Karlsruhe Institute of Technology

Search Space Size in Contraction Hierarchies

Diplomarbeit
von

Tobias Columbus

At the Departement of Informatics
Institute of Theoretical Computer Science

Erstgutachter:	Prof. Dr. Dorothea Wagner
Zweitgutachter:	Prof. Dr. Peter Sanders
Betreuende Mitarbeiter:	Dr. Ignaz Rutter
	Dr. Reinhard Bauer

Bearbeitungszeit: 15. Mai 2012 – 15. Oktober 2012

Abstract

This thesis is concerned with a so-called speedup technique for Dijkstra's algorithm: *Contraction hierarchies*. We present a model of contraction hierachies sufficiently sophisticated to capture real-world applications but yet concise enough to lend itself to theoretical investigations. This model reveals an interrelation between contraction hierarchies and filled graphs that directly relates the search space size and space consumption of contraction hierarchies to the height of elimination trees and the number of fill-edges in filled graphs, respectively. These observations facilitate the construction of contraction hierarchies with an upper bound of $\mathcal{O}(\mathrm{tw}(G) \cdot \log(n))$ and $\mathcal{O}(\sqrt{n})$ on the maximum search space size and an upper bound of $\mathcal{O}(\mathrm{tw}(G) \cdot n \log(n))$ and $\mathcal{O}(n \log(n))$ on their space consumption for graphs of treewidth $\mathrm{tw}(G)$ and graphs admitting recursive $\mathcal{O}(\sqrt{n})$-separator decompositions. We further consider the problem of locally modifying contraction hierarchies to increase the performance of subsequent distance queries.

Deutsche Zusammenfassung

Diese Arbeit beschäftigt sich mit einer sogenannten Beschleunigungstechnik für Dijkstra's Algorithmus: *Contraction Hierarchies*. Wir erarbeiten ein Modell, welches die, in der Praxis gebräuchlichen, heuristischen Contraction Hierarchies erfasst und sich trotzdem für theoretische Betrachtungen eignet. Insbesondere deckt dieses Modell einen unerwarteten Zusammenhang zwischen Contraction Hierarchies und den wesentlich älteren und besser untersuchten Problemen einen gefüllten Graphen mit wenigen Kanten und einen "elimination tree" geringer Höhe zu berechnen auf. Diese Beobachtungen ermöglichen die Konstruktion von Contraction Hierarchies mit garantierter maximaler Suchraumgröße $\mathcal{O}(\mathrm{tw}(G) \cdot \log(n))$ bzw. $\mathcal{O}(\sqrt{n})$ und garantiertem Platzverbrauch $\mathcal{O}(\mathrm{tw}(G) \cdot n \log(n))$ bzw. $\mathcal{O}(n \log(n))$ für Graphen von Baumbreite $\mathrm{tw}(G)$ bzw. Graphen mit Zerlegungen in Separatoren der Größe $\mathcal{O}(\sqrt{n})$. Schlussendlich untersuchen wir noch inwieweit lokale Änderungen die Performanz von Distanz-Abfragen in Contraction Hierarchies beeinflussen.

Declaration
I hereby confirm that this document has been composed by myself, and describes my own work, unless otherwise acknowledged in the text.

Erklärung
Ich erkläre hiermit die vorliegende Arbeit selbständig verfasst zu haben und keine außer den angegebenen Quellen verwendet zu haben.

Karlsruhe, den 15. Oktober 2012 _____

Contents

List of Figures

List of Algorithms

List of Theorems

Introduction

The problem of determining the distance between two given points of a network is one of the fundamental problems in computer science whose domain of application spans the seemingly unrelated areas of route planning, computational biology and social sciences – among others. The classic solution dating back to 1959 is the algorithm of Dijkstra [Dij59]. To be precise, Dijkstra's algorithm does not solve the problem as stated above but rather computes the distance between one prescribed source and all other points – an overhead that is negligible if the network in question is sufficiently small but becomes noticeable in networks with literally millions of points, such as e.g. the road networks of whole continents or social networks with millions of users. Stimulated by the 9th DIMACS implementation challenge [DGJ09], so called *speedup techniques for Dijkstra's algorithm* emerged. Targeted at road networks and exploiting their specific structure, some of these techniques outperform Dijkstra's algorithm by factors of 10^6. To achieve this impressive speedup, many of these algorithms employ some kind of preprocessing to augment the graph with additional information that facilitates fast subsequent distance queries. Although there is overwhelming experimental evidence of the effectiveness of these techniques, near to nothing is known about why exactly they are so successful. This thesis provides at least a partial answer to this latter question for one of these speedup techniques called *contraction hierarchies*. More precisely, we develop a formal model of contraction hierarchies and investigate the performance of distance queries with respect to this model. We are able to show that distance queries in contraction hierarchies outperform Dijkstra's algorithm for any class of graphs that admits small separators such as e.g. planar graphs. More precisely, we prove the existence of contraction hierarchies of graphs of treewidth $\mathrm{tw}(G)$ and of graphs having separators of size $\sqrt{n}$ that facilitate distance queries settling at most $\mathcal{O}(\mathrm{tw}(G) \cdot \log(n))$ and $\mathcal{O}(\sqrt{n})$ vertices, respectively. Furthermore, we investigate how our approach relates to previous work of Abraham et al. [AFGW10; ADFGW11] on the performance of contraction hierarchies. To this end, we conclude that our approach does at least not perform worse than the approach of Abraham et al. in the worst case. Finally, we are concerned with local modifications to the preprocessed network while retaining its most important properties.

Related Work

The reader may consult the recent survey of Sommer [Som12] for a comprehensive overview of many preprocessing techniques for shortest path and distance queries not only in road networks but for arbitrary static graphs. In what follows below, we give some more specific references related to contraction hierarchies and our findings.

Contraction hierarchies were first introduced by Geisberger et al. and the original article [GSSD08] is still the definitive treatment. A recent account of contraction hierarchies that focuses on their deployment in mobile route planning may also be found in [GSSV12]. Broadening the perspective, the survey of Delling et al. [DSSW09] contains an overview and experimental evaluation of the different competing speedup techniques for Dijkstra's algorithm on road networks – of which contraction hierarchies are only a single instance – that emerged during the 9th DIMACS Implementation Challenge [DGJ09]. The article [BDSSSW10] of Bauer et al. further explores the possibility to combine several of these techniques and includes a detailed experimental study. However, the currently fastest practical technique known as hub labeling due to Abraham et al. [ADGW11; ADGW12] misses in both these articles as hub labels were only discovered after the publication of [DSSW09] and [BDSSSW10]. Interestingly, it is this currently fastest algorithm that is based on a profound theoretical model of road networks that was developed by Abraham et al. in an attempt to explain the general success of speedup techniques for Dijkstra's algorithm on road networks. Not only do Abraham et al. utilise their model of road networks to devise another speedup technique in [ADGW11; ADGW12], but they also obtain guarantees for the performance of several other speedup techniques in [AFGW10; ADFGW11]. Apart from these recent results, the author is not aware of any other performance guarantees for any of the various known speedup techniques for Dijkstra's algorithm on road networks.

Another strand of theoretical research concerned explicitly with speedup techniques is the work of Bauer et al. [BDDW09; BCKKW10; BBRW12]. Beginning with the study of the problem to optimally augment a graph with a prescribed number of shortcuts in the article [BDDW09], Bauer et al. consider the computational complexity of optimally filling the degree of freedom present in many preprocessing phases of speedup techniques in [BCKKW10] and further strengthen their results for a particular technique in [BBRW12]. It is also this context in which the author's student thesis [Col09] was already concerned with search spaces in contraction hierarchies. However, in [Col09] only contraction hierarchies of paths could be dealt with satisfactorily.

Although we start off with a practical algorithm having its roots in the route planning community, our findings also relate to more theoretical developments. Several researchers have been concerned with the problem of constructing a data structure of prescribed size S that allows subsequent distance queries to be answered in time $\mathcal{O}(f(n,S))$, where f is increasing in n and decreasing in S. Some of these techniques utilise separator decompositions in their preprocessing phase and are therefore comparable to our approach to preprocessing contraction hierarchies. The first such data structure employing separator decompositions seems to have been constructed

in 1996 by Djidjev [Dji96]. Recently, Mozes and Sommer [MS12] devised a preprocessing algorithm that facilitates distance queries on planar graphs in time $\tilde{\mathcal{O}}(n/\sqrt{S})$ using only $\mathcal{O}(S)$ space.

Organisation of this Thesis

We introduce some basic notation in Chapter 1. Chapter 2 is then devoted to the development and justification of our particular model of contraction hierarchies. We derive upper bounds on both the space consumption and the running time of distance queries for this very model of contraction hierarchies in Chapter 3. Finally, we investigate the effect of local modifications to contraction hierarchies in Chapter 4. All in all, we tried hard to always use the same notation to refer to the same concepts. For the convenience of the reader, we included an index containing the most frequently used symbols and their point of first use or definition.

Acknowledgement

I am indebted to *Reinhard Bauer* and *Ignaz Rutter* for many helpful suggestions concerning both the subject matter and the style of presentation. It is their support and constant encouragement that made the endeavour of this thesis possible. Furthermore, I thankfully acknowledge the generous grant of the *Begabtenstiftung Informatik Karlsruhe*. Many of the findings presented in this thesis are generalisations of research carried out during my scholarship and these results would never have emerged if I had not studied the easier cases back then. Finally, *Felix Wellen* and *Jonathan Zachhuber* volunteered to proofread large parts of this manuscript for which I thank them a lot. All remaining quirks and errors are, of course, mine.

1 Preliminaries and Notation

In this chapter we fix notation and terminology used throughout this work. We assume the reader to be familiar with the very basics of graph theory, treated e.g. in the first chapter of [Die06].

We distinguish between directed and undirected graphs. We call the latter simply *graphs* and refer to the former as *directed graphs* or *digraphs*. Edges of digraphs are also called *arcs* and will be written uv, where u is the source and v is the target vertex. Their undirected counterparts are denoted by $\{u, v\}$. Paths in both graphs and digraphs are denoted by $p = (x_1, x_2, \ldots, x_\ell)$. The number of edges or arcs of a path p is denoted by $|p|$. A *weighted graph* or *weighted digraph* is a graph or digraph equipped with positive, real *edge lengths* $\mathrm{len}_G(u, v)$ or *arc lengths* $\mathrm{len}_G(uv)$. As usual, the length $\mathrm{len}(p)$ of a path p is the sum over all the lengths of its edges and a shortest path from s to t – or simply a shortest s-t-path – is a path of minimal length with source s and target t. We denote by $\mathrm{dist}_G(s, t)$ the length of a shortest s-t-path from some vertex s to some other vertex t.

Computing the distance $\mathrm{dist}_G(s, t)$ or a shortest path between a source vertex s and a target vertex t in a weighted graph or digraph is a classic problem of computer science. If one restricts this problem to non-negative or – like we do – even strictly positive edge weights, there is a classic solution due to Dijkstra [Dij59] known as Dijkstra's algorithm, which computes for a fixed source s the distances $\mathrm{dist}_G(s, u)$ for all $u \in V$. Essentially, Dijkstra's algorithm starts with the given source vertex $s \in V$ and visits all the vertices $u \in V$ sorted by ascending distances $\mathrm{dist}_G(s, u)$. For this purpose, it keeps for each $u \in V$ a *tentative distance* $d(u)$ that is initially set to ∞ if $u \neq s$. Starting with s of tentative distance $d(s) = 0$, it chooses in each iteration the vertex u of minimal tentative distance for which $\mathrm{dist}_G(s, u)$ is still unknown and *settles* it. Here, settling of a vertex $u \in V$ means setting $\mathrm{dist}_G(s, u)$ to $d(u)$ and *relaxing* all the outgoing arcs $uv \in A$, that is, comparing and possibly updating all the tentative distances $d(v)$ with $d(u) + \mathrm{len}_G(uv)$. We call a vertex *discovered* if its tentative distance has been updated at least once. The aforementioned choice of the vertex u of minimal tentative distance $d(u)$ is commonly implemented by some kind of priority queue that keeps track of all the vertices that are discovered but not settled yet. For this reason, we call the set of vertices that are discovered but not settled the *queue* and denote it by Q. An implementation of Dijkstra's algorithm in pseudo code may be seen in Algorithm 1.1.

Broadly speaking, Dijkstra's algorithm is correct since each vertex $u \in V$ gets settled only after all vertices with smaller tentative distances are already settled, i.e. after all

Algorithm 1.1: Dijkstra's algorithm

Data: Weighted digraph $G = (V, A)$
Input: Source vertex $s \in V$
Output: Distances $\operatorname{dist}_G(s, u)$

// Initial tentative distances
1 **foreach** $u \in V \setminus \{s\}$ **do** $d(u) \leftarrow \infty$
2 $d(s) \leftarrow 0$

// Vertices discovered but not settled.
3 $Q \leftarrow \{s\}$
4 **while** $Q \neq \emptyset$ **do**
5 Choose $u \in Q$ such that $d(u)$ is minimal
6 $Q \leftarrow Q \setminus \{u\}$
7 **foreach** $uv \in A$ **do**
8 **if** $d(u) + \operatorname{len}_G(uv) < d(v)$ **then**
9 $d(v) \leftarrow d(u) + \operatorname{len}_G(uv)$
10 **if** $v \notin Q$ **then** $Q \leftarrow Q \cup \{v\}$
11 **return** $\operatorname{dist}_G(s, u) = d(u)$

vertices that can possibly lie on a path shorter than the current tentative distance $d(u)$ are taken into consideration. Furthermore, each possible path is taken account of. All the outgoing arcs uv of any settled vertex u are relaxed, so that any vertex reachable from s is eventually inserted into the queue and thus eventually settled. Making this argument precise, requires a bit of work, which we want to invest nevertheless, for we can make good use of it in what follows below. We begin with some rather obvious remarks about the general workings of Algorithm 1.1, but state them explicitly, for we will make frequent use of them in this and the next chapter.

1.1 Remark: Consider Algorithm 1.1.

(a) The tentative distance $d(u)$ of a single vertex u decreases monotonically with each iteration of Algorithm 1.1.

(b) The minimum tentative distance among all vertices $u \in Q$ increases monotonically with each iteration of Algorithm 1.1.

(c) $d(u) \geq \operatorname{dist}_G(s, u)$ for all $u \in V$ and during the whole course of Algorithm 1.1.

(d) Algorithm 1.1 terminates. †

Proof:

(a) Observe that the tentative distance $d(v)$ of a vertex v can only change in line 9, where it is overwritten with $d(u) + \text{len}_G(uv)$. However, $d(v)$ is overwritten if and only if the new value value is less than $d(v)$. This already shows part (a).

