, v))$ or $\text{len}(\{u, v\})$. The *length* $\text{len}(p)$ of a path $p = (x_0, \ldots, x_r)$ in a weighted graph is the sum of the lengths of all edges in p, i.e. $\text{len}(p) = \sum_{i=1}^{r} \text{len}(x_{i-1}, x_i)$. The *distance* $\text{dist}_G(u, v)$ of two vertices in a weighted graph G is the length of a shortest path between u and v, i.e. $\text{dist}_G(u, v) = \min\{\text{len}(p) : p \text{ is a path between } u \text{ and } v\}$. If there is no path from u to v we let $\text{dist}_G(u, v) = \infty$. Instead of $\text{dist}_G(u, v)$ we also write $\text{dist}(u, v)$ if it is clear from the context which graph G is meant.

Dijktsra's Algorithm

Given a weighted graph $G = (V, E, \text{len})$ and a *source vertex* $s \in V$, Dijkstra's algorithm computes the distances $\text{dist}(s, u)$ from s to all vertices $u \in V$. The algorithm distinguishes between *unvisited*, *visited* and *settled* vertices. Initially all vertices $u \neq s$ are unvisited and the tentative distance is set to $\text{dist}(s, u) = \infty$. The source vertex s is marked as visited and $\text{dist}(s, s)$ is 0. In each step of the algorithm a visited vertex u with minimal tentative distance is marked as settled and all its out-edges are *relaxed*. Relaxing an edge (u, v) thereby means setting the tentative distance of v to $\min\{\text{dist}(s, v), \text{dist}(s, u) + \text{len}(u, v)\}$ and marking v as visited if it was unvisited before. The algorithm stops as soon as all vertices are either settled or unvisited. A proof of correctness may be found in [Dij59].

It is furthermore possible to compute not only the distances $\text{dist}(s, u)$ but to explicitly output the associated shortest paths. In order to achieve this, one stores for each vertex u the predecessor of u on the shortest path from s to u. Initially all vertices $u \neq s$ are unvisited and their predecessor is set to be undefined ($\perp$). From this information the actual shortest paths are easily reconstructible. Algorithm 1 is an implementation of Dijkstra's Algorithm in pseudocode. Using Fibbonacci Heaps [FT87] to find the visited vertex with the minimum tentative distance currently gives the best known theoretical worst-case runtime of $O(m + n \log n)$ for Dijkstra's algorithm on graphs with arbitrary positive edge-lengths. For special edge-weights better bounds are known. If for example all edge-weights are integral and bounded by a constant $C \geq 0$, an implementation using Radix Heaps has worst-case runtime $O(m + n \log C)$ [AMOT90].

Contraction Hierarchies

Contraction Hierarchies are a speed-up technique for Dijkstra's algorithm that use a preprocessing stage. For simplicity we consider only contractions hierarchies of undirected graphs. Note that this does not impose any restrictions on the results, as one may view undirected graphs as directed graphs with all edges being symmetric.

Let $G = (V, E, \text{len})$ be a graph and $\prec$ an order on V. The preprocessing stage of Contraction Hierarchies iteratively *contracts* the $\prec$-least vertex until G is empty. Contraction of a vertex v thereby works as follows: For each unique shortest path $p = (u, v, w)$ a new edge $\{u, w\}$ of length $\text{len}(p)$, called a *shortcut* is introduced to G. Afterwards v and its incident edges are removed from G and contraction continues with the remaining graph. The output of the preprocessing stage is a directed, acyclic, weighted graph $H = (V, A, \text{len})$, where A contains the edges and shortcuts of G directed from the $\prec$-smaller to the $\prec$-larger vertex. More formally $A = \{(u, v) : u \prec v \text{ and } \{u, v\} \text{ is an edge or a shortcut of } G\}$. The edge-lengths in H are the edge-lengths in G. We call the graph H the *contraction hierarchy* of G corresponding to the order $\prec$ and denote it by $\mathcal{H}(G, \prec)$. Algorithm 2 provides an abstract pseudocode implementation of this process and in Figure 2.1 an example for a graph and its contraction hierarchy is given.

Algorithm 1: DIJKSTRA($G = (V, E, \text{len}), s$)

Input : Weighted graph $G = (V, E, \text{len})$ and a source vertex $s \in V$
Output: Distances $\text{dist}(s, u)$ for all vertices $u \in V$
and table $\text{parent}(v)$ encoding the shortest paths from s to v

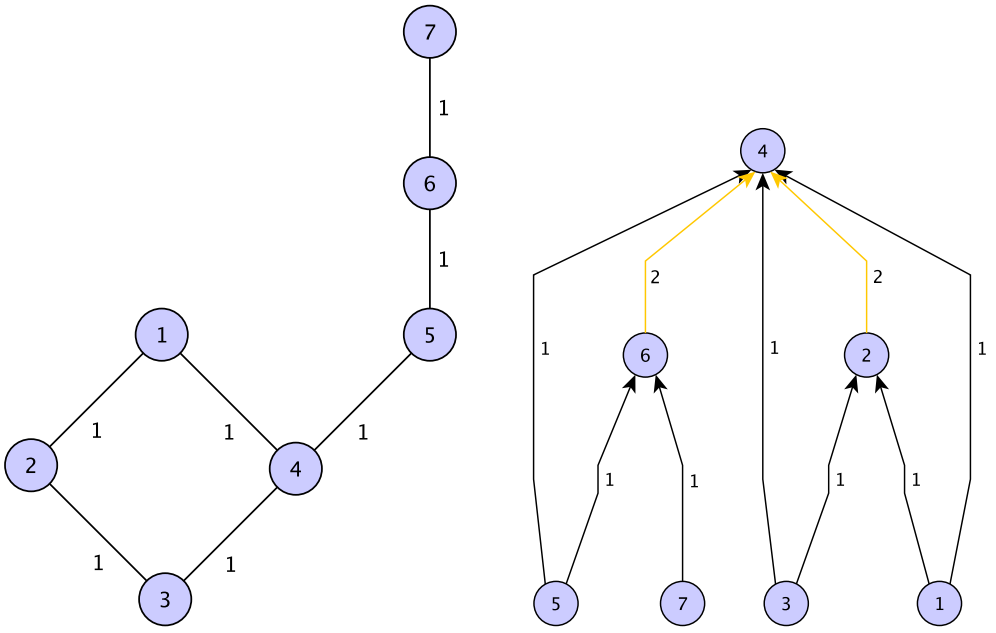
foreach $u \in V - \{s\}$ **do**
  $\text{dist}(s, u) = \infty$
  $\text{mark}(u) = \text{unvisited}$
  $\text{parent}(u) = \perp$
end
$\text{dist}(s, s) = 0$
$\text{mark}(s) = \text{visited}$
$\text{parent}(s) = s$

while $\exists u \in V$ *such that* $\text{mark}(u) = \text{visited}$ **do**
  Choose $u \in V$ with $\text{mark}(u) = \text{visited}$ and minimal $\text{dist}(s, u)$
  foreach $(u, v) \in E$ **do**
    if $\text{dist}(s, u) + \text{len}(u, v) < \text{dist}(s, v)$ **then**
      $\text{dist}(s, v) = \text{dist}(s, u) + \text{len}(u, v)$
      $\text{mark}(v) = \text{visited}$
      $\text{parent}(v) = u$
    end
  end
end

Algorithm 2: CONTRACTIONHIERARCHY($G = (V, E, \text{len}), \prec$)

Input : Weighted graph $G = (V, E, \text{len})$ and an order $\prec$ on V
Output: Contraction hierarchy $\mathcal{H}(G, \prec) = (V, A, \text{len})$

while $G \neq \emptyset$ **do**
  Choose $\prec$-least vertex $v \in V$
  foreach *unique shortest path* $p = (u, v, w)$ **do**
    $E = E \cup \{u, w\}$
    $\text{len}(u, w) = \text{len}(p)$
  end
  $G = G - \{v\}$
end
$A = \{(u, v) : u \prec v \text{ and } \{u, v\} \in E\}$



(a) A weighted graph G

(b) The contraction hierarchy $\mathcal{H}(G, \prec)$, where $u \prec v$ if u is drawn below v.

Figure 2.1: An example of a contraction hierarchy

To answer shortest-path queries in G, two Dijkstra's searches in $\mathcal{H}(G, \prec)$ are run from the source and target vertex respectively. For simplicity we consider no special stopping criterion, i.e. the algorithm stops as soon as there are only unvisited and settled vertices in both directions. There is experimental evidence, that the searchspace size of more sophisticated queries on contraction hierarchies only differs by a constant factor from this simple query algorithm. For a detailed description of contraction hierarchies and more sophisticated queries the reader may refer to [GSSD08], where directed graphs are treated as well and a proof of correctness is provided.

Complexity Classes and Reductions

If Σ is a finite alphabet, then a *decision problem* A over Σ is a subset $A \subset \Sigma^*$, where Σ^* is the set of all finite strings over Σ. The complexity classes of deterministic and non-deterministic polynomial-time acceptable decision problems are denoted by **P** and **NP** respectively. If A and B are two decision problems over alphabets Σ and Π, a *reduction* from A to B is a polynomial-time many-one reduction, i.e. a polynomial-time computable function $f\colon \Sigma^* \to \Pi^*$, where $f(x) \in B$ if and only if $x \in A$. If there exists such a reduction, we also say A *reduces to* B.

A problem A is called **NP**-*hard*, if every problem $B \in$ **NP** reduces to A. If additionally $A \in$ **NP**, A is called **NP**-*complete*. Polynomial-time many-one reductions compose, i.e. if f and g are reductions from A to B and from B to C respectively, then $g \circ f$ is a reduction from A to C. In order to prove the problem A to be **NP**-hard, it therefore suffices to give a reduction from a problem, which is already known to be **NP**-hard. We state decision problems A informally as a set of inputs together with a yes/no question and assume that A is the set of inputs with answer "yes". The reader may refer to [GJ79] for an overview of the theory of **NP**-completeness.

The concept of decision problems is not expressive enough to study approximate solutions, thus we further investigate *optimization problems*. An optimization problem A over an alphabet Σ is given by a set $I_A \subset \Sigma^*$ of *inputs*, a set $F_A(x) \subset \Sigma^*$ of *feasible solutions* for every input $x \in I_A$ and a *cost function* $c_A \colon I_A \times F_A(\cdot) \to \mathbf{N}$, that assigns to each pair $(x, y) \in I_A \times F_A(x)$ a cost value $c_A(x, y)$. In this thesis only *minimization problems* are considered, i.e. problems where, given some $x \in I_A$, one wants to compute a $y \in F_A(x)$ that minimizes the cost $c_A(x, y)$. An optimization problem A is said to be an **NP** optimization problem or **NPO** problem for short, if $I_A \in \mathbf{P}$, c_A is computable in polynomial-time and if the size of feasible solutions $y \in F_A(x)$ is bounded above by a polynomial in the size of x. An algorithm $\mathcal{A}$ is said to *approximate* a minimization problem A within a constant $\rho \geq 1$, if for all $x \in I_A$, $\mathcal{A}$ computes a $y \in F_A(x)$ such that $c_A(x, y)/\operatorname{opt}(x) \leq \rho$, where $\operatorname{opt}(x) = \min\{c_A(x, y) : y \in F_A(x)\}$. In this case $\mathcal{A}$ is called a *ρ-approximation* and A is said to be *approximable within ρ*. By **APX** we denote the class of all **NPO** problems that are approximable within some constant ρ. A *polynomial-time approximation scheme* (PTAS) for an **NPO** problem A is an algorithm $\mathcal{A}$ that takes as input a pair $(x, \varepsilon) \in I_A \times \mathbf{R}^+$ and approximates A within ε in time polynomial in the size of x.

If A and B are two **NPO** minimization problems, a *P-reduction* from A to B consists of three functions f, g and e subject to the following constraints:

(i) $f \colon I_A \to I_B$ and $g \colon I_B \times F_B(\cdot) \to F_A(\cdot)$ are polynomial-time computable.

(ii) $e \colon \mathbf{R}^+ \to \mathbf{R}^+$ is a computable function.

(iii) For all $x \in I_A$ and all $y \in F_B(f(x))$ it is

$$\frac{c_B(f(x), y)}{\operatorname{opt}(f(x))} \leq e(\varepsilon) \Rightarrow \frac{c_A(x, g(x, y))}{\operatorname{opt}(x)} \leq \varepsilon$$

If there is a P-reduction from A to B we also say that *A P-reduces to B*. It is easy to see that if A P-reduces to B and there is a PTAS for B, then there is also a PTAS for A. A problem A is called **APX**-hard, if all problems $B \in \mathbf{APX}$ P-reduce to A. As it was the case with polynomial-time many-one reductions, P-reductions compose and to prove A **APX**-hard, it thus suffices to give a P-reduction from a problem already known to be **APX**-hard. An in-depth study of **NPO** problems and approximibality can be found in [Kan92].

3. Preprocessing Size

Speedup techniques for Dijkstra's algorithm are, as already mentioned in the introduction, an attempt to make fast shortest-path queries in large graphs feasible. In route planning, for example, one has to deal with graphs that represent the road networks of whole continents. Naturally these graphs have millions of vertices and edges and it is desirable that the preprocessing stages of speedup techniques keep the amount of data as small as possible, for otherwise mobile devices with limited ressources would not be capable of handling the preprocessed graphs.