(b) When u is removed from Q in line 6, then $d(u) \leq d(v)$ for all $v \in Q$ by the choice of u. Furthermore, any tentative distance $d(v)$, that is modified during this iteration, satisfies $d(v) = d(u) + \text{len}_G(uv) \geq d(u)$ afterwards. Hence, the minimum $\min_{u \in Q} d(u)$ is monotonically increasing with each iteration of Algorithm 1.1.

(c) We do induction on the number of iterations. After initialisation of $d(-)$ in line 1 and line 2, we have $d(s) = 0$ and $d(u) = \infty$ if $u \neq s$. Our claim is thus satisfied just before the first iteration. Furthermore, $d(-)$ gets modified only in line 9, where $d(v)$ is overwritten with $d(u) + \text{len}_G(uv)$. However, the induction hypothesis implies that at this point $d(u) \geq \text{dist}_G(s, u)$, which gives

$$d(u) + \text{len}_G(uv) \geq \text{dist}_G(s, u) + \text{len}_G(uv) \geq \text{dist}_G(s, v)\,.$$

This finishes the proof.

(d) We have shown in (a) and (b) that $d(u)$ is monotonically decreasing for each single vertex u but $\min_{u \in Q} d(u)$ is monotonically increasing. As v is inserted into Q only after $d(v)$ got strictly smaller, it follows that no vertex is inserted into Q twice. Moreover, each vertex contained in Q is eventually settled and Algorithm 1.1 therefore terminates after at most $|V|$ iterations. □

According to № 1.1, $d(u)$ is decreasing and satisfies $d(u) \geq \text{dist}_G(s, u)$ during the whole course of Dijkstra's algorithm. Consequently, if $d(u) = \text{dist}_G(s, u)$ in one iteration, then $d(u) = \text{dist}_G(s, u)$ for the whole remainder of the algorithm. In order to show that Algorithm 1.1 actually computes the correct distances $\text{dist}_G(s, u)$, it therefore suffices to show that, for any vertex u, there is some point in time at which $d(u)$ and $\text{dist}_G(s, u)$ are equal. We are even able to specify this point in time a bit more precisely in terms of the following lemma.

1.2 Lemma: Each vertex $u \in V$ with $\text{dist}_G(s, u) < \infty$ is eventually settled. Furthermore, $d(u) = \text{dist}_G(s, u)$ at the latest from when its predecessor on some shortest path from s to u is settled. †

Proof: Consider a shortest path p from s to some vertex $u \in V$. We employ induction on the number of arcs in p.

$|p| = 0$: If $|p| = 0$, then $u = s$ and there is nothing to show, since s gets settled in the very first iteration and $d(s)$ is initialised to zero in line 2.

$|p| > 0$: Decompose p into $p = q \cdot (v, u)$ for some shortest path q from s to v. By induction hypothesis, v is eventually settled and $d(v) = \mathrm{dist}_G(s, v)$ during this very iteration. Therefore,

$$d(u) \leq d(v) + \mathrm{len}_G(vu) = \mathrm{dist}_G(s, v) + \mathrm{len}_G(vu) = \mathrm{dist}_G(s, u)$$

regardless of whether $d(u)$ is updated in line 9 or not. By № 1.1, part (c), it thus follows that $d(u) = \mathrm{dist}_G(s, u)$.

It only remains to show that u will eventually be settled. To this end, note that if $u \notin Q$, then u is inserted into Q when v is settled. However, each vertex in the queue eventually gets settled by № 1.1. □

We have the following immediate corollary of № 1.2, which we spell out for the sake of completeness.

1.3 Corollary: Dijkstra's algorithm is correct in the sense that it terminates and that $d(u) = \mathrm{dist}_G(s, u)$ for all $u \in V$ after its termination. †

The running time of Dijkstra's algorithm depends largely on how the choice of $u \in Q$ with minimal tentative distance $d(u)$ in line 5 is implemented. The best known method for non-negative real arc lengths – called Fibonacci heaps – is due to Fredman and Tarjan [FT87], with which Dijkstra's algorithm has time complexity $\mathcal{O}(n \cdot \log(n) + m)$, where n denotes the number of vertices and m denotes the number of arcs in G. Ahuja et al. [AMOT90] observed that this upper bound on the performance of Dijkstra's algorithm is optimal in a comparison based model of computation and thus can only be beaten if one imposes further restrictions on the input. If one allows for example only integral arc lengths, there are several data structures with which Dijkstra's algorithm has running time asymptotically less than $\mathcal{O}(n \cdot \log(n) + m)$. To the author's knowledge, the best known such data structure is due to Thorup [Tho03], who has shown that one may achieve a running time of $\mathcal{O}(n \cdot \log\log(n) + m)$ or $\mathcal{O}(n \cdot \log\log(C) + m)$, where $C = \max_{e \in A} \mathrm{len}_G(e)$.

Apart from these enhancements regarding the data structures used in the implementation of Dijkstra's algorithm, there are other, more conceptual improvements. If one restricts the attention to undirected graphs with integral edge weights, there is an algorithm with time complexity $\mathcal{O}(n + m)$ due to Thorup [Tho99]. This linear-time algorithm is no variant of Dijkstra's algorithm but actually based on a slightly different approach. However, it is not clear, whether this approach may be adapted to directed graphs or not. For directed graphs with positive, real arc lengths, there are other improvements targeted at the computation of only a single distance $\mathrm{dist}_G(s, t)$ instead of the distances $\mathrm{dist}_G(s, u)$ for all $u \in V$. Contraction hierarchies – the very subject of the thesis at hand – are one of these.

Another such improvement, which is also of relevance for our work, is called *bidirectional Dijkstra's algorithm*. Instead of only searching for shortest paths with

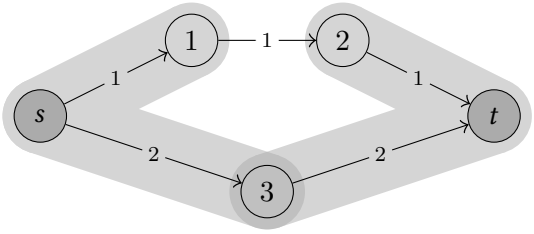


Figure 1.1: An example of a graph, for which the correct distance $\mathrm{dist}_G(s,t)$ is not known when the vertex 3 is first to be discovered by both queries of bidirectional Dijkstra's algorithm. Settled vertices are drawn in grey, while the relaxed arcs and the discovered vertices are drawn on a background shaded in lighter grey.

source vertex s, one may additionally look for shortest paths with target vertex t. If these two queries meet, one may compute the distance from s to t as the sum of $\mathrm{dist}_G(s,u)$ and $\mathrm{dist}_G(u,t)$ for some appropriate vertex u. One has to be careful though since a single vertex u discovered by both queries is usually not suited for the computation of $\mathrm{dist}_G(s,t)$ from $\mathrm{dist}_G(s,u)$ and $\mathrm{dist}_G(u,t)$. This phenomenon may also be seen in Figure 1.1, where the first vertex to be discovered by both queries is vertex 3, which does not even lie on a shortest s-t-path. One solution to this problem is not to abort, when an arbitrary vertex is discovered by both queries, but rather when one knows that all other vertices not yet settled cannot possibly lie on a path shorter than the ones already discovered. This idea may be implemented using the fact that the minimum tentative distances $M_s = \min_{x \in Q_s} d_s(x)$ and $M_t = \min_{x \in Q_t} d_t(x)$ of the vertices in the respective queues Q_s and Q_t are increasing with each iteration of Dijkstra's algorithm. The resulting algorithm is the announced bidirectional Dijkstra's algorithm. A rough description in pseudo code may be seen in Algorithm 1.2.

Although there is no asymptotic gain of performance when compared to Dijkstra's algorithm, bidirectional Dijkstra's algorithm is preferable in actual applications and additionally serves as a building block for more sophisticated shortest-path algorithms like contraction hierarchies. We close this chapter with a brief proof that Algorithm 1.2 is correct and one further remark on its performance.

1.4 Proposition: Bidirectional Dijkstra's algorithm (Algorithm 1.2) is correct. †

Proof: We use the notation from Algorithm 1.2. Let us first suppose that the algorithm terminates because both queues Q_s and Q_t are empty. In this case, there is no path from s to t in G since the original algorithm of Dijkstra is correct and one of the queries would eventually have found the source of the respective other query.

Let us now suppose that the algorithm terminates because there is some vertex $u \in V$, such that $d_s(u) + d_t(u) \leq M_s + M_t$. Choose a shortest path p from s to t. Further choose $v \in p$, such that v is settled in the query with source s and such that $\mathrm{dist}_G(s,v)$ is maximal with respect to this property. Let w denote the successor of v on p. Note

Algorithm 1.2: Bidirectional Dijkstra's algorithm

Data: Weighted digraph $G = (V, A)$
Input: Source vertex $s \in V$, target vertex $t \in V$
Output: Distance $\operatorname{dist}_G(s, t)$

/ Run one instance of Dijkstra's algorithm with source s in G and one instance with source t in G with all arcs reversed. Denote the queues by Q_s and Q_t and the tentative distances by $d_s(-)$ and $d_t(-)$, respectively. Furthermore, let $M_s = \min_{x \in Q_s} d_s(x)$ and $M_t = \min_{x \in Q_t} d_t(x)$. */*

1 **while** $\min_{u \in V} d_s(u) + d_t(u) > M_s + M_t$ *and* $Q_s \cup Q_t \neq \emptyset$ **do**
2 Choose one instance x of Dijkstra's algorithm, such that $Q_x \neq \emptyset$
3 Perform one single iteration in the chosen instance
4 **return** $\min_{u \in V} d_s(u) + d_t(u)$

that by our choice of v, its successor w on the path p cannot possibly be settled in the query starting from s, hence $d_s(w) \geq M_s$. On the other hand, the predecessor v of w on a shortest path from s to w is settled, so that $\operatorname{dist}_G(s, w) = d_s(w)$ by № 1.2. Thus, $\operatorname{dist}_G(s, w) \geq M_s$, which gives

$$\begin{aligned} \operatorname{dist}_G(s, w) + M_t \geq M_s + M_t &\geq d_s(u) + d_t(u) \\ &\geq \operatorname{dist}_G(s, t) = \operatorname{dist}_G(s, w) + \operatorname{dist}_G(w, t) \end{aligned}$$

and hence

$$M_t \geq \operatorname{dist}_G(w, t)\,.$$

Note that this last inequality implies $d_t(w) = \operatorname{dist}_G(w, t)$, because any predecessor of w on a reversed shortest path from w to t has to be settled already. Altogether, we therefore have

$$d_s(w) + d_t(w) = \operatorname{dist}_G(s, w) + \operatorname{dist}_G(w, t) = \operatorname{dist}_G(s, t)\,,$$

which finishes the proof since $d_s(x) + d_t(x) \geq \operatorname{dist}_G(s, t)$ for all $x \in V$. ▫

Note that the strategy by which an instance of Dijkstra's algorithm is chosen in line 2 has considerable impact on the performance of Algorithm 1.2 while it was completely irrelevant in Algorithm 1.1. However, there is a strategy which is guaranteed to be within a constant factor of the optimal strategy for each pair of input vertices s and t.

1.5 Remark: Suppose that in line 2 of Algorithm 1.2 the two instances are chosen alternately. Then the number of vertices settled after termination is at most twice the number of vertices settled with any other strategy. †

Proof: Consider an optimal choice consisting of k iterations and suppose that the instance with source s is chosen σ times. The instance with source t is then chosen $k-\sigma$ times and the algorithm with both instances chosen alternately terminates after at most $2 \cdot \max\{\sigma, k-\sigma\}$ iterations. □

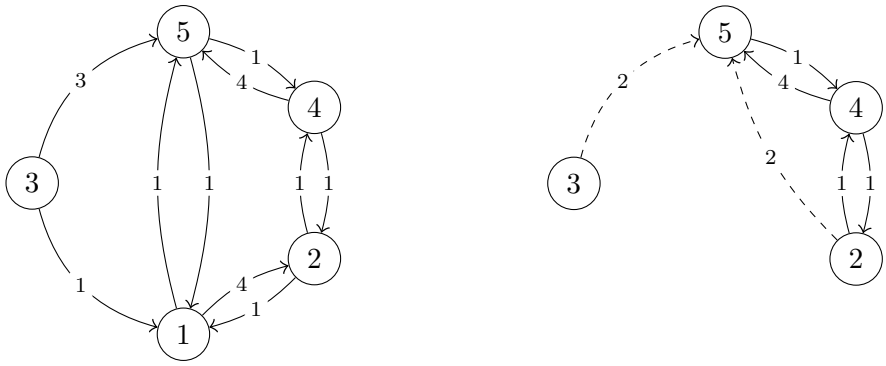
2 Modelling Contraction Hierarchies

Contraction hierarchies were originally introduced and studied by Geisberger [Geio7] and Geisberger et al. [GSSDo8]. They gave an iterative algorithm to compute two weighted digraphs $(\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$ from a weighted digraph $G = (V, A)$ and an order α on its vertices. Geisberger et al. showed that one may compute distances and even shortest paths in G using a variant of bidirectional Dijkstra's algorithm on $\bar{G}_\alpha^\wedge$ and $\bar{G}_\alpha^\vee$. We motivate and recall these results and take the opportunity to introduce notation and terminology. Moreover, we complement the ideas of Geisberger et al. with our own observations and develop a more concise definition of contraction hierarchies, which is also the reason why we denote the contraction hierarchies of [GSSDo8; Geio7] by $\bar{G}_\alpha$ instead of the simpler G_α.

2.1 Algorithmic Approach to Contraction Hierarchies

Fix a weighted digraph $G = (V, A)$ with $n = |V|$ vertices and let $v \in V$. Suppose we want to remove v from G but preserve the distances between all vertices distinct from v. If we simply delete v and its incident arcs from G, there may be vertices s and t whose distance $\mathrm{dist}_G(s, t)$ grows. More precisely, this happens if and only if v lies on all shortest paths from s to t. Any such unique shortest path p from s to t passing v necessarily contains an arc uv with target v and an arc vw with source v. Observe that the path (u, v, w) is itself such a unique shortest path, for otherwise p would be either not unique or no shortest path. In order to preserve the distances between all vertices distinct from v, it thus suffices to insert a new arc uw of length $\mathrm{len}_G(uv) + \mathrm{len}_G(vw)$ for each two arcs uv and vw as above; One can then replace any occurrence of (u, v, w) in a shortest s-t-path by the corresponding arc uw of equal length. This deletion of v together with the insertion of an arc uw for each unique shortest path (u, v, w) is called *contraction* of v. We denote the graph G after contraction of v by $G(v) = (V_v, A_v)$. The possibly new arcs uw are called *shortcuts*. An explicit example of contraction is depicted in Figure 2.1. Algorithm 2.1 is a more succinct description of contraction in pseudo code.