Therefore it is a natural question to ask for contraction hierarchies with a minimal number of shortcuts, as those are the only additional data that accumulates in the preprocessing stage of this speedup technique. In this chapter we thus deal with the following decision problem.

Problem (CH Preprocessing Size)
Input: A weighted graph $G = (V, E, \text{len})$ and a number $K \in \mathbf{N}$.

Question: Is there an order $\prec$ on V, such that $\mathcal{H}(G, \prec)$ contains at most K shortcuts?

We refer to this problem as **CHPS** and denote the associated optimization problem by $\mathbf{CHPS}_{\text{opt}}$, i.e.

Problem (Optimal CH Preprocessing Size)
Input: A weighted graph $G = (V, E, \text{len})$.

Solutions: Orders $\prec$ on V.

Cost: The number of shortcuts in $\mathcal{H}(G, \prec)$.

In what follows, it will be shown that for arbitrary graphs **CHPS** is **NP**-complete and $\mathbf{CHPS}_{\text{opt}}$ is **APX**-hard. Before we turn to the proof, we consider some easy special cases where **CHPS** and $\mathbf{CHPS}_{\text{opt}}$ can be solved in polynomial time. These considerations already give a hint where the difficulty in solving **CHPS** lies.

3.1 Restricted Graph Classes and Examples

Lemma 1: If $G = (V, E)$ is a directed, acyclic graph and $\prec$ a topological order on V, then there are no shortcuts in $\mathcal{H}(G, \prec)$.

Proof
Assume for the sake of contradiction that (u, v) is a shortcut in $\mathcal{H}(G, \prec)$. Then there is a vertex w, whose contraction resulted in the insertion of that shortcut, i.e. $w \prec u$, $w \prec v$ and there are edges (u, w) and (w, v) in G. This contradicts the assumption that $\prec$ is a topological order on V. □

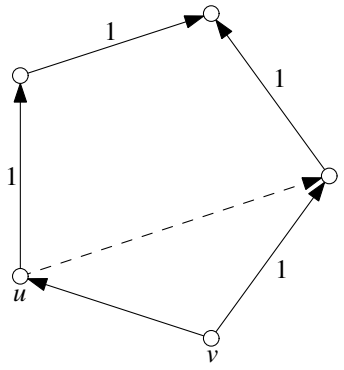


Figure 3.1: An optimal solution of $\mathbf{CHPS}_{\text{opt}}$ for a pentagon, where the dashed shortcut is only present, if $\text{len}(v, u) < 2$.

Corollary: If $G = (V, E)$ is a directed, acyclic graph, then $\mathbf{CHPS}_{\text{opt}}$ can be solved in time $O(n+m)$.

Proof
A topological order $\prec$ can be computed in time $O(n + m)$, as described in [Tar76]. □

A statement similar to Lemma 1 also holds for acyclic, undirected graphs. The crucial property of $\prec$ in the above proof is that there are no three vertices u, v, w in G with $w \prec u$, $w \prec v$ and edges (u, w) and (w, v). In the undirected case this property is resembled by the notion of bottom-up orders. We call an order $\prec$ on the vertices of a rooted tree T a *bottom-up order*, if whenever u is a child of v in T, then it follows that $u \prec v$.

Lemma 2: If $T = (V, E)$ is a rooted tree and $\prec$ a bottom-up order on V, then there are no shortcuts in $\mathcal{H}(T, \prec)$.

Proof
Suppose (u, v) is a shortcut in $\mathcal{H}(G, \prec)$. Then there is a vertex w, such that $w \prec u$, $w \prec v$ and there are edges $\{u, w\}$ and $\{v, w\}$ in T. In particular this implies that w is a child vertex of u and v, which is a contradiction. □

Corollary: If $G = (V, E)$ is an undirected, acyclic graph, then $\mathbf{CHPS}_{\text{opt}}$ can be solved in time $O(n)$.

Proof
An undirected, acyclic graph G is a disjoint union of unrooted trees. A bottom-up order on V may be computed by depth-first search with arbitrary roots in time $O(n)$. □

The preceeding two lemmas show that – regardless of edge-lengths and concrete structure – graphs without cycles always admit contraction hierarchies with no shortcuts at all and simple, linear-time algorithms to obtain such optimal contraction hierarchies were provided. This already suggests that the claimed hardness of **CHPS** has its source in the difficulty to find optimal contraction orders on cycles and especially intersecting cycles. In contrast to acyclic graphs, the insertion of shortcuts in cyclic graphs furthermore depends largely on the edge-lengths, as one can see in Figure 3.1.

3.2 Hardness Results

This whole paragraph is devoted to the proof of the following two theorems.

Theorem 1: **CHPS** is **NP**-complete.

Theorem 2: $\mathbf{CHPS}_{\text{opt}}$ is **APX**-hard and for all $\varepsilon > 0$ it is **NP**-hard to approximate $\mathbf{CHPS}_{\text{opt}}$ within $\frac{7}{6} - \varepsilon$.

We will first concentrate on proving Theorem 1 and it will emerge afterwards, that actually both results follow from the same reduction. A first, simple step in proving Theorem 1 is to assure that **CHPS** $\in$ **NP**.

Lemma 3: **CHPS** $\in$ **NP**.

Proof
A contraction hierarchy corresponding to a given order $\prec$ can be computed in polynomial time by Algorithm 2. Furthermore it can be checked in polynomial time, if the number of shortcuts in that hierarchy is less or equal than some constant K. □

For Theorem 1 it remains to show that **CHPS** is **NP**-hard. To this end we give a reduction from **Vertex Cover**:

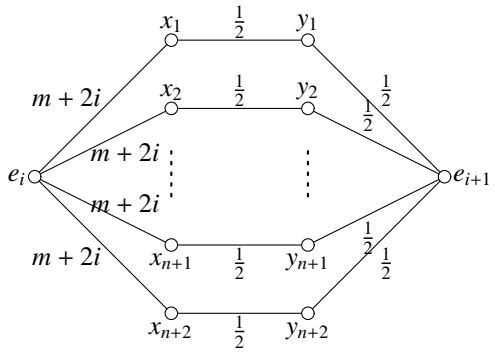
Problem (Vertex Cover)
Input: An undirected graph $G = (V, E)$ and an integer $K \in \mathbf{N}$.

Question: Is there a vertex cover $C' \subset V$ of G, of at most K nodes, i.e. a subset $C' \subset V$ such that $|C'| \leq K$ and for all edges $\{u, v\} \in E$, $u \in C'$ or $v \in C'$?

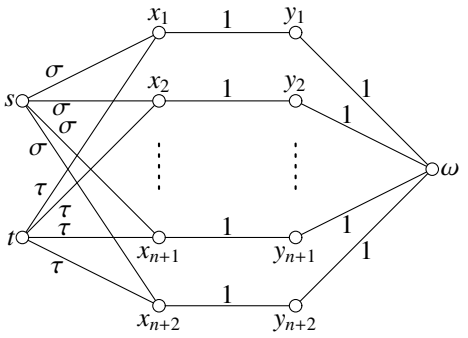
Throughout this paragraph (G, K) denotes an instance of **Vertex Cover**, where $G = (V, E)$ is an undirected graph with $n = |V|$ vertices and $m = |E|$ edges and where K is a natural number, less or equal than n. Given this instance (G, K), we construct a weighted graph $G' = (V', E')$, which admits a contraction hierarchy H with at most K shortcuts, if and only if G has a vertex cover of size at most K.

The set $V \cup E$ is a subset of the vertices V' of G'. The vertices $E \subset V'$ are henceforth referred to as *edge-vertices*. For each such edge-vertex $e = \{u, v\} \in V'$ the graph G' contains two edges $\{e, u\} \in E'$ and $\{e, v\} \in E'$. This construction can be thought of as putting one vertex on each edge e of G. Furthermore V' contains two special vertices $s, t \in V'$, where s is connected to all edge-vertices $e \in E$ and t is connected to all vertices $v \in V$. That is $\{\{s, e\} : e \in E\} \subset E'$ and $\{\{t, v\} : v \in V\} \subset E'$.

Now we fix an arbitrary order $e_1, \ldots, e_m$ on the edges of G and connect each e_i to e_{i+1} by a *honeycomb gadget* H_i that enforces the contraction order $e_i \prec e_{i+1}$. The gadget H_i can be seen in Figure 3.2a. Additionally there is a *final gadget* F connecting s and t, which is depicted in Figure 3.2b. Finally we have to choose the edge-lengths in G'. We let $\text{len}(t, v) = \frac{1}{2}m$, $\text{len}(e_i, v) = 2m$ and $\text{len}(s, e_i) = m + i$ for edge-vertices e_i and vertices $v \in V$. The edge-lengths in the gadgets are chosen according to Figure 3.2a and Figure 3.2b. The whole construction is summarized in Figure 3.3. Note that G' can be computed in polynomial time and is independent of K.



(a) The honeycomb gadget H_i enforcing contraction order $e_i \prec e_{i+1}$.



(b) The final gadget F connecting s and t. The weights σ and τ are chosen as $\sigma = 5m + \frac{1}{8}$ and $\tau = 5m$.

Figure 3.2: The gadgets used in the reduction from **Vertex Cover** to **CHPS**

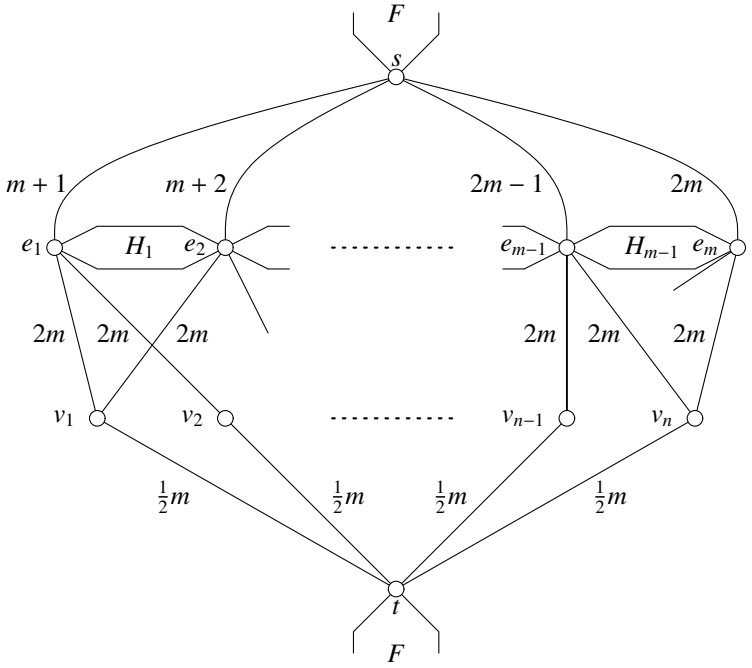


Figure 3.3: Schematic picture of G'. The honeycomb gadgets H_i are depicted by small hexagons between e_i and e_{i+1}. For readability reasons the final gadget F is only shown as half hexagons at s and t.

Lemma 4: If G contains a vertex cover of size K, then G' admits a contraction hierarchy with K shortcuts.

Proof
Let $C \subseteq V$ be a vertex cover with at most K vertices. Consider the following contraction order of the vertices of G':

1. Contract all $v \in V - C$.
   First recall that adjacency of a vertex $v \in V$ and an edge-vertex $e \in E$ in G' is equivalent to v being incident to e in G. Every $v \in V - C$ lies on paths (t, v, e) of length $\frac{5}{2}m$ from t to the edge-vertices e adjacent to v. As C is a vertex cover there is for each such edge-vertex e another $u \in C$, which is also adjacent to e and thus lies on another path (t, u, e) from t to e of equal length.

   Note that paths (e_i, v, e_j) have length $4m$ and thus are no shortest paths as the path (e_i, s, e_j) has length $2m + i + j < 4m$.

   As the considered paths (t, v, e) and (e_i, v, e_j) are the only paths via v and shortcuts are only inserted if necessary, contraction of v does not insert any shortcuts into the hierarchy.

2. Contract all edge-vertices $e \in E$ in the chosen order $e_1, \ldots, e_m$. Note that by contraction of e_i the contraction of the gadget connecting e_i to its successor e_{i+1} is implicitly included.
   Further note that contraction of e_i inserts an additional shortcut $\{a, b\}$ if and only if there is a unique shortest path (a, e_i, b) for some vertices $a, b \in V'$. Thus all paths (a, e_i, b) have to be considered to prove the claim. For this purpose we use the notation from Figure 3.4a.

   a) The path $p = (x_r, e_i, x_s)$ between the vertices x_r and x_s of H_i has length $2m + 4i$ and thus is no shortest path, as the path $(x_r, y_r, e_{i+1}, y_s, x_s)$ has length 2.

   b) The path $p = (x_r, e_i, s)$ has length $2m + 3i$, while the path (x_r, y_r, e_{i+1}, s) has length $m + i + 2$, which is, for all $i \geq 1$, less than $2m + 3i$. Therefore p is no shortest path.

   c) The path $p = (x_r, e_i, v)$ has length $3m + 2i$. Again p is no shortest path, as the path $(x_r, y_r, e_{i+1}, v', t, v)$ has length $3m + 1$, which is less than $3m + 2i$ for all $i \geq 1$.

   d) The path $p = (s, e_i, v)$ has length $3m + i$.

      Note that if e_i is the first edge-vertex adjacent to v in our fixed order $e_1, \ldots, e_m$, then p is a unique shortest path. In this case contraction of e_i results in an edge $\{e_i, v\}$ of length $3m + i$ being inserted into the hierarchy. To see that p is indeed a unique shortest path consider the following paths p' between s and v in G':

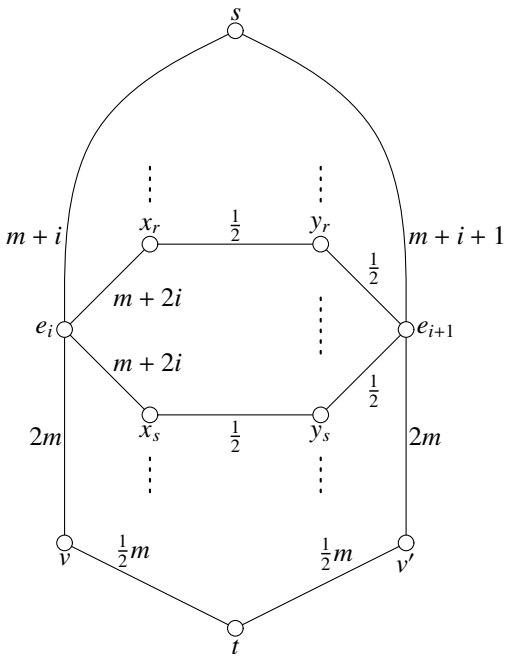
      (i) $p' = (s, e_j, v)$ for some edge-vertex e_j distinct from e_i. Then p' has length $3m + j$, which is greater than $3m + i$ as e_i is the first edge in $e_1, \ldots, e_m$ that is adjacent to v.

      (ii) $p' = (s, e_j, u, t, v)$ for some edge-vertex $e_j \neq e_i$ and some vertex $u \in V$. Then p' has length $4m + j$, which is greater than $3m + i$.

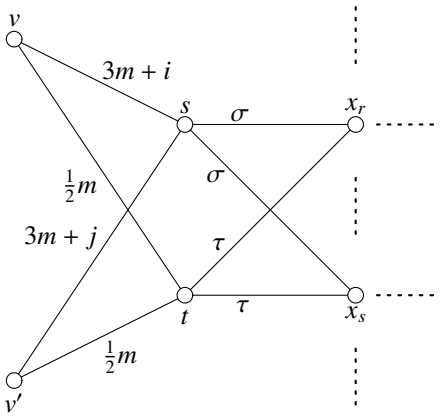
      (iii) $p' = (s, x_r, t, v)$ for some vertex x_r in the final gadget F. Then p' has length at least $10m$, which is greater than $3m + i$, too.

      As these are all relevant types of simple paths between s and v, p is a unique shortest path in G' and thus also in this contraction step.

   After contraction of e_i the remaining part of the gadget H_i consists only of the vertex e_{i+1} and simple paths (x_r, y_r, e_{i+1}). Thus it can be contracted without introducing any shortcuts. The only shortcuts being inserted into the hierarchy are between s and some vertex $v \in C$. The size of C is assumed to be K and thus exactly K shortcuts get inserted.



(a) Contraction of edge-vertex $e_i \in E$.



(b) Contraction of the special vertex s. The edges of weight $m+i$ and $m+j$ originate in the contraction of edge-vertices e_i and e_j. The vertices x_i on the right hand side of the figure are those from the final gadget F. The weights σ and τ were initially chosen as $\sigma = 5m + \frac{1}{8}$ and $\tau = 5m$.

Figure 3.4: Important steps during contraction of G' given a vertex cover $C \subseteq V$

3. Contract the special vertex s.
   Since all vertices $v \in V - C$, all edge-vertices $e \in E$ and all gadgets H_i were already contracted in Step 1 or Step 2, the remaining vertices are $\{s, t\} \cup C$ and the vertices of the final gadget F. The set of shortcuts that were inserted in the preceding steps is $\{\{s, v\} : v \in C\}$, where the edge $\{s, v\}$ has weight $3m + i$, if e_i is the first edge-vertex in the order $e_1, \ldots, e_m$ that is adjacent to v. Altogether we conclude that the remaining graph consists of the final gadget F including the special vertices s and t and the vertices $v \in C$, where each $v \in C$ is connected to s and to t.

   The remaining graph hence looks like the one shown in Figure 3.4b, from which we borrow notation for the following considerations. As one can see in the aforementioned figure, we have to take the following paths into account:

   a) The path $p = (v, s, v')$ between two vertices $v, v' \in C$ has length greater than $6m$. As the path (v, t, v') has length m the path p is clearly no shortest path.

   b) The path $p = (x_r, s, x_s)$ between two vertices x_r and x_s of the final gadget F has length $10m + \frac{1}{4}$, while the path (x_r, t, x_s) has length $10m$. Therefore p is no shortest path.

   c) The path $p = (x_r, s, v)$ has length $8m + i + \frac{1}{8}$. Again, p is no shortest path as the path (x_r, t, v) has length $5m + \frac{1}{2}m$.

   There are no other paths (a, s, b) via s and none of the considered paths is a shortest path. Therefore contraction of s does not insert any additional shortcuts into the hierarchy.

4. Contract all $v \in C$.
   After Step 3 all $v \in C$ have degree one, thus it appears that their contraction does not insert any additional shortcuts into the hierarchy.

5. After Step 4 the remaining graph consists only of the final gadget F without s and its incident edges.

   For each two distinct vertices x_r, x_s in the final gadget F the path (x_r, t, x_s) has length $10m$. Hence it is no shortest path as $(x_r, y_r, \omega, y_s, x_s)$ has length 4. Thus t can be contracted without introducing any additional shortcuts. After contraction of t the remaining part of F is the vertex ω with paths (x_r, y_r, ω) attached to it. This, too, can be contracted without inserting any new shortcuts into the hierarchy.

Overall contraction resulted in at most K shortcuts in the contraction hierarchy H of G'. □

On the other hand we have to assure that there is a vertex cover in G of size at most K, if there is an order $\prec$ on V', such that $\mathcal{H}(G', \prec)$ contains at most K shortcuts. We will first show some simpler properties that the contraction order $\prec$ must possess and then construct a vertex cover in G using these properties and the contraction order.

Lemma 5: Suppose $\prec$ is an order on V', such that $\mathcal{H}(G', \prec)$ contains at most K shortcuts. Then each edge-vertex e_i gets contracted before its successor e_{i+1} in the fixed order $e_1, \dots, e_m$.

Proof
Assume the contrary and consider the honeycomb gadget H_i between e_i and e_{i+1}. Without loss of generality let $(x_1, y_1), \dots, (x_L, y_L)$ be the pairs of vertices (x_r, y_r) in H_i, such that one of x_r, y_r gets contracted after e_{i+1}. Then there are $n+2-L$ pairs $(x_{L+1}, y_{L+1}), \dots, (x_{n+2}, y_{n+2})$, where both x_r and y_r get contracted before e_{i+1}. By assumption e_{i+1} gets contracted before e_i and for $r \geq L+1$ all x_r, y_r therefore get contracted before e_{i+1} and e_i. Now consider the following contraction orders:

1. $y_r \prec x_r, e_i, e_{i+1}$
   The path $p = (x_r, y_r, e_{i+1})$ is a path of length 1 between x_r and e_{i+1}. Apart from p there are the following other, relevant simple paths from x_r to e_{i+1}:

   a) The paths $(x_r, e_i, x_s, y_s, e_{i+1})$, where $s \neq r$, have length $2m + 4i + 1$.

   b) The paths (x_r, e_i, v, e_{i+1}), where v is some vertex $v \in V$ incident to e have length $5m+2i$.

   c) The path (x_r, e_i, s, e_{i+1}) has length $3m + 4i + 1$.

   The path p therefore is a unique shortest path and contraction of y_r before e_i, e_{i+1} and x_r inserts an additional edge $\{x_r, e_{i+1}\}$ into the hierarchy.

2. $x_r \prec y_r, e_i, e_{i+1}$
   The path $p = (e_i, x_r, y_r)$ is a path of length $m + 2i + \frac{1}{2}$ between e_i and y_r. There are the following other paths in G' between e_i and y_r:

   a) The paths $(e_i, x_s, y_s, e_{i+1}, y_r)$, where $s \neq r$, have length $m + 2i + \frac{3}{2}$.

   b) The path (e_i, s, e_{i+1}, y_r) has length $2m + 2i + \frac{3}{2}$.

   c) The path (e_i, v, e_{i+1}, y_r) has length $4m + \frac{1}{2}$.

   Hence, p is a unique shortest path and contraction of x_r before e_i, e_{i+1} and y_r results in an additional edge $\{e_i, y_r\}$.

Therefore contraction of x_r and y_r before e_i and e_{i+1} results in at least one additional edge being inserted into the hierarchy. For there are $n + 2 - L$ pairs (x_r, y_r) that get both contracted before e_i and e_{i+1}, these contractions insert at least $n + 2 - L$ shortcuts into the hierarchy.

Now consider the pairs $(x_1, y_1), \dots, (x_L, y_L)$, where at least one of x_r, y_r gets contracted after e_{i+1}. For $1 \leq s \leq L$ let z_s be the vertex $z_s \in \{x_s, y_s\}$ that is a neighbour of e_{i+1} when e_{i+1} gets contracted. For distinct z_r, z_s the path $p = (z_s, e_{i+1}, z_r)$ has length at most 2, while paths $(z_r, \dots, e_i, \dots, z_s)$ have length at least $m + 2i$, as they include the edge $\{x_r, e_i\}$ of length $m + 2i$ or the shortcut $\{y_r, e_i\}$ of

length $m + 2i + \frac{1}{2}$. Hence p is a unique shortest path and contraction of e_{i+1} before z_s, z_r and e_i inserts an additional shortcut $\{z_r, z_s\}$. As there are $\frac{1}{2}L(L-1)$ such pairs $\{z_r, z_s\}$, contraction of e_{i+1} leads to the insertion of $\frac{1}{2}L(L-1)$ shortcuts.

Altogether, contraction of e_{i+1} before e_i results in at least

$$n + 2 - L + \frac{1}{2}L(L-1) \geq n + 1 > K$$

additional shortcuts being inserted. This is a contradiction and thus e_{i+1} cannot be contracted before e_i. □

Lemma 6: Suppose $\prec$ is an order on V', such that $\mathcal{H}(G', \prec)$ contains at most K shortcuts. Then the special vertices s and t get contracted before the vertex ω in the final gadget F.

Proof
Assume the contrary, i.e. that $\alpha \in \{s, t\}$ gets contracted after ω. Further let $\alpha' \in \{s, t\}$ with $\alpha' \neq \alpha$. Partition the pairs (x_r, y_r) of vertices in F, such that $\omega \prec x_r$ or $\omega \prec y_r$ for all $1 \leq r \leq L$ and such that $x_r, y_r \prec \omega$ for all $L + 1 \leq r \leq n + 2$. Now consider the following contraction orders:

1. $y_r \prec x_r, \omega, \alpha$
   (x_r, y_r, ω) is a unique shortest path of length 2 between x_r and ω. Contraction of y_r before x_r, ω and α therefore inserts an additional edge $\{x_r, \omega\}$ into the hierarchy.
2. $x_r \prec y_r, \omega, \alpha$
   (α, x_r, y_r) is a unique shortest path and thus contraction of x_r before y_r, ω and α results in an additional edge $\{\alpha, y_r\}$.

   Note that (α', x_r, y_r) is also a unique shortest path, but we made no assumptions whether α' is already contracted or not and hence do not consider possible further edge insertions.

Contraction of x_r and y_r before α and ω inserts in any case at least one additional edge into the hierarchy. Since there are $n+2-L$ pairs (x_r, y_r) that get both contracted before ω, these contractions insert at least $n + 2 - L$ additional edges into the hierarchy.

Let z_s for $1 \leq s \leq L$ be the vertex $z_s \in \{x_s, y_s\}$ that is a neighbour of ω when ω gets contracted. For distinct z_s, z_r the path $p = (z_s, \omega, z_r)$ has length at most 4. As any path (z_s, α, z_r) or (z_s, α', z_r) has length at least $10m$, p is a unique shortest path. Contraction of ω before z_s, z_r therefore inserts an additional shortcut $\{z_s, z_r\}$. As there are $\frac{1}{2}L(L-1)$ such pairs $\{z_s, z_r\}$, contraction of ω inserts at least $\frac{1}{2}L(L-1)$ shortcuts.

Altogether, contraction of ω before α results in at least

$$n + 2 - L + \frac{1}{2}L(L-1) > K$$

additional shortcuts being inserted. This is a contradiction and thus $\alpha \prec \omega$. □

Lemma 7: Suppose $\prec$ is an order on V', such that $\mathcal{H}(G', \prec)$ contains at most K shortcuts. Then the special vertex t gets contracted after all $v \in V$.

Proof
Assume the contrary and let $v_0 \in V$ be a vertex with $t \prec v_0$. By Lemma 6 the vertex t gets contracted before ω and thus we may assume that ω is still present when t gets contracted. Consider the final gadget F and partition the pairs (x_r, y_r) of vertices in F, such that $t \prec x_r$ or $t \prec y_r$ for all $1 \leq r \leq L$ and such that $x_r, y_r \prec t$ for all $L \leq r \leq n + 2$. Now consider the following contraction orders:

1. $y_r \prec x_r, t, \omega$
   (x_r, y_r, ω) is a unique shortest path of length 2 between x_r and ω. Contraction of y_r before t, ω and x_r therefore inserts an additional edge $\{x_r, \omega\}$ into the hierarchy.