There is a subtle point to contraction, which we did not mention explicitly but which lies hidden in the formulation of Algorithm 2.1. Contraction does not necessarily preserve all arc lengths of G, for the length of an arc uw is "overwritten" if contraction causes the insertion of a shortcut with the very same source and target vertices. As the length $\mathrm{len}_{G(v)}(uw)$ of any shortcut uw is nothing but the length of a shortest path (u, v, w) from u to w in G, we at least have the inequality $\mathrm{len}_{G(v)}(xy) \leq \mathrm{len}_G(xy)$



(a) A weighted digraph G on five vertices.

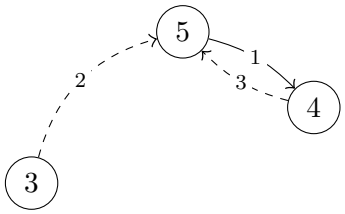
(b) The digraph G after contraction of 1.

Figure 2.1: Contraction of a single vertex. Figure 2.1a shows a digraph G on five vertices with arc lengths drawn on the respective arcs. In Figure 2.1b one may see G after contraction of the vertex 1. Shortcuts are drawn dashed and arc lengths are again drawn on the respective arcs.
Note that there are four paths that have to be taken into consideration during contraction of the vertex 1. Of these, only $(3, 1, 5)$ and $(2, 1, 5)$ are unique shortest paths, while $(3, 1, 2)$ is a shortest path but not unique and $(5, 1, 2)$ is not even a shortest path.

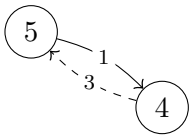
Algorithm 2.1: Contraction of a single vertex

Input: Weighted digraph $G = (V, A)$, a vertex $v \in V$
Output: $G(v) = (V_v, A_v)$, which is G after contraction of v

1 $V_v \leftarrow V \setminus \{v\}$
2 $A_v \leftarrow \left\{ xy \in A \,\middle|\, v \notin \{x, y\} \right\}$
3 $\mathrm{len}_{G(v)} \leftarrow \mathrm{len}_G$ restricted to A_v
4 **foreach** $uv, vw \in A$ **do**
5 **if** (u, v, w) *is the only shortest path between* u *and* w **then**
6 $A_v \leftarrow A_v \cup \{uw\}$
7 $\mathrm{len}_{G(v)}(uw) \leftarrow \mathrm{len}_G(uv) + \mathrm{len}_G(vw)$
8 **return** $G(v) = (V_v, A_v)$



(a) The graph shown in Figure 2.1b after contraction of the vertex 2.



(b) The digraph shown to the left after contraction of the vertex 3.

Figure 2.2: Iterated contraction of vertices of the graph shown in Figure 2.1a. Shortcuts are drawn dashed. The order α on V is assumed to be given by the values $\alpha(v)$ drawn inside the respective nodes. The graphs $G = G(1)$ and $G(2)$ are depicted in Figure 2.1a and 2.1b, respectively, while Figure 2.2a and 2.2b show the graphs $G(3)$ and $G(4)$. Note that $G(5)$ consists of a single vertex only and is thus not depicted.

for all arcs xy of G with $v \notin \{x, y\}$. We commonly refer to this fact by saying that contraction does not increase arc lengths. This phenomenon also appears in Figure 2.1, where the arc $3 \to 5$ has length 3 in Figure 2.1a but length 2 after contraction of the vertex 1 in Figure 2.1b.

Contraction of vertices may be iterated until the remaining graph is eventually empty. The structure of the intermediate graphs depends crucially on the order in which this successive contraction of vertices takes place. We therefore equip our graph G with an *order* α on its vertices, that is, with a bijective map $\alpha\colon V \to [n]$, where $[n]$ denotes the set $\{1, \dots, n\}$. Given such an order α and two vertices u and v, we say that u is *below* v or that v is *above* u, if $\alpha(u) \leq \alpha(v)$. We further say that u is *strictly below* v or that v is *strictly above* u, if u is below v and additionally $u \neq v$. We always assume that contraction proceeds from lower to higher vertices, i.e. a vertex v is contracted only after all vertices u with $\alpha(u) < \alpha(v)$ have already been contracted. We denote the graph G after the $(i-1)$-th contraction by $G(i) = (V_i, A_i)$. More precisely, we let

$$G(i) = \begin{cases} G & \text{if } i = 1 \\ G(i-1)(v) \text{ where } v = \alpha^{-1}(i-1) & \text{if } 1 < i \leq n \end{cases}$$

If one assumes that the vertices in Figure 2.1a are labelled with their respective values $\alpha(v)$, then Figure 2.1a and 2.1b show nothing but $G(1)$ and $G(2)$. The next two steps $G(3)$ and $G(4)$ for this particular example are depicted in Figure 2.2. Note that the graph $G(5)$ already consists of a single vertex only.

The fact that contraction of a single vertex $v = \alpha^{-1}(1)$ preserves the distances between all vertices distinct from v was our very motivation to study contraction

of vertices. We argued that one obtains a shortest path in $G(v)$ from a shortest path in G by replacing any occurrence of a unique shortest path (u, v, w) with the corresponding shortcut uw. We now want to make these arguments precise not only for a single contraction $G(v)$ but for the whole chain $G(1), G(2), \ldots, G(n)$ of contractions. Interestingly, a formal proof of our intuition is surprisingly involved and it seems necessary – or at least convenient – to additionally consider the problem of finding a path p_1 in $G = G(1)$ that corresponds to a given path p_k in one of the $G(k)$. This may be accomplished by successively unfolding shortcuts in p to the shortest path that led to their insertion. As an example, consider the path $(4, 5)$ of length 3 in the digraph shown in Figure 2.2a. The shortcut $4 \to 5$ was inserted upon contraction of the vertex 2 as a replacement for the path $(4, 2, 5)$ in Figure 2.1b. Moreover, the arc $2 \to 5$ in this latter path is also a shortcut that got inserted during contraction of the vertex 1 to replace the path $(2, 1, 5)$ in Figure 2.1a. Altogether, we see that if we replace the only arc in $(4, 5)$ with the path $(4, 2, 5)$, we obtain the path $(4, 2, 5)$, in which we may replace the arc $2 \to 5$ with the path $(2, 1, 5)$ to obtain a path $(4, 2, 1, 5)$ of the same length 3 as our initial path $(4, 5)$ in $G(3)$. As can also be seen in this example, unfolding of shortcuts to the corresponding paths does not change the source and target of the path in question since the unfolded paths always have the same endpoints as the original shortcut.

All our informal arguments from above concerning unfolding of shortcuts uw to shortest paths (u, v, w) and concerning the substitution of shortest paths (u, v, w) by their corresponding shortcuts uw are finally made precise in terms of the following lemma.

2.1 Lemma: Let s and t be two vertices and let $k = \min\{\alpha(s), \alpha(t)\}$.

(a) For any path p in $G(k)$ from s to t, there exist paths $p = p_k, p_{k-1}, \ldots, p_1$, such that $p_i \subseteq G(i)$ and such that the length of each p_i equals the length of p.

(b) For any shortest path p in G from s to t, there are shortest paths $p = p_1, \ldots, p_k$, such that $p_i \subseteq G(i)$ and $p \cap V_i \subseteq p_i$ and such that the length of each p_i equals the length of p.

In particular, $\mathrm{dist}_{G(k)}(s, t) = \mathrm{dist}_G(s, t)$ for all vertices s and t in $G(k)$. †

Proof:

(a) Beginning with $p_k = p$, we inductively construct the paths p_i as claimed. Suppose that $p_k, \ldots, p_i$ for some $i > 1$ are already given. Let $v = \alpha^{-1}(i - 1)$, so that $G(i) = G(i - 1)(v)$. Further let S denote the shortcuts inserted upon contraction of v. We distinguish the following cases.

$p_i \cap S = \emptyset$: If $p_i \subseteq G(i)$ contains no shortcut $uw \in S$, then all the arcs of p_i are already present in $G(i - 1)$. We may therefore simply choose $p_{i-1} = p_i$. Note that $\mathrm{len}_{G(i-1)}(p_{i-1}) = \mathrm{len}_{G(i)}(p_i)$ since during passage from $G(i-1)$ to $G(i)$ only the arc lengths of shortcuts change.

$p_i \cap S \neq \emptyset$: We obtain a path p_{i-1} in $G(i-1)$ from p_i by replacing all the shortcuts $uw \in S$ with the corresponding shortest path (u, v, w) that led to their insertion. Recall that the arc length of a shortcut uw is given by

$$\text{len}_{G(i)}(uw) = \text{len}_{G(i-1)}(uv) + \text{len}_{G(i-1)}(vw)\,.$$

The new subpaths (u, v, w) of p_{i-1} therefore have the same length as the arcs uw of p_i that they replace. As the length of any arc of p_{i-1} that is no shortcut remains unchanged during contraction of v, it follows that $\text{len}_{G(i-1)}(p_{i-1}) = \text{len}_{G(i)}(p_i)$.

(b) According to (a), all paths from s to t in $G(i)$ induce paths from s to t in G of the same length. If p is a shortest path from s to t in G, it follows that any path p_i from s to t in $G(i)$ has length at least $\text{len}_G(p)$. In order to establish that p_i is a shortest path in $G(i)$ of length $\text{len}_G(p)$ it therefore suffices to show that $\text{len}_{G(i)}(p_i) \leq \text{len}_G(p)$. This observation will be used several times in our following proof of part (b)

Starting with $p_1 = p$, we inductively construct the shortest paths p_i in $G(i)$. Suppose that $p_1, \ldots, p_i$ are given for some $1 \leq i < k$ and let $v = \alpha^{-1}(i)$, so that $G(i+1) = G(i)(v)$. Note that since p_i is a shortest path, it contains at most one subpath of the form (u, v, w). We distinguish the following cases.

p_i does not contain v: Since v is the only vertex of $G(i)$ that is not contained in $G(i+1)$, we find that p_i already is a path in $G(i+1)$. Furthermore, our assumption $p \cap V_i \subseteq p_i$ immediately yields $p \cap V_{i+1} \subseteq p_i$. Since arc lengths do not grow during contraction, we additionally have

$$\text{len}_{G(i+1)}(p_i) \leq \text{len}_{G(i)}(p_i) = \text{len}_G(p)$$

and we may therefore simply choose $p_{i+1} = p_i$.

p_i contains a unique shortest path (u, v, w) as subpath: In this case, we may more or less apply our informal argument for the contraction of a single vertex from above: If (u, v, w) is a unique shortest path, then contraction of v leads to the insertion of a shortcut uw. Recall that the length of uw in $G(i+1)$ is given by $\text{len}_{G(i+1)}(uw) = \text{len}_{G(i)}(uv) + \text{len}_{G(i)}(vw)$, so that we may replace the subpath (u, v, w) of p_i by the shortcut uw of the same length to obtain a path p_{i+1} in $G(i+1)$. Since contraction of v does not increase any of the lengths of the remaining arcs, we find that $\text{len}_{G(i+1)}(p_{i+1}) \leq \text{len}_{G(i)}(p_i) = \text{len}_G(p)$. Furthermore, we only removed the vertex v from p_i, so that $p_{i+1} = p \cap V_{i+1}$ follows from $p_i = p \cap V_i$.

p_i contains a non-unique shortest path (u, v, w) as subpath: Although we completely ignored this case in our informal arguments above, it is not entirely

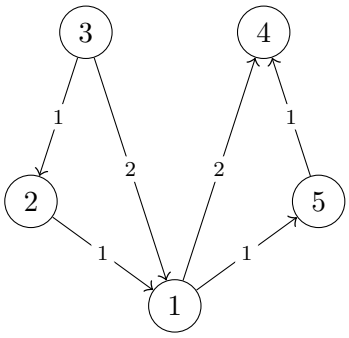


Figure 2.3: Non-unique shortest paths from the vertex 3 to the vertex 4, such that each of them contains the vertex 1. Note that the shortest path $(3, 2, 1, 5, 4)$ with a maximal number of arcs, contains the unique shortest path $(2, 1, 5)$ as subpath. This is not just a coincidence but also shown to be true in general in the proof of № 2.1.

trivial. If the subpath (u, v, w) of p_i is non-unique, one may of course replace it with another shortest path q from u to v of the same length. However, as can also be seen in Figure 2.3, there is no guarantee that this path q does not contain the vertex v. Nevertheless, if one chooses the shortest u-v-path q with some care, then it either does not contain v or contains a unique shortest path (x, v, y), so that our arguments of above apply.

More precisely, let us choose q such that it is the shortest path from u to v with a maximal number of arcs. Let us further assume that $v \in q$. In this case, q contains a shortest path (x, v, y) as subpath. If (x, v, y) is not unique, i.e. if there is another shortest path r from x to y, then r consists of at least three arcs since $r \neq (x, v, y)$. If one replaces the subpath (x, v, y) of q with the path r, one therefore obtains a shortest path from u to v consisting of at least $|q| + 1$ arcs. This contradicts our choice of q, so that we may conclude that (x, v, y) is a unique shortest path.