2. $x_r \prec y_r, t, \omega$
   (t, x_r, y_r) is a unique shortest path of length $5m + 1$ and thus contraction of x_r before t, ω and y_r results in an additional edge $\{t, y_r\}$.

Contraction of x_r and y_r before t hence inserts at least one additional edge into the hierarchy. As there are $n + 2 - L$ pairs (x_r, y_r), such that both x_r and y_r get contracted before t, the contractions of these pairs inserts at least $n + 2 - L$ additional edges into the hierarchy.

For $1 \leq r \leq L$ now let z_r be the vertex $z_r \in \{x_r, y_r\}$ that is adjacent to t, when t gets contracted. Now let p be $p = (v_0, t, x_r)$ if $z_r = x_r$ and $p = (v_0, t, x_r, y_r)$ otherwise. We consider possible shortest paths in G' between v_0 and z_r.

1. If $z_r = x_r$ there is the path $p_r = (v_0, t, x_r)$ and if $z_r = y_r$ there is the path $p'_r = (v_0, t, x_r, y_r)$. The paths p_r and p'_r have length at most $\frac{11}{2}m + 1$.

2. If $z_r = x_r$ there may additionally exist the path $q_r = (v_0, e_i, s, x_r)$ for some edge-vertex e_i and if $z_r = y_r$ there may exist the path $q'_r = (v_0, e_i, s, x_r, y_r)$. The paths q_r and q'_r have length at least $8m + i$ which is greater than $\frac{11}{2}m + 1$.

Hence we may conclude that p has length at most $5m + \frac{1}{2}m + 1$ and is a unique shortest path. Since contraction only preserves distances and does not create new shortest paths, $p = (v_0, t, z_r)$ is a unique shortest path when t gets contracted, too. Contraction of t therefore results in the insertion of an additional shortcut $\{v_0, z_r\}$. As there are L such neighbours z_r of t, contraction of t inserts at least L additional edges.

Altogether contraction of v after t results in $n + 2 - L + L > K$ additional shortcuts, which is a contradiction. □

Lemma 8: Suppose $\prec$ is an order on V', such that $\mathcal{H}(G', \prec)$ contains at most K shortcuts. Then all edge-vertices $e_i \in E$ get contracted before s.

Proof
Assume the contrary, i.e. that there is some edge-vertex $e_i \in E$ that gets contracted after s. Consider the final gadget F and partition the pairs (x_r, y_r) of vertices in F, such that for all $1 \leq r \leq L$ it is $s \prec x_r$ or $s \prec y_r$ and such that for all $L + 1 \leq r \leq n + 2$ it is $x_r, y_r \prec s$. By Lemma 6 s gets contracted before ω and thus $x_r \prec s$ and $y_r \prec s$ imply $x_r \prec \omega$ and $y_r \prec \omega$ respectively. Now consider the following contraction orders.

1. $y_r \prec x_r, s, \omega$
   (x_r, y_r, ω) is a unique shortest path of length 2 between x_r and ω. Contraction of y_r before s, ω and x_r therefore inserts an additional edge $\{x_r, \omega\}$ into the hierarchy.

2. $x_r \prec y_r, s, \omega$
   (s, x_r, y_r) is a unique shortest path of length $5m + \frac{1}{8} + 1$ and thus contraction of x_r before s, ω and y_r results in an additional edge $\{s, y_r\}$.

Contraction of x_r and y_r before s hence inserts at least one additional edge into the hierarchy. As there are $n + 2 - L$ pairs (x_r, y_r), such that both x_r and y_r get contracted before s, the contractions of these pairs inserts at least $n + 2 - L$ additional edges into the hierarchy.

For $1 \leq r \leq L$ let z_r be the vertex $z_r \in \{x_r, y_r\}$ that is adjacent to s, when s gets contracted. The path $p = (e_i, s, z_r)$ has length at most $6m + i + \frac{1}{8} + 1$, while the path $p' = (e_i, u, t, z_r)$ in G', where u is some vertex $u \in V$, has length at least $7m + \frac{1}{2}m$. For p is a unique shortest path in G', p is a unique shortest path, when s gets contracted, too. Contraction of s therefore inserts a shortcut $\{e_i, z_r\}$ into the hierarchy. As there are L such vertices z_r contraction of s results in the insertion of at least L such shortcuts.

Altogether contraction of e_i after s resulted in $n + 2 - L + L > K$ additional shortcuts, which is a contradiction. □

The above lemmas may be summarized by the following two simple observations:

(**I**) E gets contracted in our chosen order $e_1, \dots, e_m$.

(**II**) $V \prec t$ and $E \prec s$, that is whenever we encounter vertices $v \in V$ or edge-vertices $e \in E$, we may assume that the vertex t or the vertex s respectively are not contracted yet.

In the final step of this proof we will construct a vertex cover for the original graph G and prove that it contains K vertices, if the contraction hierarchy of G' contains K shortcuts. Furthermore we will see that the vertex cover may be computed in polynomial time.

Lemma 9: Suppose $\prec$ is an order on V', such that the corresponding contraction hierarchy has K shortcuts. Then there is a vertex cover $C \subset V$ in G of size K, which can be computed in polynomial time.

Proof
For each vertex $v \in V$ that is incident to at least one edge in G let $e_{\min}(v)$ be the first edge-vertex in the fixed order $e_1, \dots, e_m$ that is incident to v, that is $e_{\min}(v) = e_M$, where $M = \min\{i : e_i \text{ is incident to } v\}$. We partition the edges E of G in two disjouint sets E_1 and E_2 as follows:

$$E = \underbrace{\{\{u,v\} \in E : e_{\min}(u) \prec u \text{ or } e_{\min}(v) \prec v\}}_{=E_1} \cup \underbrace{\{\{u,v\} \in E : u \prec e_{\min}(u) \text{ and } v \prec e_{\min}(v)\}}_{=E_2}$$

Note that in the contraction order described in Lemma 4, all edges of G were contained in E_1. The vertices v, such that $e_{\min}(v) \prec v$ were exactly those vertices that were in the given vertex cover $C \subset V$. Now we define for each edge $e \in E$ the *cover vertex* $v(e)$ of e as follows:

$$v(e) = \begin{cases} u \in V \text{ incident to } e \text{ in } G, \text{ such that } e_{\min}(u) \prec u & e \in E_1 \\ \prec\text{-maximal } u \in V \text{ incident to } e \text{ in } G & e \in E_2 \end{cases}$$

Note that, given the order $\prec$ and an edge $e \in E$, it is possible to compute $v(e)$ in polynomial time.

Claim 1: $C = \{v(e) : e \in E\}$ is a vertex cover in G.

Let $e = \{u, v\} \in E$. Then $v(e) = u$ or $v(e) = v$ by definition of the cover vertex $v(e)$.

Claim 2: $C = \{v(e) : e \in E\}$ has size at most K.

As there are at most K shortcuts in the contraction hierarchy, it suffices to show that there is an injective mapping $M\colon C \to S$, where S is the set of shortcuts. We will construct this mapping M on $v(E_1)$ by assigning to each vertex $v = v(e) \in v(E_1)$ a shortcut $\{s, v\}$. Furthermore M on $v(E_2)$ will be given by assigning a shortcut $\{t, e\}$ to each vertex $v = v(e) \in v(E_2)$.

Observe that $v(E_1)$ and $v(E_2)$ are disjoint, as $u \in v(E_1) \cap v(E_2)$ would imply $e_{\min}(u) \prec u \prec e_{\min}(u)$. Since the shortcuts assigned to $v \in v(E_1)$ and $v \in v(E_2)$ are of different kind, it is clear that $M\colon C \to S$ is well-defined and injective on $C = v(E_1) \cup v(E_2)$, if it is well-defined and injective on $v(E_1)$ and $v(E_2)$.

First consider a vertex $v = v(e) \in C$ with $e \in E_1$. Then $e = \{u, v\}$ for some other vertex $u \in V$. The vertex v gets, by definition of E_1, contracted after $e_i = e_{\min}(v)$. As v and e_i are not contracted yet, the vertices s and t are – by Observation (**II**) – also present, when e_i gets contracted. Now consider possible paths between s and v in G'.

1. The path $p = (s, e_i, v)$ has length $3m + i$. For any other e_j the path (s, e_j, v) has length $3m + j$ and since $e_i = e_{\min}(v)$ the path p is a unique shortest path among the paths (s, e_j, v).
2. For some vertex $u \neq v$ and some edge-vertex $e_j \neq e_i$ the path $p = (s, e_j, u, t, v)$ has length $4m + j$, which is greater than $3m + i$.

3. The path $p = (s, x_r, t, v)$, where x_r is a vertex of the final gadget F, has length $10m + \frac{5}{8}$.

As these paths are all relevant paths between s and v in G', p is a unique shortest path in G' between s and v and therefore a unique shortest path, when e_i gets contracted. Thus contraction of e_i inserts an additional edge $\{s, v\}$ into the hierarchy and we let $M(v) = \{s, v\}$. Note that for $v, v' \in v(E_1)$, $M(v) = M(v')$ implies $v = v'$, i.e. M is injective on $v(E_1)$.

Finally we have to account for the vertices $v(e)$ with edges $e \in E_2$. Let u, v be the vertices incident to e in G – or equivalently adjacent to e in G'. By definition of E_2, the vertices u and v get contracted before $e_{\min}(u)$ and $e_{\min}(v)$ respectively. In particular both of them get contracted before e itself. By Observation (**II**) the vertices s and t are not contracted, when u or v get contracted. Consider the paths $p_u = (t, u, e)$ and $p_v = (t, v, e)$, each of length $3m$. Apart from p_u and p_v the only relevant path between t and e is $p' = (t, x_r, s, e)$, where x_r is some vertex of the final gadget F. The length of p' is greater than $10m$ and thus p_u and p_v are shortest paths.

Without loss of generality let $u \prec v$. In particular this implies $v(e) = v$. When v gets contracted, u and the path p_u are already contracted and p_v is a unique shortest path. Contraction of v hence inserts a shortcut $\{t, e\}$ into the hierarchy. In this case we let $M(v) = \{t, e\}$. Observe that M is injective on $v(E_2)$, as $M(v(e)) = M(v(e'))$ implies $e = e'$ and thus $v = v'$.

Furthermore, given the order $\prec$ the set $C = \{v(e) : e \in E\}$ is computable in polynomial time, as each $v(e)$ is computable in polynomial time. □

This finishes the proof of Theorem 1, as G' is constructible from G in polynomial time and Lemma 4 and Lemma 9 imply that G has a vertex cover of size at most K if and only if G' admits a contraction hierarchy with at most K shortcuts. Additionally, there is, by Lemma 9, a polynomial-time computable function g that maps a graph G and an order $\prec$ on V' to a vertex cover C of G, such that if $\mathcal{H}(G', \prec)$ contains K shortcuts, then $|C| \leq K$. Thus it appears that the above reduction is actually a P-reduction from **Vertex Cover** to **CHPS**. Furthermore, for any ρ-approximation $\mathcal{A}$ of **CHPS**, $g(G, \mathcal{A}(G'))$ is a ρ-approximation for **Vertex Cover**. Hence, Theorem 2 follows from the below-mentioned results.

Theorem (Papadimitriou and Yannakakis [PY88]): Vertex Cover on graphs with bounded degree is **APX**-complete.

This theorem implies that **CHPS** is **APX**-hard. Furthermore there is a tight bound on approximation ratios for **Vertex Cover** due to Håstad.

Theorem (Håstad [Hås01]): For any $\varepsilon > 0$ it is **NP**-hard to approximate **Vertex Cover** within $\frac{7}{6} - \varepsilon$.

Theorem 2 implies, that there can be no PTAS for **CHPS** in the general case, unless **P** = **NP**. However, the graphs occuring in applications are often of very special kind and it is not clear whether the lower bound of $\frac{7}{6}$ holds also for these restricted graph classes. In road networks, for example, the vertex degree is bounded by some small constant, but the above reduction works only with vertices of very high degree. In particular there might even exist a PTAS for these restricted cases.

4. Searchspace Size

In this chapter we investigate the searchspace size of contraction hierarchies, i.e. the number of vertices that get settled during a shortest-path query on a contraction hierarchy H. For brevity and simplicity we consider only unidirectional Dijkstra's search, that is only one direction of the bidirectional query in contraction hierarchies. Observe that, if $u \prec v$ and there is a directed path from u to v in H, then v will eventually get settled in Dijkstra's algorithm with source vertex u. As $\text{dist}_H(u, v) = \infty$, if and only if there is no directed path from u to v in H,

$$\overrightarrow{\mathcal{V}_H}(u) = \{v \in V : u \prec v \text{ and } \text{dist}_H(u, v) < \infty\}$$

is the set of settled vertices during Dijkstra's algorithm with source vertex u in the hierarchy H. Consequently $\mathcal{V}_H(u) = |\overrightarrow{\mathcal{V}_H}(u)|$ will be regarded as a measure for the number of settled vertices during that search and

$$\mathcal{V}(H) = \sum_{u \in V} \mathcal{V}_H(u)$$

as a measure for the average searchspace size. If it is clear from the context, which hierarchy H is meant, we also write $\overrightarrow{\mathcal{V}}$ and $\mathcal{V}$ instead of $\overrightarrow{\mathcal{V}_H}$ and $\mathcal{V}_H$. The following lemma is an important observation and the key result for all lower bounds that will be shown below.

Lemma 10: Let $G = (V, E)$ be a connected, undirected graph, $\prec$ an order on V and $H = \mathcal{H}(G, \prec)$. Further let v be $\prec$-maximal and assume v to be separating in G. Denote by $G_i = (V_i, E_i)$ the components of $G - \{v\}$.

(i) Then v is separating in H and for each G_i there is a component H_i of $H - \{v\}$ with vertices V_i.

(ii) If there is only a single edge $\{u, v\} \in E$ such that $u \in V_i$, then $H_i = \mathcal{H}(G_i, \prec)$.

(iii) If all G_i fulfill the requirements of (ii), then

$$\mathcal{V}(H) = n + \sum_{i=1}^{d} \mathcal{V}(H_i)$$

Proof

(i) Assume that there is an edge (x, y) in H, where $x \in V_i$ and $y \in V_j$ for $i \neq j$. We may choose (x, y) such that x is $\prec$-minimal with the property that there are edges $(x, a), (x, b)$ in H with $a \in V_i, b \in V_j, i \neq j$. Note that (x, y) has to be a shortcut, since otherwise x and y are vertices of the same G_i. Hence there is a vertex z, whose contraction led to the insertion of (x, y). That means $z \prec x$, $z \prec y$ and there are edges (z, x) and (z, y) in H. As x was chosen $\prec$-minimal it follows that $(z, x), (z, y)$ have their endpoints in the same V_i, thus x and y belong to the same G_i, which is a contradiction.

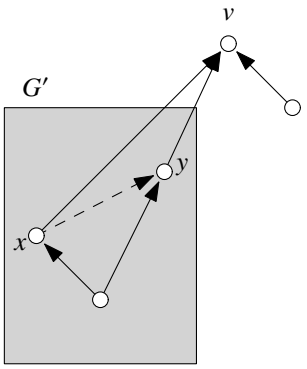


Figure 4.1: A counterexample for Lemma 10 (ii) in the general case. If all edge-lengths are chosen to be 1, then the shortcut (x, y) gets inserted into $\mathcal{H}(G', \prec)$ but not into the hierarchy $\mathcal{H}(G, \prec)$.

(ii) If there is only a single edge $\{u, v\} \in E$ such that $u \in V_i$, then there are no simple paths (u, v, w) with both $u, w \in V_i$. In particular this implies that there is no difference between contraction of G_i as a separate graph and contraction of G_i as a component of G, as there are no paths via v between vertices of G_i. Hence $H_i = \mathcal{H}(G_i, \prec)$.

(iii) Suppose there is a vertex $u \in V$ with $v \notin \overrightarrow{\mathcal{V}_H}(u)$. Without loss of generality let u be the $\prec$-maximal such vertex. Then there are no edges (u, w) in H, as otherwise $v \in \overrightarrow{\mathcal{V}_H}(w) \subset \overrightarrow{\mathcal{V}_H}(u)$. G is connected and hence we may choose a path $p = (u = x_0, \ldots, x_r = v)$ in G. Then p is also contained in H, but $x_i \prec u$ for all $1 \leq i < r$. A simple inductive argument over the hoplength of p now shows that there is a shortcut (u, v) in H, which contradicts $v \notin \overrightarrow{\mathcal{V}_H}(u)$.

In particular, this implies that $\mathcal{V}_H(u) = \mathcal{V}_{H_i}(u) + 1$ for all $u \in V_i$, as H_i is a component of $H - \{v\}$. Let $n_i = |V_i|$, then

$$\begin{aligned}\mathcal{V}(H) &= \mathcal{V}_H(v) + \sum_i \sum_{u \in V_i} \mathcal{V}_{H_1}(u) + 1 \\ &= 1 + \sum_i \mathcal{V}(H_i) + n_i \\ &= n + \sum_i \mathcal{V}(H_i)\end{aligned}$$

which finishes this proof. □

Remark: Statement (ii) in the above lemma is not true in general, as one may see in Figure 4.1.

4.1 Sorting Numbers

Our analysis of average searchspace makes use of *sorting numbers* [Slo09], which are given by the maximum number of comparisons for sorting n elements by binary insertion. In this paragraph we recall the definition and some properties of sorting numbers before we return to Contraction Hierarchies.

Definition 1 (Sorting numbers): For $n \in \mathbf{N}$ the sorting number $B(n)$ is given by

$$B(n) = \sum_{i=1}^{n} \lceil \log i \rceil$$

Lemma 11: Let $n \in \mathbf{N}$, then

$$B(n) = B\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + B\left(\left\lceil \frac{n}{2} \right\rceil\right) + n - 1$$

Proof
We do induction on n.

$n = 1$

$$B(1) = B(0) + B(1) + 1 - 1 = B\left(\left\lfloor \frac{1}{2} \right\rfloor\right) + B\left(\left\lceil \frac{1}{2} \right\rceil\right) + 1 - 1$$

$n > 1$ By induction hypothesis we get

$$B(n) = B(n-1) + \lceil \log n \rceil = B\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) + B\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + \lceil \log n \rceil + n - 2$$

If n is even, then $\left\lfloor \frac{n-1}{2} \right\rfloor = \frac{n}{2} - 1$ and $\left\lceil \frac{n-1}{2} \right\rceil = \frac{n}{2}$. Furthermore $\lceil \log n \rceil = \lceil \log \frac{n}{2} \rceil + 1$ and thus we get

$$\begin{aligned} B(n) &= B\left(\frac{n}{2} - 1\right) + B\left(\frac{n}{2}\right) + \lceil \log n \rceil + n - 2 \\ &= B\left(\frac{n}{2}\right) + B\left(\frac{n}{2}\right) + n - 1 \end{aligned}$$

If n is odd, then $\left\lceil \frac{n-1}{2} \right\rceil = \frac{n-1}{2} = \left\lfloor \frac{n-1}{2} \right\rfloor$. Additionally $\lceil \log n \rceil = \lceil \log \frac{n+1}{2} \rceil + 1$ and we get

$$\begin{aligned} B(n) &= B\left(\frac{n-1}{2}\right) + B\left(\frac{n-1}{2}\right) + \left\lceil \log \frac{n+1}{2} \right\rceil + n - 1 \\ &= B\left(\frac{n-1}{2}\right) + B\left(\frac{n+1}{2}\right) + n - 1 \end{aligned}$$

This finishes the proof. □

Lemma 12: Let $n, n_1, n_2 \in \mathbf{N}$ such that $n_1 + n_2 = n$, then

$$B(n_1) + B(n_2) \geq B\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + B\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

Proof
Without loss of generality let $n_1 \geq n_2$. As $n_1 + n_2 = n$ this implies in particular $n_1 \geq \left\lceil \frac{n}{2} \right\rceil$ and $\left\lfloor \frac{n}{2} \right\rfloor \geq n_2$. Now this lemma follows directly from the monotonicity of log and the definition of sorting numbers:

$$\begin{aligned}
B(n_1) + B(n_2) &= \sum_{i=1}^{n_1} \lceil \log i \rceil + \sum_{i=1}^{n_2} \lceil \log i \rceil \\
&= \sum_{i=1}^{\lceil \frac{n}{2} \rceil} \lceil \log i \rceil + \sum_{i=\lceil \frac{n}{2} \rceil + 1}^{n_1} \lceil \log i \rceil + \sum_{i=1}^{n_2} \lceil \log i \rceil \\
&\geq B\left(\left\lceil \frac{n}{2} \right\rceil\right) + \sum_{i=n_2+1}^{n_2+n_1-\lceil \frac{n}{2} \rceil} \lceil \log i \rceil + \sum_{i=1}^{n_2} \lceil \log i \rceil \\
&= B\left(\left\lceil \frac{n}{2} \right\rceil\right) + B\left(\left\lfloor \frac{n}{2} \right\rfloor\right)
\end{aligned}$$

which was to show. □

There is also a nice closed formula for $B(n)$. In particular, it implies $B(n) = O(n \log n)$, which is of interest for our analysis of average searchspace size.

Lemma 13: Let $n \in \mathbf{N}$, then

$$B(n) = n \lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$$

Proof
We do induction on n. If $n = 1$, the equality may be easily checked.
If $n > 1$, then

$$\begin{aligned}
B(n) &= B(n-1) + \lceil \log n \rceil \\
&= (n-1) \lceil \log n - 1 \rceil - 2^{\lceil \log n - 1 \rceil} + 1 + \lceil \log n \rceil
\end{aligned}$$

If $\lceil \log n - 1 \rceil = \lceil \log n \rceil$, the claimed equality follows immediately. Now $\lceil \log n - 1 \rceil < \lceil \log n \rceil$ if and only if $n = 2^k + 1$ for some $k \in \mathbf{N}$. In that case it is

$$\begin{aligned}
B(n) &= (n-1) \lceil \log n - 1 \rceil - 2^{\lceil \log n - 1 \rceil} + 1 + \lceil \log n \rceil \\
&= (n-1) \cdot k - \underbrace{2^k}_{=(n-1)} + 1 + k + 1 \\
&= nk - (n-1) + 2 \\
&= nk + n - \underbrace{2(n-1)}_{=2 \cdot 2^k} + 1 \\
&= n(k+1) - 2^{k+1} + 1 \\
&= n \lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1
\end{aligned}$$

That finishes the proof. □

4.2 Searchspace Size of Paths

In this paragraph we account contraction hierarchies with optimal searchspace size for paths. While these results may not seem relevant for practical purposes, they provide insight into some theoretical aspects, though. It will turn out that the average searchspace size in a contraction hierarchy of a path is closely linked to the sequence of sorting numbers. We will show that for an optimal contraction hierarchy H of a path of n vertices, $\mathcal{V}(H) = O(n \log n)$. More specifically, the minimum average searchspace $\mathcal{V}(p_n)$ of a path p_n with n vertices is exactly $B(n+1)$, which is by Lemma 13 $O(n \log n)$.

Lemma 14: Let p_n be the path consisting of n vertices. Then $\mathcal{V}(p_n) \geq B(n+1)$, where $B(n)$ is the n-th sorting number as defined above.

Proof
We proof this lemma by induction on the number n of vertices. If $n = 1$, there is nothing to show, as $B(2) = 1$

Now let $n > 1$. Furthermore let $\prec$ be an order on the vertices $V(p_n)$ of p_n and let $H = \mathcal{H}(p_n, \prec)$. Finally let v be the $\prec$-largest vertex in H. Removal of v splits H into two graphs H_1 and H_2 of n_1 and n_2 vertices. These graphs H_i are by Lemma 10 contraction hierarchies of p_{n_1} and p_{n_2} respectively. Furthermore $\mathcal{V}_{H_i}(v) = \mathcal{V}_H(v) - 1$ for all $v \in V(H_i)$ and thus

$$\mathcal{V}(H) = \mathcal{V}(H_1) + n_1 + \mathcal{V}(H_2) + n_2 + 1$$

By induction hypothesis we have $\mathcal{V}(H_i) \geq B(n_i + 1)$. Hence we may apply Lemma 12 and Lemma 11 to obtain

$$\begin{aligned} \mathcal{V}(H) &\geq B\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) + B\left(\left\lceil \frac{n+1}{2} \right\rceil\right) + n \\ &= B(n+1) \end{aligned}$$

which was to show. □

Furthermore it is clear from the above proof that the lower bound $B(n+1)$ is tight. Hence we can derive the recursive Algorithm 3 to compute a searchspace-optimal contraction hierarchy of p_n.

Algorithm 3: OptimalPathOrder(p)

Input : Path $p = (V, E)$ of n vertices
Output: Order on V, such that the corresponding contraction hierarchy H has average searchspace $\mathcal{V}(H) = B(n+1)$

```
if n = 0 then
    return ⟨ ⟩
else
    Pick vertex v ∈ V separating p into paths p1 and p2 of length ⌊(n−1)/2⌋ and ⌈(n−1)/2⌉
    return OptimalPathOrder(p1) ∘ OptimalPathOrder(p2) ∘ ⟨v⟩
end
```

Note that the algorithm has runtime $O(n)$, if we can pick the vertex v and compute the paths p_1 and p_2 in constant time. If the path p_n is encoded by the number of vertices n and for all $1 \leq i < n$ vertex i is implicitly connected to vertex $i+1$, it is clearly possible to choose the vertex v and compute p_1 and p_2 in constant time. But as n can be encoded in $\lceil \log n \rceil$ bits, the algorithm then has exponential runtime in the input size. However, if p_n is represented by a standard graph data structure like adjacency lists, we may precompute such an assignment of integers to vertices in time $O(n)$ and the algorithm has linear runtime in the input size.

If one considers the algorithm again, one can even guess what searchspace size has in common with binary insertion, where the sequence $B(n)$ has its origin. For each vertex $u \in p_n$ the vertices w such that $u \prec w$ and $\mathrm{dist}_H(u, w) < \infty$ are exactly those vertices that u would be compared with, when being inserted into the sequence by binary insertion. The central vertex v is the $\prec$-largest vertex in every searchspace and binary insertion always starts with this vertex. Furthermore if u occurs in the first half of p_n the next comparison in binary insertion would be with the central vertex of the first half of p_n, which is actually the second $\prec$-largest vertex in the searchspace of u.

Eventually we turn to the minimization of maximal searchspace in contraction hierarchies of paths.

Lemma 15: Let p_n be the path consisting of n vertices. Then

$$\max_{v \in V} \mathcal{V}(v) \geq \lceil \log n + 1 \rceil$$

Furthermore this bound is tight, i.e. there is a contraction hierarchy of p_n, such that $\max_{v \in V} \mathcal{V}(v) = \lceil \log n + 1 \rceil$.