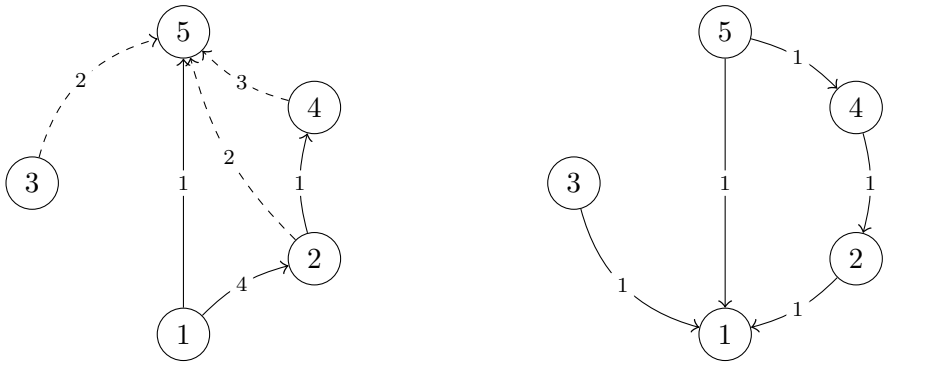
If one replaces the subpath (u, v, w) of p_i with the path q, one thus either obtain a path p_i' with $v \notin p_i'$ or a path p_i' containing a unique shortest path (x, v, y) as subpath. In either case, our previous arguments show that there is a shortest s-t-path $p_{i+1} \subseteq G(i + 1)$ of length equal to that of p_i' such that v is the only vertex possibly contained in p_i' but not in p_{i+1}. As both (u, v, w) and q are shortest paths, it follows that p_i' and p_i have the same length and thus $\text{len}_{G(i+1)}(p_{i+1}) = \text{len}_G(p)$, too. Furthermore, v is the only vertex that is contained in p_i but possibly not in p_i', so that $p \cap V_{i+1} \subseteq p_{i+1}$ follows from $p \cap V_i \subseteq p_i$. Altogether, we see that p_{i+1} is in fact the desired path from s to t in $G(i + 1)$. ▫

Let us now try to motivate the construction of the digraphs $\bar{G}_\alpha^\wedge$ and $\bar{G}_\alpha^\vee$ of Geisberger et al. mentioned at the very beginning of this paragraph. Consider the problem of computing the distance $\mathrm{dist}_G(s,t)$ between two vertices s and t. We have just shown that contraction does not alter the distances between all the remaining vertices and it therefore suggests itself that in order to compute $\mathrm{dist}_G(s,t)$ it suffices to consider the possibly smaller graph $G(k)$, where $k = \min\{\alpha(s), \alpha(t)\}$. Moreover, one might switch to an even smaller graph $G(l)$ once one has taken account of all the paths $(s = x_1, \ldots, x_r)$ and $(x_r, \ldots, x_1 = t)$ with $k \leq \alpha(x_1), \ldots, \alpha(x_r) \leq l$. Intuitively, this distance-query should outperform Dijkstra's algorithm, for one only has to deal with successively smaller and smaller graphs $G(l)$. However, storing all these graphs $G(l)$ is a tremendous overhead and one therefore rather works with something like the union of all $G(l)$ and ensures this process of "going upwards" by other means. More precisely, one stores two digraphs $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$, where $\bar{G}_\alpha^\wedge = (V, A_\alpha^\wedge)$ contains all arcs occurring in one of the $G(l)$ whose target vertex lies above its source vertex and $\bar{G}_\alpha^\vee = (V, A_\alpha^\vee)$ contains all arcs whose target vertex lies below its source vertex. Formally, $A_\alpha^\wedge$ and $A_\alpha^\vee$ are given by

$$\begin{aligned} A_\alpha^\wedge &= \left\{ uv \in \bigcup_{i=1}^{n} A_i \;\middle|\; \alpha(u) < \alpha(v) \right\} \\ A_\alpha^\vee &= \left\{ uv \in \bigcup_{i=1}^{n} A_i \;\middle|\; \alpha(u) > \alpha(v) \right\} \end{aligned} \tag{2.1}$$

The length of an arc uv in $A_\alpha^\wedge$ or $A_\alpha^\vee$ is given by the minimum over all $\mathrm{len}_{G(i)}(uv)$, where $uv \in A_i$. To not further complicate our notation, we denote this length by $\mathrm{len}_G^\alpha(uv)$ instead of $\mathrm{len}_{\bar{G}_\alpha^\wedge}(uv)$ or $\mathrm{len}_{\bar{G}_\alpha^\vee}(uv)$. Likewise, the length of a shortest path from u to v in one of the graphs $\bar{G}_\alpha^\wedge$ and $\bar{G}_\alpha^\vee$ is denoted by $\mathrm{dist}_G^\wedge$(u,v) and $\mathrm{dist}_G^\vee$(u,v), respectively. We call the pair $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$ the *contraction hierarchy* of G. In Figure 2.4 one may see the contraction hierarchy for our example from Figure 2.1 and Figure 2.2. Note that since all arcs in $\bar{G}_\alpha^\wedge$ point upwards and all arcs in $\bar{G}_\alpha^\vee$ point downwards, it follows that Dijkstra's algorithm visits only vertices in successively higher levels of our chain $G(1), \ldots, G(n)$ of digraphs if it is given $\bar{G}_\alpha^\wedge$ or $\bar{G}_\alpha^\vee$ with reversed arcs as input.

Contraction hierarchies may easily be computed from a given weighted digraph G and an order α on its vertices. For this purpose, one simply contracts all the vertices of G and picks up all the arcs of $\bar{G}_\alpha^\wedge$ and $\bar{G}_\alpha^\vee$ during the course of this computation. An implementation of this approach in pseudo code may be seen in Algorithm 2.2. This algorithm essentially is the one given in [GSSD08] and [Gei07], so that we have just recovered the definition of contraction hierarchies due to Geisberger et al. We are currently not concerned with the running time of Algorithm 2.2, but will return to it in Chapter 2.4, where there will be some results at our disposal that facilitate substantial simplifications of Algorithm 2.2.



(a) The upwards-directed digraph $\bar{G}_\alpha^\wedge$.

(b) The downwards-directed digraph $\bar{G}_\alpha^\vee$.

Figure 2.4: The contraction hierarchy $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$ associated with the digraph G shown in Figure 2.1a and the order α on its vertices indicated by the values $\alpha(v)$ drawn inside the respective nodes.

The reader should be aware that our considerations of № 2.1 do *not* apply to neither $\bar{G}_\alpha^\wedge$ nor $\bar{G}_\alpha^\vee$. Even though shortest paths in G correspond to shortest paths in each of the $G(i)$ and vice versa, separating the arcs pointing upwards from those pointing downwards does not preserve this nice property. This means, that one does not obtain a shortest path in G from a shortest path in $\bar{G}_\alpha^\wedge$ or $\bar{G}_\alpha^\vee$ by simply unfolding shortcuts as done in the proof of № 2.1. Moreover, the shortest paths $p_i \subseteq G(i)$ that we constructed in № 2.1 for any shortest path p in G may contain both arcs pointing upwards and arcs pointing downwards, so that they are generally neither paths in $\bar{G}_\alpha^\wedge$ nor in $\bar{G}_\alpha^\vee$. These phenomena also occur in the contraction hierarchy shown in Figure 2.4a, where the path $(1, 2, 4)$ is a shortest path of length 5 in $\bar{G}_\alpha^\wedge$ but $\operatorname{dist}_G(1, 4) = 2$ as may be seen in Figure 2.1a. Similarly, $(3, 1, 5, 4)$ is a shortest path of length 3 in G but neither $\bar{G}_\alpha^\wedge$ nor $\bar{G}_\alpha^\vee$ contains a path from 3 to 4.

2.2 Formal Approach to Contraction Hierarchies

Before we explore the relation between shortest paths in G and shortest paths in $\bar{G}_\alpha^\wedge$ and $\bar{G}_\alpha^\vee$ any further, we would like to give another characterisation of contraction hierarchies. We are particularly interested in a kind of "global" description of the arcs of $\bar{G}_\alpha$, for this would greatly simplify some arguments about contraction hierarchies. We feel obliged to point out that an earlier but far less sophisticated account of what follows below already appeared in previous work of the author [Colo9] and Bauer et al. [BCKKW10]. All arguments concerning the validity of our model as well as the extension to directed graphs are novel results, though.

Algorithm 2.2: Computation of a contraction hierarchy $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$

Input: Weighted digraph $G = (V, A)$, an order α on V
Output: Contraction hierarchy $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$

1 $A_\alpha^\wedge \leftarrow \emptyset$
2 $A_\alpha^\vee \leftarrow \emptyset$
3 **for** $i = 1$ **to** n **do**
  /* $V = V_i$ and $A = A_i$ */
4 $A_\alpha^\wedge \leftarrow A_\alpha^\wedge \cup \left\{ uv \in A \mid \alpha(u) < \alpha(v) \right\}$
5 $A_\alpha^\vee \leftarrow A_\alpha^\vee \cup \left\{ uv \in A \mid \alpha(u) > \alpha(v) \right\}$
6 $v \leftarrow \alpha^{-1}(i)$
7 Contract v /* See Algorithm 2.1 */
8 **return** $\bar{G}_\alpha^\wedge = (V, A_\alpha^\wedge)$ and $\bar{G}_\alpha^\vee = (V, A_\alpha^\vee)$

Recall that during contraction of v a shortcut st is inserted if and only if (s, v, t) is a unique shortest path. If one iterates contraction, it may happen that the arcs sv or vt are shortcuts themselves, so that the shortest path (s, v, t) as it occurs during contraction of v actually corresponds to a shortest path from s to t in G consisting of more than two arcs. Furthermore, it may be the case that there were initially many shortest paths between s and t but that all vertices on these shortest paths – except s, v and t – were already contracted. In order to find a criterion for the presence of a shortcut st in $\bar{G}_\alpha$ that only depends on G and α, we therefore have to drop the requirements that the shortest path causing the insertion of st is unique or consists of only two arcs. Nevertheless, for (s, v, t) to be a unique shortest path upon contraction of v, it is mandatory that there exists no shortest s-t-path in G containing a vertex w strictly above v and distinct from s and t; For when v gets contracted, this vertex w would not have been contracted yet and would still lie on a shortest s-t-path distinct from (s, v, t). However, any such vertex w will eventually be contracted, so that its contraction may also cause the insertion of st – provided that s or t does not get contracted beforehand. Summarising the above discussion, we see that a shortcut st is present in $\bar{G}_\alpha$ only if all vertices v distinct from s and t and additionally lying on a shortest s-t-path are contracted before s and t. As the order of contraction is given by our bijective map α on V, this is equivalent to saying that the only vertices that are above s or above t and that additionally lie on a shortest s-t-path are s and t themselves. We claim that this latter condition for the presence of a shortcut st in the contraction hierarchy is already sufficient.

Before turning to the actual proof, let us introduce a little notation. We denote by $P_\alpha(s, t)$ the set of vertices that are above s or above t and that lie on a shortest

path from s to t in G. One may formally define $P_\alpha(s,t)$ by

$$P_\alpha(s,t) = \Big\{ v \in V \;\Big|\; \alpha(v) \geq \min\{\alpha(s),\alpha(t)\} \text{ and } \operatorname{dist}_G(s,v) + \operatorname{dist}_G(v,t) = \operatorname{dist}_G(s,t) < \infty \Big\}. \tag{2.2}$$

The reader should note that our claim that $\bar{G}_\alpha$ contains a shortcut st if and only if the sole vertices above s or above t that lie on a shortest s-t-path are s and t themselves, is equivalent to the claim that $\bar{G}_\alpha$ contains a shortcut st if and only if $P_\alpha(s,t) = \{s,t\}$. With these abbreviations at hand, we are now able to succinctly state our criterion for the presence of a shortcut in $\bar{G}_\alpha$.

Theorem 1 *Global description of the contraction hierarchy $\bar{G}_\alpha = (\bar{G}^\wedge_\alpha, \bar{G}^\vee_\alpha)$*
Let $G = (V,A)$ be a weighted digraph and let $\alpha\colon V \to [n]$ be an order on its vertices. The arcs $A^\wedge_\alpha$ and $A^\vee_\alpha$ of $\bar{G}^\wedge_\alpha$ and $\bar{G}^\vee_\alpha$ are then precisely the sets

$$A^\wedge_\alpha = \Big\{ uv \in A \;\Big|\; \alpha(u) < \alpha(v) \Big\} \cup \Big\{ uv \;\Big|\; \alpha(u) < \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\} \Big\}$$
$$A^\vee_\alpha = \Big\{ uv \in A \;\Big|\; \alpha(u) > \alpha(v) \Big\} \cup \Big\{ uv \;\Big|\; \alpha(u) > \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\} \Big\}.$$

Furthermore, the length of a shortcut uv in $\bar{G}^\wedge_\alpha$ or $\bar{G}^\vee_\alpha$ is given by $\operatorname{dist}_G(u,v)$. ‡

Proof: Recall from (2.1) that $A^\wedge_\alpha \cup A^\vee_\alpha$ is a partition of all the arcs $\bigcup A_i$ occurring in one of the intermediate contractions $G(i) = (V_i, A_i)$. Since $A = A_1$, it follows from the definition of $A^\wedge_\alpha$ and $A^\vee_\alpha$ that

$$\Big\{ uv \in A \;\Big|\; \alpha(u) < \alpha(v) \Big\} \subseteq A^\wedge_\alpha \qquad \text{and} \qquad \Big\{ uv \in A \;\Big|\; \alpha(u) > \alpha(v) \Big\} \subseteq A^\vee_\alpha.$$

Furthermore, any arc $st \in A^\wedge_\alpha \setminus A$ or $st \in A^\vee_\alpha \setminus A$ is a shortcut. It therefore suffices to show that each shortcut st satisfies $P_\alpha(s,t) = \{s,t\}$ and that there is an arc st contained in one of the A_i for any two vertices s and t with $P_\alpha(s,t) = \{s,t\}$.

To this end, let us first spell out our discussion that lead us to this very claim. Let s and t be two vertices with $\operatorname{dist}_G(s,t) < \infty$, let $k = \min\{\alpha(s),\alpha(t)\}$ and assume $P_\alpha(s,t) \neq \{s,t\}$. We may then choose some vertex $v \in P_\alpha(s,t) \setminus \{s,t\}$. By the definition of $P_\alpha(s,t)$, there exists a shortest path p from s to t in G that contains v. Since, furthermore, $\alpha(v) > k$ by our choice of v, it follows from № 2.1 that there is for each $i \in \{1,\dots,k\}$ some shortest path p_i from s to t in $G(i)$ that also contains v. Observe that $\alpha(w) < k$ for any vertex w that is contracted before both s and t. Any shortest path (s,w,t) encountered during contraction of such a vertex w therefore cannot be unique, for there are still the shortest s-t-paths p_i containing v and hence distinct from (s,w,t). Consequently, if $P_\alpha(s,t) \neq \{s,t\}$, then st cannot be a shortcut and any shortcut st therefore satisfies $P_\alpha(s,t) = \{s,t\}$.

Now consider any two vertices s and t, such that $P_\alpha(s,t) = \{s,t\}$. As above, we let $k = \min\{\alpha(u),\alpha(v)\}$. We further choose a shortest path p from s to t in G. By № 2.1

there exists a shortest s-t-path p_k in $G(k)$ of length $\mathrm{len}_G(p)$. We claim that p_k in fact consists of a single arc only, so that p_k is the sought shortcut. Assume the contrary, i.e. that p_k contains a vertex v distinct from both s and t. Then v lies strictly above s or strictly above t since v is a vertex of $G(k)$. Furthermore, v lies on a shortest s-t-path in $G(k)$ and by № 2.1 there is then a shortest path in G that also contains v. Altogether, this implies $v \in P_\alpha(s,t)$, which contradicts our assumption $P_\alpha(s,t) = \{s,t\}$. Thus, p_k actually consists only of the arc st whose existence we claimed. Moreover, № 2.1 implies that the length of st in G_α is equal to $\mathrm{dist}_G(s,t)$ as $p_k = (s,t)$ has the same length as the shortest path p from s to t in G. □

This theorem provides a concise description of the arcs of a contraction hierarchy $\bar{G}_\alpha$. Moreover, not only does it shed some light on the structure of contraction hierarchies, but also suggests an alternative definition. Consider any arc st of G that is no unique shortest path. Removing it from G does not change any distances, for there still remains a shortest path from s to t. On the other hand, if st is a unique shortest path, then $P_\alpha(s,t) = \{s,t\}$ anyway. Therefore, it thus turns out that one might equally well work with the following alternative definition of contraction hierarchies.

For a given weighted graph $G = (V,A)$ and an order α on its vertices, we thus define $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$, where $G_\alpha^\wedge = (V, A_\alpha^\wedge)$ and $G_\alpha^\vee = (V, A_\alpha^\vee)$, by

$$\begin{aligned} A_\alpha^\wedge &= \left\{ uv \;\middle|\; \alpha(u) < \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\} \right\} \\ A_\alpha^\vee &= \left\{ uv \;\middle|\; \alpha(u) > \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\} \right\}. \end{aligned} \tag{2.3}$$

Inspired by Theorem 1 we further define arc lengths $\mathrm{len}_G^\alpha(uv)$ on $A_\alpha^\wedge$ and $A_\alpha^\vee$ by

$$\mathrm{len}_G^\alpha(uv) = \mathrm{dist}_G(u,v)\,. \tag{2.4}$$

As for $\bar{G}_\alpha$, we denote the distance from u to v in $G_\alpha^\wedge$ and $G_\alpha^\vee$ by $\mathrm{dist}_G^\wedge$(u,v) and $\mathrm{dist}_G^\vee$(u,v), respectively. We call this pair $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$ of directed, weighted graphs a *contraction hierarchy* of G, too. The reader awaiting some genuine examples of this new kind of contraction hierarchies will be put off until Figure 2.8, where we consider a digraph and one of its contraction hierarchies G_α that actually deviates from $\bar{G}_\alpha$.

For the rest of this work, we almost exclusively work with G_α instead of $\bar{G}_\alpha$. Nevertheless, we gain much of our intuition of contraction hierarchies from the iterative construction presented in Chapter 2.1. In the next few lines, we therefore show how to transfer some of the notions which arose during our inspection of the graphs $G(i)$ and $\bar{G}_\alpha$ to the contraction hierarchies G_α. Recall that an arc uw of $G(i)$ or $\bar{G}_\alpha$ is called a shortcut if it was newly inserted or its length was overwritten during contraction of a vertex v. Observe that the length of uw is overwritten if and only if $\mathrm{len}_G(uw) > \mathrm{dist}_G(u,w)$ and if all the shortest paths from u to w lie below u and v, i.e. if $P_\alpha(u,w) = \{u,w\}$. Inspired by these observations, we call *shortcuts* those arcs uw of G_α that are either not contained in G or satisfy $\mathrm{len}_G(uw) > \mathrm{dist}_G(u,w)$.

Given any such shortcut $uw \in A_\alpha^\wedge$ or $uw \in A_\alpha^\vee$, we are even able to recover the vertex whose contraction would have caused the insertion of uw in $\bar{G}_\alpha$. For this purpose, consider the set

$$S = \left\{ v \in V \smallsetminus \{u, w\} : \operatorname{dist}_G(u, v) + \operatorname{dist}_G(v, w) = \operatorname{dist}_G(u, w) \right\}. \tag{2.5}$$

Note that S is nonempty, for otherwise uw would be an arc of G of length $\operatorname{dist}_G(u, v)$. Further note that $\alpha(v) < \alpha(u)$ and $\alpha(v) < \alpha(w)$ for all $v \in S$, because any $v \in S$ with $\alpha(v) \geq \alpha(u)$ or $\alpha(v) \geq \alpha(w)$ would be an element of $P_\alpha(u, w) = \{u, w\}$. We call the vertex $v \in S$ with maximal $\alpha(v)$ the *supporting vertex* of uw. It is implicit in the proof of Theorem 1 that contraction of this supporting vertex v leads to the insertion of uw in $\bar{G}_\alpha$. Spelled out more precisely, the argument runs as follows: By the choice of v, there is a shortest path p from u to w that contains v. According to № 2.1, this path induces shortest paths p_i from u to w in $G(i)$ for each $1 \leq i \leq \alpha(v)$. Any shortest path (u, x, w) with $\alpha(x) < \alpha(v)$ is therefore not unique and contraction of such a vertex x cannot lead to the insertion of the arc uw. However, uw is an arc of $\bar{G}_\alpha$ by Theorem 1 and as v gets contracted last among all vertices lying on a shortest path from u to w, it follows that contraction of v causes the insertion of uw.

It already follows from the above argument that at least $\bar{G}_\alpha$ contains arcs uv and vw connecting the supporting vertex v with the endpoints of the shortcut uw. This fact may also be deduced from the mere definition of G_α and that of the supporting vertex, though. We include this argument, for we consider the development of contraction hierarchies from the definition of G_α to be interesting on its own. Our argument will make use of the following elementary observation.

2.2 Remark: Let G be a weighted digraph and let H be the digraph obtained from G by reversing all the arcs of G. Then $G_\alpha^\wedge = H_\alpha^\vee$ and $G_\alpha^\vee = H_\alpha^\wedge$. †

Proof: Write $P_\alpha^G(-,-)$ and $P_\alpha^H(-,-)$ to distinguish the sets $P_\alpha(-,-)$ in G from those in H. Observe that $\operatorname{dist}_G(s, t) = \operatorname{dist}_H(s, t)$ and hence $P_\alpha^G(s, t) = P_\alpha^H(t, s)$. Our claim now follows from the definition (2.3) of G_α and H_α. □

The above remark allows us to prove some statement about G_α by only taking $G_\alpha^\wedge$ into consideration, for the analogous statement for $G_\alpha^\vee$ follows from № 2.2 by reversing all the arcs. This strategy is exemplified in the proof of the subsequent lemma and will be exploited a many more times in what follows further below.

2.3 Lemma: Let uw be a shortcut in G_α and let v be its supporting vertex. Then G_α also contains arcs $uv \in A_\alpha^\vee$ and $vw \in A_\alpha^\wedge$. †

Proof: It suffices to prove the existence of $uv \in A_\alpha^\vee$, for the existence of $vw \in A_\alpha^\wedge$ then follows from № 2.2. Let us assume for the sake of contradiction that $uv \notin A_\alpha^\vee$. The construction of the supporting vertex v of uw implies $\operatorname{dist}_G(u, v) < \infty$ and hence $P_\alpha(u, v) \supseteq \{u, v\}$. As $uv \notin A_\alpha^\vee$, there is then a vertex $x \in P_\alpha(u, v)$ such

that $x \notin \{u, v\}$. Since $x \in P_\alpha(u, v) \setminus \{u, v\}$ and $\alpha(u) > \alpha(v)$, it follows that $\alpha(x) > \alpha(v)$ and

$$\operatorname{dist}_G(u, v) = \operatorname{dist}_G(u, x) + \operatorname{dist}_G(x, v)\,. \tag{2.6}$$

Moreover, it follows from $\alpha(x) > \alpha(v)$ and the definition of the supporting vertex that v lies on a shortest path from u to w but that no such shortest path contains the vertex x, i.e. that

$$\operatorname{dist}_G(u, w) = \operatorname{dist}_G(u, v) + \operatorname{dist}_G(v, w) \tag{2.7}$$

and

$$\operatorname{dist}_G(u, w) < \operatorname{dist}_G(u, x) + \operatorname{dist}_G(x, w)\,. \tag{2.8}$$

Using these three relations and the triangle inequality, one then computes

$$\begin{aligned}
\operatorname{dist}_G(u, w) &= \operatorname{dist}_G(u, v) + \operatorname{dist}_G(v, w) && \text{by (2.7)}\\
&= \operatorname{dist}_G(u, x) + \operatorname{dist}_G(x, v) + \operatorname{dist}_G(v, w) && \text{by (2.6)}\\
&\geq \operatorname{dist}_G(u, x) + \operatorname{dist}_G(x, w) && \text{by the triangle inequality}\\
&> \operatorname{dist}_G(u, w)\,. && \text{by (2.8)}
\end{aligned}$$

This is obviously contradictory. □

We coined the term *supporting arcs* for the arcs uv and vw whose existence is guaranteed by the preceding lemma. We also write $\sup(uw) = (uv, vw)$ to express that uv and vw are the supporting arcs of a given shortcut uw. Observe that if uv is a supporting arc of uw, then $\alpha(v) < \alpha(w)$ by the definition of the supporting vertex v of uw. So, even if uv is again a shortcut with supporting arcs, this chain of nested supporting shortcuts cannot descend indefinitely, for there are only finitely many vertices in G. Technically speaking, this allows us to do induction on the depth of nested shortcuts below a given arc uw. In order to make this precise, we define the *shortcut depth* $\operatorname{scd}(uw)$ of an arc uw of G_α by

$$\operatorname{scd}(uw) = \begin{cases} \operatorname{scd}(uv) + \operatorname{scd}(vw) & uw \text{ is a shortcut and } \sup(uw) = (uv, vw) \\ 1 & \text{otherwise.} \end{cases} \tag{2.9}$$

Note that $\operatorname{scd}(uw) \geq 1$ for all arcs uw of G_α. Further note that the shortcut depths of the supporting arcs uv and vw of any shortcut uw are strictly less than the shortcut depth of uw, so that we may indeed do induction on the shortcut depth scd.

Given only the definition of G_α, we just recovered the vertex v and the associated shortest path (u, v, w) whose contraction would lead to the insertion of a given shortcut uw of $\bar{G}_\alpha$. The length of uw in $\bar{G}_\alpha$ is then given by the sum of the lengths of the supporting arcs uv and vw. This is also true in G_α, where it follows immediately from our definition (2.4) of the arc lengths of G_α.

2.4 Remark: Let uw be a shortcut in G_α and let $\sup(uw) = (uv, vw)$. The length of uw is then given by $\mathrm{len}^\alpha_G(uw) = \mathrm{len}^\alpha_G(uv) + \mathrm{len}^\alpha_G(vw)$. †

Proof: Let $uw \in A^\wedge_\alpha$ be a shortcut and let $\sup(uw) = (uv, vw)$. Note that

$$\mathrm{len}^\alpha_G(uw) = \mathrm{dist}_G(u, w)\,, \quad \mathrm{len}^\alpha_G(uv) = \mathrm{dist}_G(u, v)\,, \quad \mathrm{len}^\alpha_G(vw) = \mathrm{dist}_G(v, w)$$

by our definition (2.4) of the arc lengths in G_α. Further recall from our construction (2.5) of the supporting vertex v that $\mathrm{dist}_G(u, w) = \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$, which completes the proof. □

In view of № 2.3 and № 2.4, G_α still possesses the most essential properties of $\bar{G}_\alpha$ that derived from its construction by iterated contraction of vertices. We will discuss a further generalisation of contraction hierarchies based upon these similarities between $\bar{G}_\alpha$ and G_α in Chapter 2.4, but are content with G_α for the moment.

2.3 Contraction Hierarchies and Shortest Paths

We saw above that even though shortest paths in the graphs $G(i)$ correspond to shortest path in G and vice versa, there is no such relation between shortest paths in G_α and those in G. In this paragraph we show how one can nonetheless utilise the contraction hierarchy G_α to compute distances and shortest paths in G. These considerations naturally lead to the notions of *search space* and *search space size* that play a predominant role in the remainder of this work.

Recall that the contraction hierarchy $\bar{G}_\alpha$ is nothing but the union of all the intermediate contractions $G(i)$, where the arcs pointing upwards are separated from those pointing downwards. Any shortest path from some source vertex s to some target vertex t in G induces a shortest s-t-path p of equal length in $G(k)$, where $k = \min\{\alpha(s), \alpha(t)\}$. One may decompose p into subpaths $p_1, \ldots, p_r$, such that each of the p_i consists either of arcs pointing upwards or of arcs pointing downwards only. With this decomposition of p, one finds that each p_i is a path in $\bar{G}^\wedge_\alpha$ or $\bar{G}^\vee_\alpha$. In order to determine the distance $\mathrm{dist}_G(s, t)$, it thus suffices to get hold of this decomposition $p = p_1 \cdot \cdots \cdot p_r$ of p. It even suffices to consider decompositions $p = p_1 \cdot p_2$ of p, so that one may actually compute them with a simple, bidirectional variant of Dijkstra's algorithm. The deployment of this very algorithm to compute $\mathrm{dist}_G(s, t)$ and a proof of its correctness are the main goals of this paragraph, while the rest of this work is devoted to the study of its performance.

We begin with a remark about the sets $P_\alpha(s, t)$ used in the definition of G_α. Essentially, these $P_\alpha(s, t)$ encode shortest paths from s to t and hence share some essential properties with them. The two single most important common features are the fact that any subpath q of a shortest path p is again a shortest path and that one obtains further shortest paths by replacing q in p with another shortest path q' having the

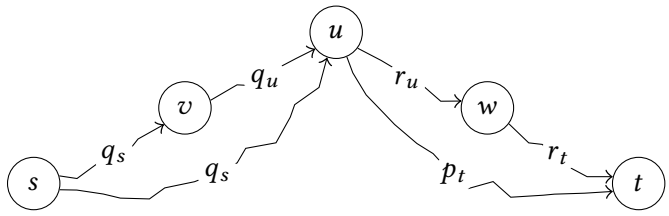


Figure 2.5: Sketch of all the shortest paths $p = p_s \cdot p_t$, $q = q_s \cdot q_u$ and $r = r_u \cdot r_t$ occurring in the proof of № 2.5.

same source and target as q. The following remark simply spells out these properties for the sets $P_\alpha(s,t)$.

2.5 Remark: Let $s, t \in V$ be two vertices and let $u \in P_\alpha(s,t)$.

(a) $P_\alpha(s,u) \subseteq P_\alpha(s,t)$ and $P_\alpha(u,t) \subseteq P_\alpha(s,t)$.

(b) If $u \neq t$, then $P_\alpha(s,u) \subsetneq P_\alpha(s,t)$ and if $u \neq s$, then $P_\alpha(u,t) \subsetneq P_\alpha(s,t)$.

(c) If $w \in P_\alpha(u,t)$, then $u \in P_\alpha(s,w)$. Likewise, $u \in P_\alpha(w,t)$ for all $w \in P_\alpha(s,u)$. †

Proof: Note that each claim consists of two completely symmetric statements. Hence, by № 2.2, it suffices to prove just one of them. The various paths and vertices occurring in our proof are also sketched in Figure 2.5.