Proof
We do induction on n. If $n = 1$, p_n consists of a single vertex v only and $\mathcal{V}(v) = 1$. Thus $\max \mathcal{V}(v) = \lceil \log n + 1 \rceil$.

If $n > 1$, let $\prec$ be an order on the vertices $V(p_n)$ of p_n and let $H = \mathcal{H}(p_n, \prec)$. Further let v be the $\prec$-maximal vertex in H and let H_1 and H_2 be the two components of $H - \{v\}$. As in the proof of Lemma 14 denote by n_1 and n_2 the number of vertices of H_1 and H_2, respectively. By Lemma 10 H_1 and H_2 are contraction hierarchies of p_{n_1} and p_{n_2} and the induction hypothesis implies

$$\max_{v \in V(H_i)} \mathcal{V}_{H_i}(v) \geq \lceil \log n_i + 1 \rceil$$

As $\mathcal{V}_H(v) = \mathcal{V}_{H_i}(v) + 1$ and since $\max\{n_1, n_2\} \geq \left\lceil \frac{n-1}{2} \right\rceil$ we get

$$\begin{aligned}
\max_{v \in V} \mathcal{V}_H(v) &= \max \left\{ \max_{v \in V(H_1)} \mathcal{V}_H(v), \max_{v \in V(H_2)} \mathcal{V}_H(v), \mathcal{V}_H(v) \right\} \\
&\geq \max \left\{ 1 + \left\lceil \log \left(\left\lceil \frac{n-1}{2} \right\rceil + 1 \right) \right\rceil, 1 \right\} \\
&\geq \left\lceil \log 2 + \log \frac{n+1}{2} \right\rceil \\
&= \lceil \log n + 1 \rceil
\end{aligned}$$

Furthermore, by induction hypothesis, the lower bound is tight for H_1 and H_2 and we may assume that

$$\max_{v \in V(H_i)} \mathcal{V}_{H_i}(v) = \lceil \log n_i + 1 \rceil$$

If we choose the $\prec$-largest vertex v in a such a way that $n_1 = \left\lceil \frac{n-1}{2} \right\rceil$ and $n_2 = \left\lfloor \frac{n-1}{2} \right\rfloor$, we get

$$\max_{v \in V} \mathcal{V}_H(v) = 1 + \left\lceil \log \left\lceil \frac{n-1}{2} \right\rceil + 1 \right\rceil = 1 + \left\lceil \log \left\lceil \frac{n+1}{2} \right\rceil \right\rceil$$

Now

$$\begin{aligned}
& 1 + \left\lceil \log \left\lceil \frac{n+1}{2} \right\rceil \right\rceil > \lceil \log n + 1 \rceil \\
\Leftrightarrow \quad & 2 \left\lceil \frac{n+1}{2} \right\rceil > n + 1 \text{ and } 2 \left\lceil \frac{n+1}{2} \right\rceil > 2^k \text{ and } n + 1 \leq 2^k \text{ for some } k \in \mathbf{N} \\
\Leftrightarrow \quad & n \text{ even and } n + 2 > 2^k \text{ and } n + 1 \leq 2^k \text{ for some } k \in \mathbf{N}
\end{aligned}$$

This can obviously not happen, since $n > 2^k - 2$ and $n \leq 2^k - 1$ which implies $n = 2^k - 1$, but n has to be even. Therefore the bound is tight for n, which finishes the proof. □

The proof already shows that Algorithm 3 returns an order on the vertices of p_n for which the corresponding contraction hierarchy has maximal searchspace size $\max_{v \in V} \mathcal{V}(v) = \lceil \log n + 1 \rceil$. In particular this implies that both the optimum average and the optimum maximal searchspace size are reached in the same contraction hierarchy, which makes the comparison of these two values quite interesting. On the one hand the searchspace size of no vertex exceeds $\lceil \log n + 1 \rceil$ and on the other hand the sum of all searchspace sizes is by Lemma 13 $(n+1)\lceil \log n + 1 \rceil - O(n)$. Thus it appears, that in the optimal contraction hierarchy, the searchspace size of most vertices is approximately $\log n + 1$.

4.3 Searchspace Size of Trees

In order to study the searchspace of contraction hierarchies for cycle-free graphs it is convenient to change the perspective. Instead of counting the vertices $\mathcal{V}(u)$ that get settled during Dijkstra's algorithm with start vertex u, we count for each vertex u the number of vertices v with $u \in \overrightarrow{\mathcal{V}}(v)$. This change of perspective does of course not alter the overall measure of searchspace

$$\mathcal{V}(H) = \sum_{u \in V} \mathcal{V}(u) = \sum_{u \in V} |\overrightarrow{\mathcal{V}}(u)| = \sum_{u \in V} |\{v \in V : v \in \overrightarrow{\mathcal{V}}(u)\}| = \sum_{v \in V} |\{u \in V : v \in \overrightarrow{\mathcal{V}}(u)\}|$$

If one studies this characterization of searchspace on cycle-free graphs, one naturally deals with a family $\mathcal{T}$ of trees, the definition of which is given here.

Definition 2: Let $T = (V, E)$ be an unrooted tree and $\prec$ be an order on V. The set $\mathcal{T}(T, \prec)$ is defined recursively:

(i) If T consists of one single vertex, then $\mathcal{T}(T, \prec) = \{T\}$.

(ii) If T has more than one vertex let v be the $\prec$-largest vertex and let $d = d(v)$ be the degree of v in T. Then v separates T into d subtrees $T_1, \ldots, T_d$ of T. In this case we let

$$\mathcal{T}(T, \prec) = \{T\} \cup \bigcup_{i=1}^{d} \mathcal{T}(T_i, \prec_i)$$

where $\prec_i$ is the order $\prec$ restricted to V.

If it is clear from the context, which order $\prec$ or which tree T is meant, we simply write $\mathcal{T}, \mathcal{T}(T)$ or $\mathcal{T}(\prec)$.

As each vertex v of T is the root of one tree in $\mathcal{T}$ there are exactly n elements in $\mathcal{T}$. The following lemma shows that $\mathcal{T}$ stands in very close relation to our model of searchspace size.

Lemma 16: Let $T = (V, E)$ be an unrooted tree, $\prec$ an order on V and $H = \mathcal{H}(T, \prec)$. Then

$$\mathcal{V}(H) = \sum_{t \in \mathcal{T}(T, \prec)} |V(t)|$$

where $V(t)$ denotes the set of vertices of the tree t.

Proof
The equality can easily be shown by induction on $n = |V|$. If $n = 1$, there is nothing to show. If $n > 1$ let v be the $\prec$-largest vertex and $d = deg(v)$ be the degree of v in T. The vertex v separates T into trees $T_1, \ldots, T_d$, for which we denote the restriction of $\prec$ on $V(T_i)$ by $\prec_i$ and the induced hierarchies by $H_1, \ldots, H_d$.

As v occurs in $\overrightarrow{\mathcal{V}}(u)$ for all $u \in V(T)$, we get $\mathcal{V}(H) = 1 + \sum_{i=1}^{d} \mathcal{V}(H_i) + |V(H_i)|$ and thus

$$\begin{aligned}\mathcal{V}(H) &= 1 + \sum_{i=1}^{d} \mathcal{V}(H_i) + |V(H_i)| \\ &= n + \sum_{t \in \mathcal{T}(T_i, \prec_i)} |V(t)| \\ &= \sum_{t \in \mathcal{T}(T, \prec)} |V(t)|\end{aligned}$$

which finishes the proof. □

The problem of minimizing the sum $\sum |V(t)|$ is an interesting problem on its own, which might for example have applications in tree layout algorithms. Unfortunately neither polynomial-time algorithms nor hardness results are known to the author and attempts to obtain such failed. Thus we do not further pursue this track but prove a lower bound for average searchspace size, which is a straight-forward generalization of the lower bounds for paths.

Intuition suggests, that if a tree T contains a simple path p of length l, then the searchspace size is at least $B(l+1)$. Furthermore, all $n - l$ vertices that are not in p have searchspace at least one which results in a lower bound of $B(l+1) + n - l$.

Lemma 17: Let $T = (V, E)$ be an unrooted tree with diameter Δ_T and let H be a contraction hierarchy of T. Then

$$\mathcal{V}(H) \geq B(\Delta_T + 2) + n - 1 - \Delta_T$$

Proof
The inductive proof works completely analogous to the one of Lemma 14. If $n = 1$, there is nothing to show, as $B(2) + 1 - 1 - 0 = 1$

Now let $n > 1$. If $T = (V, E)$ has diameter Δ_T, there is a simple path p_k consisting of $k = \Delta_T + 1$ vertices in T. Let $\prec$ be an order on V and $v \in V$ be $\prec$-maximal. As T is acyclic, v is separating in G and H. We now distinguish the cases $v \in p_k$ and $v \notin p_k$.

1. If $v \notin p_k$, p_k is contained in a component of $H - \{v\}$ and a component of $G - \{v\}$. Let H_1 and G_1 be these particular components and denote the further components of H by $H_2, \ldots, H_d$. Further let n_i denote the number of vertices in H_i. Then the component G_1 of $G - \{v\}$ has diameter at least Δ_T and by induction hypothesis and Lemma 10 we get

   $$\mathcal{V}(H_1) \geq B(\Delta_T + 2) + n_1 - 1 - \Delta_T$$

   For $i \neq 1$ we have the trivial inequality $\mathcal{V}(H_i) \geq n_i$. Moreover, as T is connected, Lemma 10 implies $\mathcal{V}_H(u) = \mathcal{V}_{H_i}(u) + 1$ and thus

   $$\begin{aligned}\mathcal{V}(H) &= \mathcal{V}_H(v) + \mathcal{V}(H_1) + n_1 + \sum_{i=2}^{d} \mathcal{V}(H_i) + n_i \\ &\geq B(\Delta_T + 2) - 1 - \Delta_T + 1 + \sum_{i=1}^{d} 2n_i \\ &= B(\Delta_T + 2) - 1 - \Delta_T + n + (n-1) \\ &\geq B(\Delta_T + 2) + n - 1 - \Delta_T\end{aligned}$$

2. If on the other hand, $v \in p_k$, removal of v splits p_k into two paths p_1, p_2 of lengths k_1, k_2 such that $k_1 + k_2 = k - 1 = \Delta_T$. For $i = 1, 2$ let p_i be contained in the components H_i of

$H - \{v\}$ and G_i of $G - \{v\}$. As above denote by $H_3, \ldots, H_d$ the further components of $H - \{v\}$ and by n_i the number of vertices in H_i. For $i = 1, 2$, G_i has diameter at least $k_i - 1$ and by Lemma 10 and the induction hypothesis we get

$$\mathcal{V}(H_i) \geq B(k_i + 1) + n_i - k_i$$

For $i > 2$ we use again the inequality $\mathcal{V}(H_i) \geq n_i$. Using Lemma 12 and Lemma 11 we get

$$\begin{aligned}
\mathcal{V}(H) &= \mathcal{V}_H(v) + \mathcal{V}(H_1) + n_1 + \mathcal{V}(H_2) + n_2 + \sum_{i=3}^{d} \mathcal{V}(H_i) + n_i \\
&\geq B(k_1 + 1) + B(k_2 + 1) - k_1 - k_2 + 1 + \sum_{i=1}^{d} 2n_i \\
&\geq B\left(\left\lceil \frac{k_1 + k_2 + 2}{2} \right\rceil\right) + B\left(\left\lfloor \frac{k_1 + k_2 + 2}{2} \right\rfloor\right) - \Delta_T + n + (n - 1) \\
&= \left(B\left(\left\lceil \frac{\Delta_T + 2}{2} \right\rceil\right) + B\left(\left\lfloor \frac{\Delta_T + 2}{2} \right\rfloor\right) + \Delta_T + 1\right) - \Delta_T - 1 - \Delta_T + n + (n - 1) \\
&= B\left(\Delta_T + 2\right) + n - 1 - \Delta_T + \underbrace{(n - \Delta_T - 1)}_{\geq 0}
\end{aligned}$$

This finishes the proof. □

Corollary: Let $T = (V, E)$ be a connected, acyclic graph. There is a contraction hierarchy H of T, such that $\mathcal{V}(H) = B(\Delta_T + 2)$ if and only if T is a path.

Proof
If $\mathcal{V}(H) = B(\Delta_T + 2)$ we get $B(\Delta_T + 2) \geq B(\Delta_T + 2) + n - 1 - \Delta_T$ and thus $\Delta_T + 1 \geq n$, which is only possible if T is a path. □

By Lemma 13 it is $B(\Delta_T + 2) + n - 1 - \Delta_T = O(\Delta_T \log \Delta_T + n)$. This lower bound for searchspace size in trees is not that tight as one might wish. In the proof no assumptions about the components H_i for $i \geq 2$ are made and effectively, all vertices that are not on the path p_k of hoplength Δ_T are completely ignored. Thus there remains room for improvement. One might use stronger assumptions on the structure of T or provide a better analysis of the vertices not on p_k to get better lower bounds.

Furthermore it seems straightforward to generalize these lower bounds to arbitrary graphs. However, the author was not successful with this attempt and it seems that another proof technique or a generalization of Lemma 10 is necessary to obtain such lower bounds. The difficulty with arbitrary graphs, is that edges or vertices cannot be removed as easy as in acyclic graphs, for the removal of an edge or vertex may introduce shortcuts, which let the searchspace grow in an uncontrolled way, even if the vertex or edge in question is at the top or bottom of the hierarchy.