(a) Consider any vertex $u \in P_\alpha(s,t)$, i.e. some vertex u that lies on a shortest path p from s to t and that additionally satisfies $\alpha(u) \geq \min\{\alpha(s), \alpha(t)\}$. Decompose the path p as $p = p_s \cdot p_t$, where p_s is a shortest path from s to u and p_t is a shortest path from u to t. The length of any shortest path q from s to u coincides with the length of p_s, as both are shortest paths. Any such shortest path q may therefore be prolonged by p_t in order to obtain another shortest path $q \cdot p_t$ from s to t. It follows that each vertex v that lies on a shortest path from s to u, also lies on a shortest path from s to t. Moreover, $\alpha(v) \geq \min\{\alpha(s), \alpha(u)\}$ and $\alpha(u) \geq \min\{\alpha(s), \alpha(t)\}$ imply $\alpha(v) \geq \min\{\alpha(s), \alpha(t)\}$. Altogether, we have just shown $P_\alpha(s,u) \subseteq P_\alpha(s,t)$.

(b) Let us now assume $t \in P_\alpha(s,u)$. As $t \in P_\alpha(s,u)$, there exists a decomposition $q = q_s \cdot q_u$ of any shortest path q from s to u into shortest paths q_s from s to t and q_t from t to u, respectively. Since u lies on a shortest s-t-path by assumption, $u = t$ is immediate. Consequently, if $u \neq t$, then $t \notin P_\alpha(s,u)$ and hence $P_\alpha(s,u) \subsetneq P_\alpha(s,t)$.

(c) Finally, let $w \in P_\alpha(u,t)$. There is a shortest path $r = r_u \cdot r_t$ from u to t, such that r_u is a shortest path from u to v and r_t is a shortest path from v to t.

Let q be a shortest path from s to u. Then $q \cdot r$ is a shortest path from s to t since $\mathrm{dist}_G(s,t) = \mathrm{dist}_G(s,u) + \mathrm{dist}_G(u,t)$ by our choice of $u \in P_\alpha(s,t)$. This implies in particular, that the subpath $q \cdot r_u$ of $q \cdot r$ is a shortest path, too. Furthermore, as $\alpha(w) \geq \min\{\alpha(u), \alpha(t)\}$ and $\alpha(u) \geq \min\{\alpha(s), \alpha(t)\}$, we have $\alpha(w) \geq \min\{\alpha(s), \alpha(u)\}$ and therefore $w \in P_\alpha(s,u)$. □

With these technical prerequisites at hand, we are now able to prove the main result concerning shortest paths in G and G_α. The following theorem is due to Geisberger et al. and is arguably the single most important theoretical result in [Gei07] and [GSSD08].

Theorem 2 *Relation between shortest paths in G and shortest paths in G_α*
Let $G = (V, A)$ be a weighted digraph, let α be an order on its vertices and let s and t be arbitrary vertices. Then

$$\mathrm{dist}_G(s,t) = \min_{u \in V} \mathrm{dist}^\wedge_G(s,u) + \mathrm{dist}^\vee_G(u,t)\,.$$

If additionally $\mathrm{dist}_G(s,t) < \infty$, then the minimum on the right hand side of above equation is always assumed by some $u \in P_\alpha(s,t)$ with $\alpha(u) > \min\{\alpha(s), \alpha(t)\}$, such that $\mathrm{dist}^\wedge_G(s,u) = \mathrm{dist}_G(s,u)$ and such that $\mathrm{dist}^\vee_G(u,t) = \mathrm{dist}_G(u,t)$. ‡

Before delving into the main part of the proof of Theorem 2, we would like to show that at least $\mathrm{dist}_G(s,t)$ does not exceed $\min_{u \in V} \mathrm{dist}^\wedge_G(s,u) + \mathrm{dist}^\vee_G(u,t)$. This inequality readily follows from our definition (2.4) of the arc lengths in G_α and will be used afterwards to prove that both values do indeed coincide.

2.6 Lemma: Let $G = (V, A)$ be a weighted digraph and let α be an order on its vertices. Then

$$\mathrm{dist}_G(s,t) \leq \min_{u \in V} \mathrm{dist}^\wedge_G(s,u) + \mathrm{dist}^\vee_G(u,t)$$

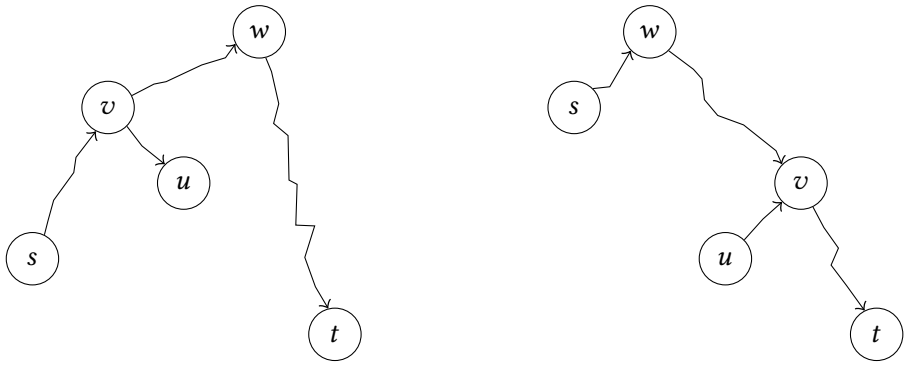
for all $s, t \in V$. †

Proof: Let $p = (s = x_1, \ldots, x_k = u)$ and $q = (u = x_k, \ldots, x_\ell = t)$ be paths in $G^\wedge_\alpha$ and $G^\vee_\alpha$, respectively. Recall from the definition (2.4) of the arc lengths $\mathrm{len}^\alpha_G(x_i x_{i+1})$ in G_α that they are given by $\mathrm{dist}_G(x_i, x_{i+1})$, so that the triangle inequality implies

$$\mathrm{len}(p) + \mathrm{len}(q) = \sum_{i=1}^{k-1} \mathrm{dist}_G(x_i, x_{i+1}) + \sum_{i=k}^{\ell-1} \mathrm{dist}_G(x_i, x_{i+1}) \geq \mathrm{dist}_G(s,t)\,.$$

As p and q were chosen arbitrarily, our claim follows. □

Observe that № 2.6 finishes the proof of Theorem 2 in the case $\mathrm{dist}_G(s,t) = \infty$, so that we may concentrate on the case $\mathrm{dist}_G(s,t) < \infty$ in what follows below. Note further that № 2.6 implies $\mathrm{dist}_G(s,t) \leq \mathrm{dist}^\wedge_G(s,t)$ for all s and t. We may therefore



(a) The vertex $u \in P_\alpha(s,t)$ lies above s.

(b) The vertex $u \in P_\alpha(s,t)$ lies below s.

Figure 2.6: Visualisation of the strategy used in the main part of the proof of Theorem 2.

infer equality of $\mathrm{dist}_G(s,t)$ and $\mathrm{dist}^\wedge_G(s,t)$ from only $\mathrm{dist}_G(s,t) \geq \mathrm{dist}^\wedge_G(s,t)$. This observation and the analogous statement for $\mathrm{dist}^\vee_G(s,t)$ will be used without further mention in the following proof.

Proof of Theorem 2**:** We want to show that there exists some vertex $u \in P_\alpha(s,t)$, such that

$$\mathrm{dist}_G(s,u) = \mathrm{dist}^\wedge_G(s,u) \qquad \text{and} \qquad \mathrm{dist}_G(u,t) = \mathrm{dist}^\vee_G(u,t)\,, \tag{2.10}$$

because any $u \in P_\alpha(s,t)$ satisfies $\mathrm{dist}_G(s,t) = \mathrm{dist}_G(s,u) + \mathrm{dist}_G(u,t)$. Moreover, we require that this vertex u may be chosen in such a way that $\alpha(u) > \min\{\alpha(s), \alpha(t)\}$.

The general strategy of the proof is as follows: We do induction on the cardinality of the set $P_\alpha(s,t)$, where the base clause $P_\alpha(s,t) = \{s,t\}$ turns out to be rather trivial since in this case $st \in A^\wedge_\alpha$ or $st \in A^\vee_\alpha$. In the induction step, we construct the desired vertex in the following manner. Choose $u \in P_\alpha(s,t) \setminus \{s,t\}$ and apply the induction hypothesis to s and u. This yields a vertex v together with paths from s to v in $G^\wedge_\alpha$ and from v to u in $G^\vee_\alpha$. We may then apply our induction hypothesis to t and v and obtain a vertex w and paths from v to w in $G^\wedge_\alpha$ and from w to t in $G^\vee_\alpha$. Verifying our claimed property (2.10) for this vertex w essentially amounts to assembling the various paths obtained during its construction. The relative positioning of these vertices s, t, u, v and w and the associated paths may be seen in Figure 2.6a. There is still one technical difficulty stemming from the fact that the vertex u does not necessarily lie strictly above s. As indicated in Figure 2.6b, this obstacle, too, may be overcome by simply applying the induction hypothesis to slightly different pairs of vertices. Let us now turn these suggestive sketches into rigorous proof:

$P_\alpha(s,t) = \{s,t\}$: By the definition of G_α, it is either $st \in A^\wedge_\alpha$ or $st \in A^\vee_\alpha$. If $st \in A^\wedge_\alpha$, then we may choose $u = t$ since

$$\mathrm{dist}_G(s,t) = \mathrm{len}^\alpha_G(st) \geq \mathrm{dist}^\wedge_G(s,t)\,.$$

and hence $\operatorname{dist}_G(s,t) = \operatorname{dist}^{\wedge}_G(s,t)$. Similarly, if $st \in A^{\vee}_{\alpha}$, then $u = s$ possesses all the claimed properties.

$P_\alpha(s,t) \supsetneq \{s,t\}$: Choose $u \in P_\alpha(s,t)$ such that $u \notin \{s,t\}$. As already indicated above, we need to distinguish the following two cases.

$\alpha(u) > \alpha(s)$: Then $P_\alpha(s,u)$ is a proper subset of $P_\alpha(s,t)$ by № 2.5, and our induction hypothesis implies that there is some vertex $v \in P_\alpha(s,u)$ such that

$$\begin{aligned} \operatorname{dist}^{\wedge}_G(s,v) &= \operatorname{dist}_G(s,v)\,, \\ \operatorname{dist}^{\vee}_G(v,u) &= \operatorname{dist}_G(v,u) \end{aligned} \tag{2.11}$$

and such that $\alpha(v) > \min\{\alpha(s),\alpha(u)\}$. Our assumption $\alpha(u) > \alpha(s)$ gives $\min\{\alpha(s),\alpha(u)\} = \alpha(s)$ and hence $v \neq s$. By № 2.5, $P_\alpha(v,t)$ is therefore a proper subset of $P_\alpha(s,t)$ and we may once again apply our induction hypothesis to obtain a vertex $w \in P_\alpha(v,t)$ such that

$$\operatorname{dist}^{\wedge}_G(v,w) = \operatorname{dist}_G(v,w)\,, \tag{2.12}$$

$$\operatorname{dist}^{\vee}_G(w,t) = \operatorname{dist}_G(w,t) \tag{2.13}$$

and such that $\alpha(w) > \min\{\alpha(v),\alpha(t)\}$. Using the earlier established inequality $\alpha(v) > \alpha(s)$, we obtain $\alpha(w) > \min\{\alpha(s),\alpha(t)\}$. Moreover, according to № 2.5, $v \in P_\alpha(s,u) \subseteq P_\alpha(s,t)$ and $w \in P_\alpha(v,t)$ together imply $v \in P_\alpha(s,w)$, hence $\operatorname{dist}_G(s,v) + \operatorname{dist}_G(v,w) = \operatorname{dist}_G(s,w)$. Employing (2.11) and (2.12), we compute

$$\begin{aligned} \operatorname{dist}^{\wedge}_G(s,w) &\leq \operatorname{dist}^{\wedge}_G(s,v) + \operatorname{dist}^{\wedge}_G(v,w) \\ &= \operatorname{dist}_G(s,v) + \operatorname{dist}_G(v,w) \\ &= \operatorname{dist}_G(s,w) \end{aligned}$$

This finishes this case as, by (2.13), $\operatorname{dist}^{\vee}_G(w,t) = \operatorname{dist}_G(w,t)$ is also satisfied.

$\alpha(u) < \alpha(s)$: Note that in this case $\min\{\alpha(s),\alpha(t)\} = \alpha(t)$ since our choice of $u \in P_\alpha(s,t)$ implies $\alpha(u) \geq \min\{\alpha(s),\alpha(t)\}$. Furthermore, $u \neq s$ by assumption and $P_\alpha(u,t)$ is thus a proper subset of $P_\alpha(s,t)$ by № 2.5. The induction hypothesis therefore implies the existence of some $v \in P_\alpha(u,t)$, such that

$$\begin{aligned} \operatorname{dist}^{\wedge}_G(u,v) &= \operatorname{dist}_G(u,v)\,, \\ \operatorname{dist}^{\vee}_G(v,t) &= \operatorname{dist}_G(v,t) \end{aligned} \tag{2.14}$$

and such that $\alpha(v) > \min\{\alpha(u),\alpha(t)\}$. Recall that $\alpha(u) \geq \alpha(t)$, which implies $\alpha(v) > \alpha(t)$.

Note that we would be finished if $v = s$ since (2.14) would then imply $\operatorname{dist}^{\vee}_G(s,t) = \operatorname{dist}_G(s,t)$ and $\alpha(s) > \alpha(t)$ is satisfied anyway. We may therefore assume that $v \neq s$. Since $v \in P_\alpha(u,t) \subseteq P_\alpha(s,t)$, this implies in particular, that $P_\alpha(s,v)$ is a proper subset of $P_\alpha(s,t)$. We may thus apply our induction hypothesis once again to obtain a vertex $w \in P_\alpha(s,v)$, such that

$$\operatorname{dist}^{\wedge}_G(s,w) = \operatorname{dist}_G(s,w)\,, \tag{2.15}$$
$$\operatorname{dist}^{\vee}_G(w,v) = \operatorname{dist}_G(w,v) \tag{2.16}$$

and such that $\alpha(w) > \min\{\alpha(s), \alpha(v)\}$. We already saw that $\alpha(v) > \alpha(t)$ and $\alpha(s) > \alpha(t)$, which together give $\alpha(w) > \min\{\alpha(s), \alpha(t)\}$. Furthermore, $w \in P_\alpha(s,v)$ implies $v \in P_\alpha(w,t)$ by № 2.5. This means in particular that $\operatorname{dist}_G(w,v) + \operatorname{dist}_G(v,t) = \operatorname{dist}_G(w,t)$ and hence

$$\begin{aligned}\operatorname{dist}^{\vee}_G(w,t) &\leq \operatorname{dist}^{\vee}_G(w,v) + \operatorname{dist}^{\vee}_G(v,t)\\ &= \operatorname{dist}_G(w,v) + \operatorname{dist}_G(v,t) \qquad \text{by (2.16) and (2.14)}\\ &= \operatorname{dist}_G(w,t)\,.\end{aligned}$$

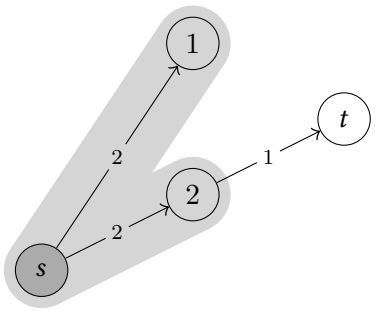
This eventually finishes this case because $\operatorname{dist}^{\wedge}_G(s,w) = \operatorname{dist}_G(s,w)$ is satisfied by (2.15). □

By Theorem 2 one may compute distances $\operatorname{dist}_G(s,t)$ in G by computing the appropriate distances $\operatorname{dist}^{\wedge}_G(s,u)$ and $\operatorname{dist}^{\vee}_G(u,t)$ in a contraction hierarchy G_α of G. With a little care, it is even possible to extract a shortest path between s and t in G from this computation in G_α in time proportional to the number of arcs on this shortest path. We do not consider this problem any further but point the interested reader to [Gei07], where all the details are carried out. Let us focus on the computation of distances in G_α instead.