5. Integer Linear Programs for Optimal Contraction Hierarchies

It is possible to compute hierarchies with optimal searchspace by 0/1-integer linear programs. This section provides one such 0/1-integer linear program.

The first attempt to obtain a linear program for optimal search space in contraction hierarchies was to abstract from the arcs in the hierarchy and somehow express the overall searchspace independent of the presence of single edges. However this approach was not very fruitful, as the searchspace seems to be intimately connected to the presence of single arcs. Instead of further pursuing that strategy we explicitly model the process of vertex contraction and shortcut insertion. Given an order $\prec$ on the vertices V we decide for each possible arc (u, v) whether (u, v) is present in the hierarchy corresponding to $\prec$ or not. From this information the searchspace $\overrightarrow{\mathcal{V}}(u)$ is easily reconstructible.

The variables of the linear program can be subdivided into four different kinds. For each two vertices $u, v \in V$ our 0/1-integer linear program has the following variables. There are two *order-variables* x_{uv} and x_{vu} modeling an order $\prec$ on the vertices, where $x_{uv} = 1$, if and only if $u \prec v$. Furthermore there are two *edge-variables* e_{uv} and e_{vu}, where $e_{uv} = 1$, if and only if there is an arc (u, v) in the hierarchy H corresponding to the order $\prec$. Finally there are *searchspace-variables* s_{uv} and s_{vu}, where $s_{uv} = 1$, if and only if v is in the searchspace of u in the hierarchy H.

Formally the set of variables of our linear program is given by

$$\{x_{uv} : u, v \in V\} \cup \{e_{uv} : u, v \in V\} \cup \{s_{uv} : u, v \in V\}$$

The constraints can again be divided into three different types. To enforce that each order $\prec$ on V induced by the order-variables is well-defined, there is a set of *order-constraints*. Additionally there are *edge-constraints* that relate the edge-variables with the order $\prec$ induced by the order-variables. Finally there are *searchspace-constraints* that guarantee that s_{uv} is 1 if and only if $v \in \overrightarrow{\mathcal{V}}(u)$.

The order $\prec$ on the vertices V expressed by the order-variables x_{uv} has to be reflexive, antisymmetric, transitive and total. These requirements can be expressed by the following constraints:

$x_{uu} = 1$	for all $u \in V$	Reflexivity
$x_{uv} + x_{vu} \leq 1$	for all distinct $u, v \in V$	Antisymmetry
$x_{uv} + x_{vw} - x_{uw} \leq 1$	for all $u, v, w \in V$	Transitivity
$x_{uv} + x_{vu} \geq 1$	for all $u, v \in V$	Totality

Assume for now, that for any solution of the linear program $e_{uv} = 1$ if and only if (u, v) is an arc in the hierarchy corresponding to $\prec$. Then we can formulate the search-constraints as follows:

$s_{uu} = 1$	for all $u \in V$	Each vertex u is contained in $\overrightarrow{\mathcal{V}}(u)$
$s_{uw} + e_{wv} - s_{uv} \leq 1$	for all $u, v, w \in V$	$v \in \overrightarrow{\mathcal{V}}(u)$ if there is some $w \in \overrightarrow{\mathcal{V}}(u)$ and an arc (w, v)

In order to express, whether the hierarchy corresponding to $\prec$ contains a specific arc (u, v) we define for each three vertices $u, v, w \in V$ an auxiliary *shortcut-constant* $c_{uv}(w)$, which indicates whether w lies on a shortest path between u and v.

$$c_{uv}(w) = \begin{cases} 1 & \text{if } w \notin \{u, v\} \text{ and } w \text{ lies on a shortest path between } u \text{ and } v \\ 0 & \text{otherwise} \end{cases}$$

All values $c_{uv}(w)$ can obviously be precomputed in polynomial time. Finally we turn to the edge-constraints, which are the most complicated part of the linear program. Recall that e_{uv} should be 1, if (u, v) is an arc in the hierarchy corresponding to $\prec$. This requirement can be expressed by the following constraints:

$e_{uu} = 0$	for all $u \in V$	No self-loops
$e_{uv} + e_{vu} \geq 1$	for all edges $\{u, v\} \in E$	Insert all edges of G
$e_{uv} - x_{uv} \leq 0$	for all distinct $u, v \in V$	Do not insert arcs (u, v), where $u \succ v$
$e_{uv} + x_{vu} + \sum_{w \in V} c_{uv}(w) \cdot x_{uw} \geq 1$	for all distinct $u, v \in V$	Insert arc (u, v) if $u \prec v$ and there is no shortest path (u, w, v), such that $u \prec w$
$e_{uv} + c_{uv}(w) \cdot x_{uw} \leq 1$	for all distinct $u, v \in V$ and all $w \in V$	Do not insert arc (u, v) if there is some shortest path (u, w, v), such that $u \prec w$

Now we only have to specify the target function of this linear program. There are two possibilities. If we optimize for minimal searchspace, the target function can be chosen as $\sum s_{uv}$, while minimization of $\sum e_{uv}$ solves the problem **CHPS**, as considered in Chapter 3. In the latter case one could drop the searchspace-variables and searchspace-constraints, as they stand in no relation to the target function. By construction it is clear that any optimal solution to the linear program induces an optimal contraction hierarchy.

Lemma 18: Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices.

(i) There is a 0/1-integer linear program $A_V \cdot x \leq b$ with target function $c^T x = \min$, such that the matrix A_V has $O(n^2)$ columns and $O(n^3)$ rows and such that each minimal solution x corresponds to a contraction hierarchy H of G with minimal average searchspace size $\mathcal{V}(H)$.

(ii) There is a 0/1-integer linear program $A_E \cdot x \leq b$ with target function $c^T x = \min$, such that the matrix A_E has $O(n^2)$ columns and $O(n^3)$ rows and such that each minimal solution x corresponds to a contraction hierarchy H of G with a minimal number of shortcuts.

6. Experiments

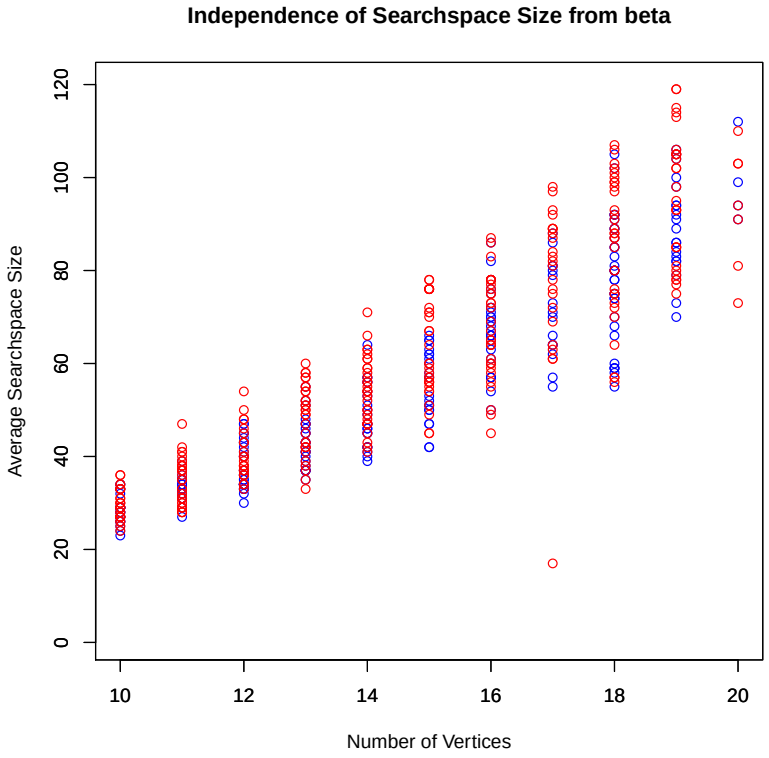
The above integer linear programs were implemented in Java using CPLEX [cpl] as black-box solver for linear programs. Despite the power of today's integer linear program solvers, there were instances with only 50 vertices that took hours to solve. Especially highly symmetric graphs, i.e. rectangular grid graphs or regular trees turned out to be hard. Thus, systematic experiments were only carried out for very small instances, which hinders the interpretation of results, as it is hardly possible to reason about asymptotic behaviour or impact of parameters. In particular, experiments were conducted with two different types of random graphs.

1. *Waxman graphs W* were generated by choosing uniformly at random $n \in \{10, \dots, 20\}$ points in the unit square and inserting an edge $\{u, v\}$ between the points u and v with probability $\alpha \cdot e^{-d(u,v)/\beta}$, where $d(u, v)$ denotes the euclidean distance of the points u and v. Finally, to make the graph connected, the largest component of the resulting graph was chosen. The parameters α and β were chosen as $\alpha \in \{0.1, 0.2, 0.3, 0.4\}$ and $\beta \in \{1.0, 1.1, 1.2, \dots, 3.0\}$.
2. *Trees T* with $n \in \{10, \dots, 20\}$ vertices were generated by the following procedure. Initially T consists of n vertices and no edges. In each step two components of T and a vertex in each component are chosen uniformly at random and an edge is inserted between those vertices. This procedure is carried out, until the graph is connected.

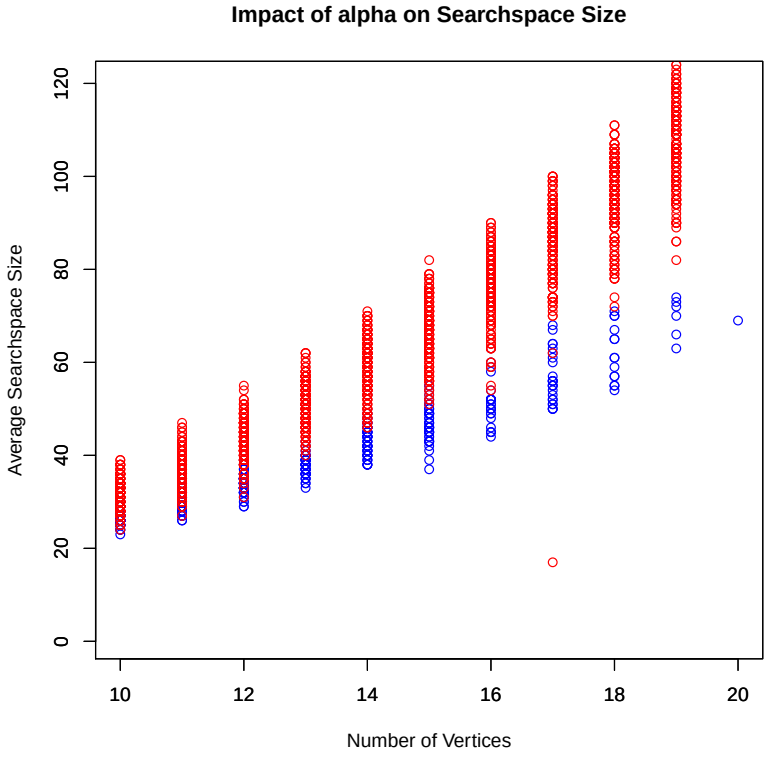
In the following paragraphs we denote by $\mathcal{V}(G)$ always the minimum average searchspace size of contraction hierarchies of the graph G.

Waxman Graphs

Waxman graphs are of geometric nature, in the sense that the parameter β governs the probability of long edges. The behaviour of minimum average searchspace size on geometric graphs is of special interest for most real-world applications, as many graphs are of geometric origin and furthermore searchspace size is the main performance factor. Unfortunately our experiments indicate little impact of the geometric structure of Waxman graphs on $\mathcal{V}$. In particular, $\mathcal{V}$ seems to be independent of the parameter β, as may be seen in Figure 6.1a. The author thinks, that the impact of β should grow for large n, as the underlying geometry of the graph gains importance when the graphs get larger. The parameter α however, has impact on the average searchspace size as can be seen in Figure 6.1b. This seems quite natural, as higher values of α increase the overall probability of edges in Waxman graphs and thus searchspace size grows.



(a) Scatter plot of $\mathcal{V}(W)$ for Waxman graphs W with arbitrary $\alpha, \beta = 1$ (blue dataset) and $\beta = 3$ (red dataset)



(b) Scatter plot of $\mathcal{V}(W)$ for Waxman graphs W with arbitrary $\beta, \alpha = 0.1$ (blue dataset) and $\alpha = 0.4$ (red dataset)

Figure 6.1: Impact of parameters α and β on $\mathcal{V}(W)$ for Waxman graphs W.