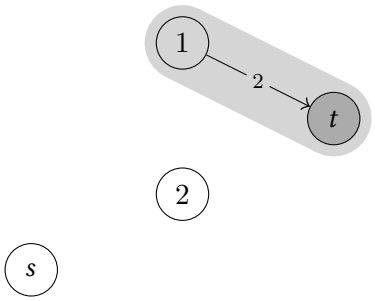
Algorithm 2.3: Naive shortest path query in contraction hierarchies

Data: Contraction hierarchy $G_\alpha = (G^{\wedge}_\alpha, G^{\vee}_\alpha)$ of some weighted graph $G = (V,E)$
Input: Vertices $s, t \in V$
Output: Distance $\operatorname{dist}_G(s,t)$
/* *Run one instance of Dijkstra's algorithm with source s in $G^{\wedge}_\alpha$ and one instance with source t in $G^{\vee}_\alpha$ with all arcs reversed. Denote the tentative distances by $d_s(-)$ and $d_t(-)$.* */

1 Run Dijkstra's algorithm with source s in $G^{\wedge}_\alpha$
2 Run Dijkstra's algorithm with source t in $G^{\vee}_\alpha$ with reversed arcs
3 **return** $\min_{u \in V} d_s(u) + d_t(u)$



(a) The graph $G_\alpha^\wedge$ and the state of the query starting from s. Observe that $\min_{x \in Q_s} d_s(x) = 2$.



(b) The graph $G_\alpha^\vee$ and the state of the query starting from t. Observe that $\min_{x \in Q_t} d_t(x) = 2$.

Figure 2.7: An example of a contraction hierarchy G_α, where the halting criterion of bidirectional Dijkstra's algorithm does not work. The correct distance $\operatorname{dist}_G(s,t)$ is not known, even though $\min_{u \in V} d_s(u) + d_t(u) = d_s(1) + d_t(1) = 4$ and $M_s + M_t = 4$. Settled vertices are drawn in grey, while the relaxed arcs and the discovered vertices are drawn on a background shaded in lighter grey.

Consider Algorithm 2.3. By the correctness of Dijsktra's algorithm, it follows that $d_s(u) = \operatorname{dist}_G^\wedge(s,u)$ and $d_t(u) = \operatorname{dist}_G^\vee(u,t)$ when the algorithm terminates in line 3. According to Theorem 2, Algorithm 2.3 is thus correct, i.e. it returns $\operatorname{dist}_G(s,t)$. Note that Algorithm 2.3 closely resembles the bidirectional query Algorithm 1.2 in the sense that there are two concurrent instances of Dijkstra's algorithm – one starting from s and one starting from t in the reverse direction. As with bidirectional Dijkstra's algorithm in general graphs, there is a point in time at which these two concurrent queries meet and the value of $\operatorname{dist}_G(s,t)$ is already known but Algorithm 2.3 continues nonetheless. Recall from № 1.4 that bidirectional Dijkstra's algorithm may terminate as soon as

$$\min_{u \in V} d_s(u) + d_t(u) \leq M_s + M_t \,, \tag{2.17}$$

where $d_s(-)$ and $d_t(-)$ denote the respective tentative distances and M_s and M_t denote the minimum tentative distance among all vertices contained in the queues Q_s and Q_t of the queries starting from s and from t, respectively. Similarly, we may improve on Algorithm 2.3 by introducing an appropriate criterion for when to abort the shortest path computations in $G_\alpha^\wedge$ and $G_\alpha^\vee$. However, it is not possible to employ the criterion (2.17) in Algorithm 2.3 without further modification. This stems from the fact that in plain graphs all the arcs and vertices on a shortest s-t-path are reachable from both s and t, while in contraction hierarchies there may be arcs or vertices reachable only from s or only from t. This problem may also be seen in Figure 2.7, where the query in $G_\alpha^\vee$ did not relax the arc $2 \to t$, which would have been taken into consideration in a bidirectional query on a plain graph. As a consequence, (2.17) is satisfied but $\operatorname{dist}_G(s,t)$ is still not known in this example. One solution to this problem

is not to terminate until

$$\min_{u \in V} d_s(u) + d_t(u) \leq \min\{M_s, M_t\}\,. \tag{2.18}$$

This criterion is actually the one that we make use of in Algorithm 2.4, our improved variant of Algorithm 2.3. Even though this criterion might seem rather weak on first sight, it appears to be difficult to devise a stronger but still correct criterion. To see this, consider our example from Figure 2.7 again. Therein, as well as in many other examples, the distance of s and t in G is completely determined by only the distance of s and t in $G_\alpha^\wedge$. That is, the algorithm must not abort while M_s is still greater than the currently smallest tentative distance $d_s(u) + d_t(u)$ from s to u.

Algorithm 2.4: Improved shortest path query in contraction hierarchies

Data: Contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$
Input: Source vertex $s \in V$, target vertex $t \in V$
Output: Distance $\operatorname{dist}_G(s,t)$

/* *Run one instance of Dijkstra's algorithm with source s in $G_\alpha^\wedge$ and one instance with source t in $G_\alpha^\vee$ with all arcs reversed. Denote the queues by Q_s and Q_t and the tentative distances by $d_s(-)$ and $d_t(-)$, respectively. Further let $M_s = \min_{x \in Q_s} d_s(x)$ and $M_t = \min_{x \in Q_t} d_t(x)$.* */

1 **while** $\min_{u \in V} d_s(u) + d_t(u) > \min\{M_s, M_t\}$ **do**
2 Choose one instance x of Dijkstra's algorithm, such that $Q_x \neq \varnothing$
3 Perform one single iteration in the chosen instance
4 **return** $\min_{u \in D} d_s(u) + d_t(u)$

2.7 Proposition: Bidirectional Dijkstra's algorithm on contraction hierarchies with halting criterion (2.18) is correct. †

Proof: We make use of the notation in Algorithm 2.4. Suppose that there is some vertex $u \in V$, such that

$$d_s(u) + d_t(u) \leq \min\{M_s, M_t\}\,. \tag{2.19}$$

Recall from № 1.1, that $d_s(u) \geq \operatorname{dist}_G^\wedge(s,u)$ and $d_t(u) \geq \operatorname{dist}_G^\vee(u,t)$. Employing these inequalities in (2.19), one computes

$$M_s \geq d_s(u) + d_t(u) \geq \operatorname{dist}_G^\wedge(s,u) + \operatorname{dist}_G^\vee(u,t) \geq \operatorname{dist}_G(s,t) \tag{2.20}$$

$$\text{and } M_t \geq d_s(u) + d_t(u) \geq \operatorname{dist}_G^\wedge(s,u) + \operatorname{dist}_G^\vee(u,t) \geq \operatorname{dist}_G(s,t)\,. \tag{2.21}$$

Now consider some vertex $v \in V$ such that $\operatorname{dist}_G(s,t) = \operatorname{dist}_G^\wedge(s,v) + \operatorname{dist}_G^\vee(v,t)$. Note that there is at least one such vertex by Theorem 2. Our computations (2.20)

and (2.21) from above show that $M_s \geq \operatorname{dist}^{\wedge}_G(s,v)$ and $M_t \geq \operatorname{dist}^{\vee}_G(v,t)$. As the minimum tentative distances M_s and M_t among all the vertices in the respective queue Q_s or Q_t are both monotonically increasing, each vertex in one of the queues Q_s or Q_t may from this point on only be updated with tentative distances strictly greater than $\operatorname{dist}^{\wedge}_G(s,v)$ or $\operatorname{dist}^{\vee}_G(v,t)$. Since Dijkstra's algorithm is still correct when considered on $G^{\wedge}_{\alpha}$ or $G^{\vee}_{\alpha}$ only, it follows that $d_s(v) = \operatorname{dist}^{\wedge}_G(s,v)$ and $d_t(v) = \operatorname{dist}^{\vee}_G(v,t)$. Hence

$$\min_{u \in V} d_s(u) + d_t(u) \leq \operatorname{dist}^{\wedge}_G(s,v) + \operatorname{dist}^{\vee}_G(v,t) = \operatorname{dist}_G(s,t)\,.$$

This finishes the proof, because $\min_{u \in V} d_s(u) + d_t(u) \geq \operatorname{dist}_G(s,t)$ is always satisfied. □

The reader should note that the strategy with which an instance of Dijkstra's algorithm is chosen in line 2 of Algorithm 2.4 may have considerable impact on its performance. As it was the case with bidirectional Dijkstra's algorithm, there is a simple strategy that guarantees a running time at most twice the optimum. Both the strategy and the proof of its performance are precisely the same as in № 1.5.

Special halting criteria like the one in Algorithm 2.4 are an opportunity to increase performance which one is rarely willing to neglect in practise. In theory, however, these special halting criteria introduce major technical difficulties while simpler algorithms are still amenable to concise and clear analysis. For this very reason, we chose Algorithm 2.3 as starting point for our studies of contraction hierarchies. However, we are mainly concerned with upper bounds on the performance of Algorithm 2.3 and these upper bounds naturally hold for Algorithm 2.4, too. Moreover, there is experimental evidence in [Gei07] that at least on road networks Algorithm 2.3 settles only a small multiple of the number of vertices settled by Algorithm 2.4. For these reasons, we end our detour to special halting criteria and focus exclusively on the simple query Algorithm 2.3. As there are then no special halting criteria, the instance of Dijkstra's algorithm with source s eventually settles all the vertices $u \in V$ with $\operatorname{dist}^{\wedge}_G(s,u) < \infty$. Likewise, all the vertices $u \in V$ with $\operatorname{dist}^{\vee}_G(u,t) < \infty$ are settled by the query starting with t. The sets

$$\mathcal{S}(s, G^{\wedge}_{\alpha}) = \left\{ u \in V \;\middle|\; \operatorname{dist}^{\wedge}_G(s,u) < \infty \right\}$$

and

$$\mathcal{R}(t, G^{\vee}_{\alpha}) = \left\{ u \in V \;\middle|\; \operatorname{dist}^{\vee}_G(u,t) < \infty \right\}$$

are thus precisely the sets of vertices settled during these two queries. More generally, we define

$$\begin{aligned} \mathcal{S}(v,G) &= \left\{ u \in V \;\middle|\; \operatorname{dist}_G(v,u) < \infty \right\} \\ \mathcal{R}(v,G) &= \left\{ u \in V \;\middle|\; \operatorname{dist}_G(u,v) < \infty \right\} \end{aligned} \tag{2.22}$$

for any directed, acyclic graph G and any $v \in V$. We call $\mathcal{S}(v,G)$ the *search space* and $\mathcal{R}(v,G)$ the *reverse search space* of v in G. The cardinalities of these sets $\mathcal{S}(v,G)$

and $\mathcal{R}(v, G)$ are the *search space size* and *reverse search space size* of v, respectively. We denote these values by $\#\mathcal{S}(v, G)$ and $\#\mathcal{R}(v, G)$. An example of a contraction hierarchy G_α together with specific search spaces $\mathcal{S}(s, G_\alpha^\wedge)$ and $\mathcal{R}(t, G_\alpha^\vee)$ in G_α is depicted in Figure 2.8.

Despite the fact that the runtime of Dijkstra's algorithm depends not only on the number of settled vertices but also on the number of relaxed arcs, the former value is considered as a rough measure for the performance of a specific query algorithm by Bauer et al. [BDDW09; BCKKW10; BBRW12]. We also comply with this model but denote the actual running time of Algorithm 2.3 or Algorithm 2.4 by $\mathsf{T}_{\text{query}}$ to distinguish it from the search space size. Finding the contraction hierarchy G_α on which Algorithm 2.3 is fastest in our chosen model is equivalent to finding an order α that minimises

$$\max_{s,t \in V} \#\mathcal{S}(s, G_\alpha^\wedge) + \#\mathcal{R}(t, G_\alpha^\vee)\,.$$

Having chosen Algorithm 2.3 as starting point for our studies, we see that $\#\mathcal{S}(s, G_\alpha^\wedge)$ and $\#\mathcal{R}(t, G_\alpha^\vee)$ are actually independent of each other, so that we might equally well ask for an order α, such that

$$\max_{s \in V} \#\mathcal{S}(s, G_\alpha^\wedge) + \max_{t \in V} \#\mathcal{R}(t, G_\alpha^\vee) \tag{2.23}$$

is minimal. We additionally simplify the problem and only ask for an order α that minimises the *maximum search space size* $\mathcal{S}_{\max}(G_\alpha)$ given by

$$\mathcal{S}_{\max}(G_\alpha) = \max\left\{ \#\mathcal{S}(v, G_\alpha^\wedge), \#\mathcal{R}(v, G_\alpha^\vee) \;\middle|\; v \in V \right\}.$$

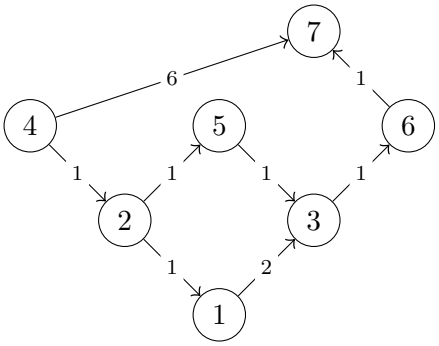
We justify this last simplification by the fact that $\mathcal{S}_{\max}(G_\alpha)$ differs from (2.23) by at most a factor of two. For an arbitrary digraph G, we also write $\mathcal{S}_{\max}(G)$ and $\mathcal{R}_{\max}(G)$ to denote the values $\max_{v \in V} \#\mathcal{S}(v, G)$ and $\max_{v \in V} \#\mathcal{R}(v, G)$, respectively. With this notation, our above definition of $\mathcal{S}_{\max}(G_\alpha)$ immediately implies that $\mathcal{S}_{\max}(G_\alpha)$ is equal to $\max\{\mathcal{S}_{\max}(G_\alpha^\wedge), \mathcal{R}_{\max}(G_\alpha^\vee)\}$. We further denote by $\mathcal{S}_{\max}(G)$ the *minimum maximum search space size*, that is the value $\min_\alpha \mathcal{S}_{\max}(G_\alpha)$.