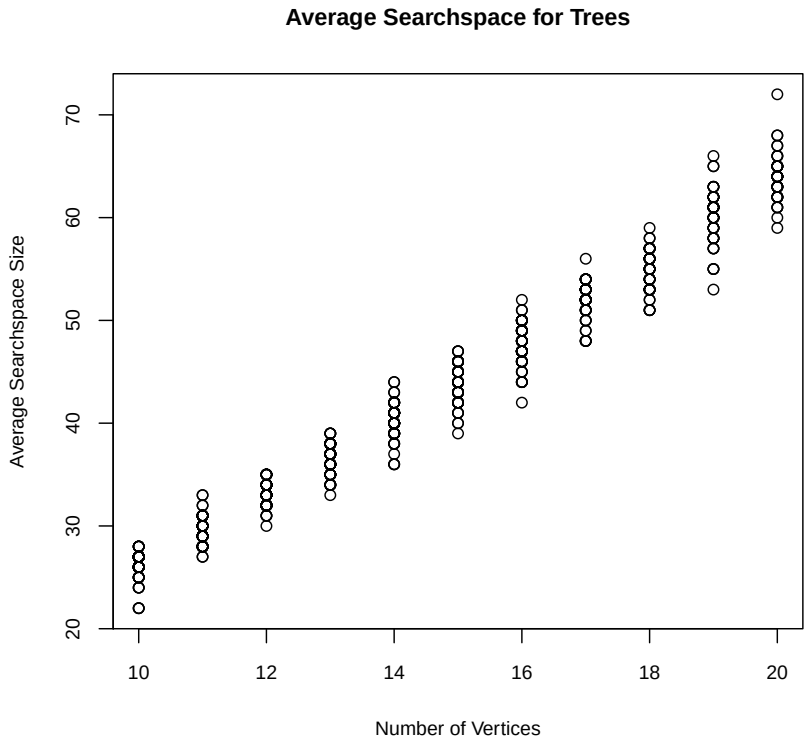


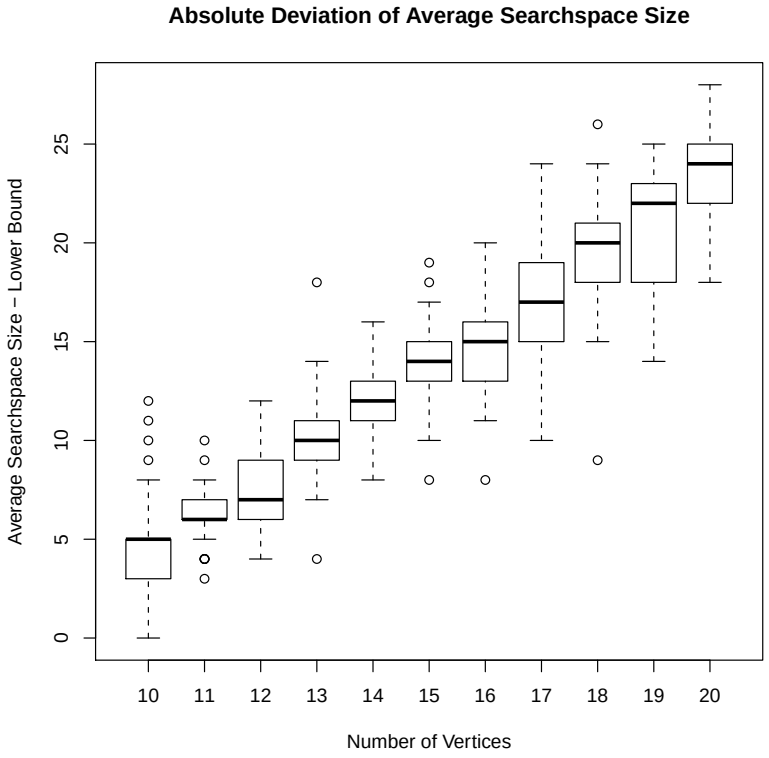
Figure 6.2: $\mathcal{V}(T)$ for random trees T with increasing number of vertices.

Trees

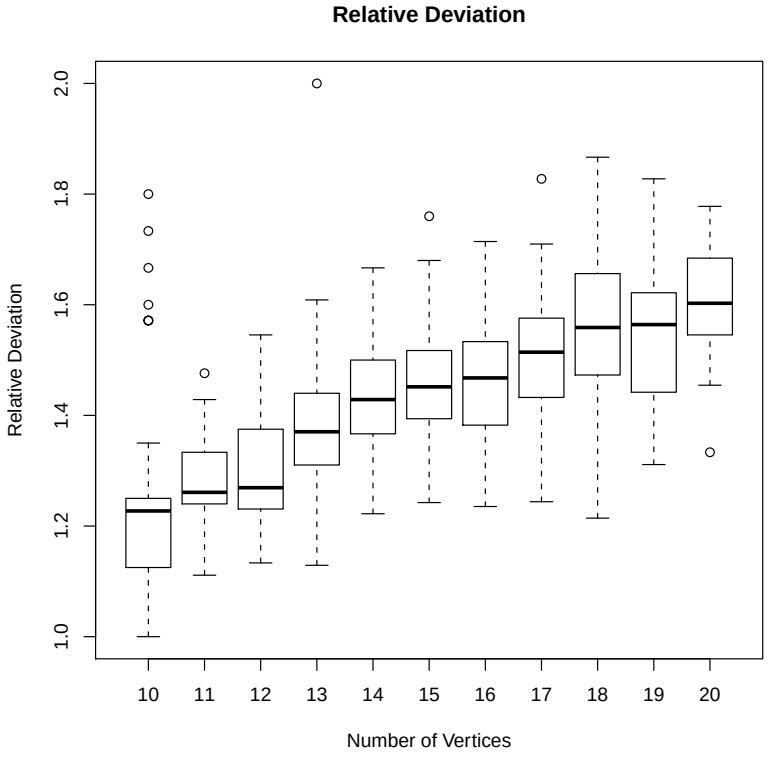
Of special interest, with respect to this thesis, are experiments evaluating the lower bounds of minimum searchspace size. In this paragraph we denote by $B(T)$ the lower bound for $\mathcal{V}(T)$ of Lemma 17. Recall that $B(T) = O(\Delta_T \log \Delta_T + n)$, where Δ_T is the diameter of T. In order to understand the quality of the lower bound $B(T)$ we consider two measures of deviation of $\mathcal{V}(T)$ from $B(T)$. The *absolute deviation* $\mathcal{V}(T) - B(T)$ of $V(T)$ from $B(T)$, the *relative deviation* $(\mathcal{V}(T) - B(T))/B(T) = \mathcal{V}(T)/B(T) - 1$ of $\mathcal{V}(T)$ from $B(T)$. For simplicity we choose $\mathcal{V}(T)/B(T)$ as a measure of relative deviation.

Generally it appears that $\mathcal{V}(T)$ is growing linearly in n, as can be seen in Figure 6.2, which agrees with the lower bound of $O(\Delta_T \log \Delta_T + n)$. Furthermore the absolute deviation $\mathcal{V}(T) - B(T)$ seems to grow linearly in the number n of vertices in T. As $B(T)$ is also linear in n, this suggests that $\mathcal{V}(T) = O(B(T))$. By contrast, the relative deviation $\mathcal{V}(T)/B(T)$ depicted in Figure 6.3b indicates that $\mathcal{V}(T)/B(T)$ grows at least logarithmically with the number of vertices, which would imply $\mathcal{V}(T) = O(B(T) \cdot \log n)$. The latter bound of $\mathcal{V}(T) = O(B(T) \cdot \log n)$ seems more plausible, as we already observed in the proof of Lemma 17 that there are $n - \Delta_T$ vertices of T that do not contribute at all to $B(T)$.

Altogether these experiments suggest that there is some function $f(n) < cn$ for some constant c, such that $\mathcal{V}(T) = O(f(n) \cdot B(T))$, which is a quite strong argument that the lower bound $B(T)$ is nearly tight – even on arbitrary trees.



(a) Boxplot of the absolute deviation $\mathcal{V}(T) - B(T)$



(b) Boxplot of the relative deviation $\mathcal{V}(T)/B(T)$

Figure 6.3: Deviation of $\mathcal{V}(T)$ for random trees T from the lower bound $B(T)$ from Lemma 17.

7. Conclusion

In this thesis contraction hierarchies are studied from a theoretical point of view. The author is able to prove that it is **NP**-complete to determine, if a graph admits a contraction hierarchy with only a given number of additional shortcuts. Additionally the corresponding optimization problem of finding the contraction hierarchy with a minimal number of shortcuts is shown to be **APX**-hard. Provided $\mathbf{P} \neq \mathbf{NP}$, the P-reduction used in that proof results in a lower bound of $\frac{7}{6}$ for possible approximation ratios, hence polynomial-time approximation schemes for this problem are very unlikely to exist. However it remains open, whether approximation algorithms exist and even polynomial-time approximation schemes might still exist for restricted graph classes.

Furthermore a simple model of search space in contraction hierarchies suited for theoretical work is developed and two special cases of contraction hierarchies are examined with respect to this model. Starting with these examinations it is then possible to derive lower bounds for the performance of contraction hierarchies on paths and trees. It is shown that average searchspace size of $O(n \log n)$ is optimal for paths and algorithms to compute contraction hierarchies with this searchspace size are provided. Additionally, a lower bound of $\log n + 1$ for maximal searchspace size in contraction hierarchies of paths is given. The author further shows that $O(\Delta \log \Delta + n)$, where Δ is the diameter, is a lower bound for the average searchspace size in contraction hierarchies of trees. However, it is not clear if and how these results generalize to arbitrary graphs, which seems to be an interesting question for future work.

Theoretical investigation of contraction hierarchies is a rather young topic and thus there remain many unanswered questions. It is still unproven – yet very likely – that computing search-space minimal contraction hierarchies is **NP**-hard. The complexity of this very question for restricted graph classes – like trees or planar graphs – remains open, too. Furthermore it is unknown whether there are approximation or fixed parameter tractable algorithms for searchspace minimal contraction hierarchies. The challenging task of finding such algorithms has not been addressed at all in this thesis.

Finally this work embeds in the broader framework of theoretical investigation of speedup techniques for Dijkstra's algorithm in general. There one is confronted with similar questions concerning other techniques like Arc-Flags, ALT or Highway Node Routing. In [BCK+10] it is shown that optimal preprocessing is **NP**-hard for these techniques, but there are neither algorithms with performance guarantees nor impossibility results known to the author.

Acknowledgement

First and foremost I want to thank my advisor Reinhard Bauer for supporting me in a multitude of ways and for having always time and patience for questions and interesting discussions. My parents exercise even more patience, for which I owe them a debt of gratitude. Finally, I want to thank Anna, who bears an everyday life with me, which I think, ranks first on the patience scale.

Bibliography

[AMOT90] R. K. Ahuja, K. Mehlhorn, J. Orlin, and R. E. Tarjan, "Faster algorithms for the shortest path problem," *J. ACM*, vol. 37, no. 2, pp. 213–223, 1990.

[BCK+10] R. Bauer, T. Columbus, B. Katz, M. Krug, and D. Wagner, "Preprocessing speed-up techniques is hard," 2010, submitted to *International Conference on Complexity and Algorithms*.

[BD09] R. Bauer and D. Delling, "SHARC: Fast and Robust Unidirectional Routing," *ACM Journal of Experimental Algorithmics*, vol. 14, pp. 2.4–2.29, May 2009, special Section on Selected Papers from ALENEX 2008. [Online]. Available: http://doi.acm.org/10.1145/1498698.1537599

[cpl] "CPLEX," mathematical programming optimizer, ILOG, http://www.ilog.com/products/cplex. [Online]. Available: http://www.ilog.com/products/cplex

[Dij59] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, no. 1, pp. 269–271, 1959.

[FT87] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[GH05] A. V. Goldberg and C. Harrelson, "Computing the Shortest Path: A* Search Meets Graph Theory," in *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*, 2005, pp. 156–165.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GKW06] A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for A*: Efficient Point-to-Point Shortest Path Algorithms," in *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*. SIAM, 2006, pp. 129–143.

[GSSD08] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," in *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, ser. Lecture Notes in Computer Science, C. C. McGeoch, Ed., vol. 5038. Springer, June 2008, pp. 319–333.

[Hås01] J. Håstad, "Some optimal inapproximability results," *J. ACM*, vol. 48, no. 4, pp. 798–859, 2001.

[HKMS09] M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling, "Fast Point-to-Point Shortest Path Computations with Arc-Flags," in *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, ser. DIMACS Book, C. Demetrescu, A. V. Goldberg, and D. S. Johnson, Eds. American Mathematical Society, 2009, vol. 74, pp. 41–72.

[HSW04] M. Holzer, F. Schulz, and T. Willhalm, "Combining speed-up techniques for shortest-path computations," in *Proceedings Third International Workshop on Experimental*

and Efficient Algorithms (WEA), ser. LNCS, vol. 3059. Springer, 2004, pp. 269–284.

[Kan92] V. Kann, "On the approximability of np-complete optimization problems," Ph.D. dissertation, Royal Institute of Technology Stockholm, 1992.

[PY88] C. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," in *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1988, pp. 229–234.

[Slo09] N. J. A. Sloane, "The On-Line Encyclopedia of Integer Sequences," 2009. [Online]. Available: http://www.research.att.com/~njas/sequences/

[SS05] P. Sanders and D. Schultes, "Highway Hierarchies Hasten Exact Shortest Path Queries," in *Proceedings of the 13th Annual European Symposium on Algorithms (ESA'05)*, ser. Lecture Notes in Computer Science, vol. 3669. Springer, 2005, pp. 568–579.

[SS07] D. Schultes and P. Sanders, "Dynamic Highway-Node Routing," in *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, ser. Lecture Notes in Computer Science, C. Demetrescu, Ed., vol. 4525. Springer, June 2007, pp. 66–79.

[Tar76] R. E. Tarjan, "Edge-disjoint spanning trees and depth-first search," *Acta Inf.*, vol. 6, pp. 171–185, 1976.

[WW07] D. Wagner and T. Willhalm, "Speed-up techniques for shortest-path computations," in *STACS*, 2007, pp. 23–36.