We further define the *unidirectional average search space size* $\#\mathcal{S}(G)$ of a directed acyclic graph G as

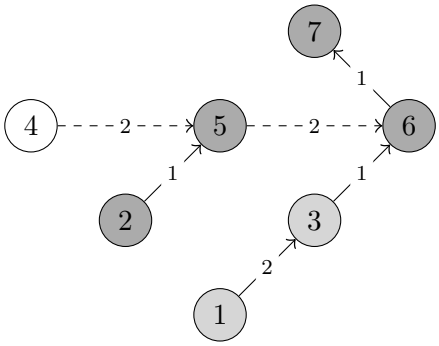
$$\#\mathcal{S}(G) = \frac{1}{n} \cdot \sum_{v \in V} \#\mathcal{S}(v, G)\,. \tag{2.24}$$

Note that $u \in \mathcal{R}(v, G)$ if and only if $v \in \mathcal{S}(u, G)$ by the definition (2.22) of $\mathcal{S}(-, G)$ and $\mathcal{R}(-, G)$. Consequently, the two maps $\delta_{\mathcal{S}}$ and $\delta_{\mathcal{R}}$ given by

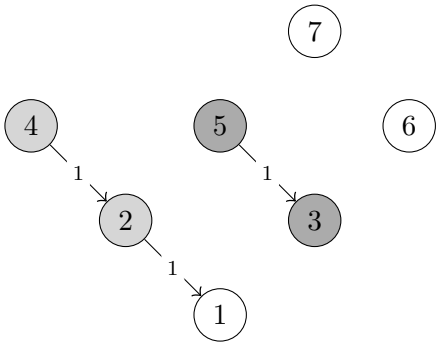
$$\delta_{\mathcal{S}}(u, v) = \begin{cases} 1 & \text{if } v \in \mathcal{S}(u, G) \\ 0 & \text{otherwise} \end{cases} \qquad \delta_{\mathcal{R}}(u, v) = \begin{cases} 1 & \text{if } u \in \mathcal{R}(v, G) \\ 0 & \text{otherwise} \end{cases}$$



(a) A weighted graph $G = (V, A)$ on seven vertices. The values $\alpha(v)$ for an order $\alpha : V \to \{1, \dots, 7\}$ are drawn inside the respective nodes $v \in V$.



(b) The upwards directed graph $G_\alpha^\wedge$ associated with the digraph G and the order α shown on the left. All vertices in $\mathcal{S}(2, G_\alpha^\wedge)$ are shaded in darker grey, while all vertices in $\mathcal{R}(3, G_\alpha^\wedge)$ are shaded in lighter grey.



(c) The downwards directed graph $G_\alpha^\vee$ associated with the digraph G and the order α shown in Figure 2.8a. All vertices in $\mathcal{S}(5, G_\alpha^\vee)$ are shaded in darker grey, while all vertices in $\mathcal{R}(2, G_\alpha^\vee)$ are shaded in lighter grey.

Figure 2.8: Figure 2.8a shows a weighted digraph $G = (V, A)$ on seven vertices and an order $\alpha : V \to \{1, \dots, 7\}$ on V. Figure 2.8b and Figure 2.8c depict the upwards and downwards directed graphs $G_\alpha^\wedge$ and $G_\alpha^\vee$ associated with the graph G and the given order on its vertices. Note in particular that the arc $4 \to 7$ of G is not present in G_α anymore. Moreover, one may see the search space and reverse search space of single vertices in Figure 2.8b and Figure 2.8c.
Arc lengths $\text{len}(e)$ are always drawn on the respective arcs and shortcuts of the contraction hierarchy are drawn dashed.

are actually equal. Utilising the equality of these two maps, one computes

$$\begin{aligned}\#\mathcal{S}(G) = \frac{1}{n} \cdot \sum_{u \in V} \#\mathcal{S}(u, G) &= \frac{1}{n} \cdot \sum_{u \in V} \sum_{v \in V} \delta_{\mathcal{S}}(u, v) \\ &= \frac{1}{n} \cdot \sum_{v \in V} \sum_{u \in V} \delta_{\mathcal{R}}(u, v) = \frac{1}{n} \cdot \sum_{v \in V} \#\mathcal{R}(v, G)\,.\end{aligned} \tag{2.25}$$

Hence, a definition of unidirectional average reverse search space size in the spirit of (2.24) is superfluous, for it would coincide with unidirectional average search space size anyway.

We close this paragraph with a brief computation showing that $\#\mathcal{S}(G_\alpha^\wedge) + \#\mathcal{S}(G_\alpha^\vee)$ is indeed nothing but the average number of vertices settled by Algorithm 2.3. To this end, observe that the average number of settled vertices is given by

$$\frac{1}{n^2} \sum_{(s,t) \in V \times V} \#\mathcal{S}(s, G_\alpha^\wedge) + \#\mathcal{R}(t, G_\alpha^\vee)\,,$$

which may be recast as

$$\begin{aligned}\frac{1}{n^2} \sum_{(s,t) \in V \times V} \#\mathcal{S}(s, G_\alpha^\wedge) + \#\mathcal{R}(t, G_\alpha^\vee) &= \frac{1}{n^2} \left(n \sum_{s \in V} \#\mathcal{S}(s, G_\alpha^\wedge) + n \sum_{t \in V} \#\mathcal{R}(t, G_\alpha^\vee) \right) \\ &= \frac{1}{n} \sum_{s \in V} \#\mathcal{S}(s, G_\alpha^\wedge) + \frac{1}{n} \sum_{t \in V} \#\mathcal{R}(t, G_\alpha^\vee) \\ &= \#\mathcal{S}(G_\alpha^\wedge) + \#\mathcal{S}(G_\alpha^\vee)\,,\end{aligned}$$

where the last equality follows from (2.25). For a given weighted digraph G and an order α on its vertices, we therefore define the *average search space size* $\mathcal{S}_{\text{avg}}(G_\alpha)$ as the sum $\mathcal{S}_{\text{avg}}(G_\alpha) = \#\mathcal{S}(G_\alpha^\wedge) + \#\mathcal{S}(G_\alpha^\vee)$. Just as we consider $\mathcal{S}_{\max}(G_\alpha)$ to be a measure for the worst-case performance, we will treat $\mathcal{S}_{\text{avg}}(G_\alpha)$ as a measure for the average-case performance of Algorithm 2.3 when run with input G_α.

2.4 Weak Contraction Hierarchies

In this paragraph, we recapitulate our different models of contraction hierarchies one last time and devise a definition that captures not only G_α and $\bar{G}_\alpha$ but also many more graphs akin to contraction hierarchies. The following is therefore both a summary of Chapter 2 and a supplement to its contents.

Motivated by the problem of removing vertices from a given graph while preserving the distances between all remaining vertices, we defined and studied contraction of single vertices in Chapter 2.1. On the basis of iterated contraction, we then devised an algorithm that computes a contraction hierarchy $\bar{G}_\alpha = (\bar{G}_\alpha^\wedge, \bar{G}_\alpha^\vee)$ from a

given weighted digraph G and an order α on its vertices. We asserted that the contraction hierarchy $\bar{G}_\alpha$ essentially contains all the information about shortest paths and distances in G. Our subsequent analysis of $\bar{G}_\alpha$ ultimately led to Theorem 1 in which we gave a concise description of $\bar{G}_\alpha$ by means of only the order α and the shortest paths in G. This characterisation of $\bar{G}_\alpha$ also revealed that it actually contains arcs that do not account for shortest paths or distances at all. Removing those arcs led us to define a second notion of contraction hierarchy, which we denoted by G_α. In № 2.3 and № 2.4 we argued that our two notions $\bar{G}_\alpha$ and G_α of contraction hierarchies still have some essential properties in common. More precisely, we established that a shortcut uw in both $\bar{G}_\alpha$ and G_α has supporting arcs uv and vw, such that $\mathrm{len}^\alpha_G(uw) = \mathrm{len}^\alpha_G(uv) + \mathrm{len}^\alpha_G(vw)$. The existence of these supporting arcs is at the heart of contraction as we introduced it in Chapter 2.1 and is yet a feature of our abstract definition, too. Then again, the abstract contraction hierarchies G_α contain precisely the information that is necessary to reconstruct distances in G from distances in G_α as we have seen in Theorem 2. Finding such a pair of graphs was the very motivation to define $\bar{G}_\alpha$ in the first place. It therefore appears to us that the presence of all the arcs of G_α and a well-defined notion of supporting arcs are the two most important characteristics shared by both $\bar{G}_\alpha$ and G_α. We take these as the defining properties of a *weak contraction hierarchy*, that is, a weak contraction hierarchy H_α of a given weighted digraph $G = (V, A)$ is a pair $H_\alpha = (H^\wedge_\alpha, H^\vee_\alpha)$ of digraphs $H^\wedge_\alpha = (V, B^\wedge_\alpha)$ and $H^\vee_\alpha = (V, B^\vee_\alpha)$, such that the following conditions (w1), (w2) and (w3) are met.

(w1) $G_\alpha \subseteq H_\alpha$

(w2) $\alpha(u) < \alpha(v)$ for each $uv \in B^\wedge_\alpha$ and each $vu \in B^\vee_\alpha$

(w3) If uw is an arc of H_α that is not contained in G, then there is at least one pair of arcs $uv \in B^\vee_\alpha$ and $vw \in B^\wedge_\alpha$.

In the remainder of this paragraph, we indicate how to extend our previous findings for contraction hierarchies to weak contraction hierarchies and investigate the relationship between different weak contraction hierarchies – an important issue that becomes apparent only in this generality. For this purpose we keep a weighted digraph $G = (V, A)$ and an order α on its vertices fixed. As usual, we denote by $G_\alpha = (G^\wedge_\alpha, G^\vee_\alpha)$ the contraction hierarchy of G with respect to α. In addition to this data, we also fix a weak contraction hierarchy $H_\alpha = (H^\wedge_\alpha, H^\vee_\alpha)$ whose arcs will be denoted by $B^\wedge_\alpha$ and $B^\vee_\alpha$ as above.

Recall from Chapter 2.1 and Chapter 2.2 that we insisted on calling certain arcs of $\bar{G}_\alpha$ or G_α shortcuts even though they were contained in the graph G underlying the contraction hierarchy in question. More precisely, an arc uw of G is called a shortcut in $\bar{G}_\alpha$ or G_α if there is some vertex v whose contraction during the computation of $\bar{G}_\alpha$ would cause the length of uw to be overwritten, i.e. if $\mathrm{len}_G(uw) > \mathrm{dist}_G(u, w)$ and additionally $P_\alpha(u, w) = \{u, w\}$. We adopt this definition as it stands for weak

contraction hierarchies. That is, an arc uw of H_α is a *shortcut*, if it is either not contained in G or if $P_\alpha(u,w) = \{u,w\}$ and $\mathrm{len}_G(uw) > \mathrm{dist}_G(u,w)$. When it is requisite to distinguish shortcuts in G_α from those in H_α, we call the latter *weak shortcuts*.

2.8 Remark: There are arcs $uv \in B_\alpha^\vee$ and $vw \in B_\alpha^\wedge$ for each shortcut uw of a weak contraction hierarchy H_α. †

Proof: If uw is no arc of G, condition (w3) immediately implies the existence of arcs $uv \in B_\alpha^\wedge$ and $vw \in B_\alpha^\vee$. If, on the other hand, uw is contained in G, our definition of shortcuts in H_α then implies that uw is a shortcut of G_α for which we already reassured the existence of supporting arcs in № 2.3. □

Although distances and arc lengths are only secondary in the remaining chapters, we still want to point out that all the results concerning distances in G and distances in G_α that we established in Chapter 2.3 hold for weak contraction hierarchies H_α, too. To this end, consider the arc lengths $\mathrm{len}_H^\alpha(uv)$ on H_α given by

$$\mathrm{len}_H^\alpha(uw) = \begin{cases} \min\limits_{\substack{uv \in B_\alpha^\vee \\ vw \in B_\alpha^\wedge}} \mathrm{len}_H^\alpha(uv) + \mathrm{len}_H^\alpha(vw) & uw \text{ is a shortcut} \\ \mathrm{len}_G(uw) & \text{otherwise.} \end{cases} \tag{2.26}$$

Observe that $\mathrm{len}_H^\alpha(-)$ is well-defined by virtue of № 2.8. Just as with distances in contraction hierarchies, we write $\mathrm{dist}_H^\wedge(u,v)$ and $\mathrm{dist}_H^\vee(u,v)$ to denote the distance of two vertices u and v in $H_\alpha^\wedge$ and $H_\alpha^\vee$, respectively. Observe that the above definition of arc lengths $\mathrm{len}_H^\alpha(-)$ in H_α coincides with the definition of arc lengths $\mathrm{len}_G^\alpha(-)$ in $\bar{G}_\alpha$ when considered as a weak contraction hierarchy. To see this, recall that the length of any arc uw of $\bar{G}_\alpha$ is given by the minimum of the lengths $\mathrm{len}_{G(i)}(uw)$ of uw in the graphs $G(i)$. This obviously equals $\mathrm{len}_G(uw)$ in the case that uw is no shortcut and is nothing but the minimum of the lengths of all the paths (u,v,w) encountered during the contraction of some vertex v in the case that uw is a shortcut. However, any such path (u,v,w) corresponds to a pair $uv \in A_\alpha^\vee$ and $vw \in A_\alpha^\wedge$ of arcs and vice versa. All in all, the length $\mathrm{len}_G^\alpha(uw)$ of a shortcut uw in $\bar{G}_\alpha$ is given by the minimum of $\mathrm{len}_G^\alpha(uv) + \mathrm{len}_G^\alpha(vw)$ over all pairs $uv \in A_\alpha^\vee$ and $vw \in A_\alpha^\wedge$ of arcs – the definition of the length of uw in H_α. Note that this also implies that $\mathrm{len}_H^\alpha(-)$ coincides with the arc lengths $\mathrm{len}_G^\alpha(-)$ in G_α considered as a weak contraction hierarchy, for the arc lengths in G_α are precisely those in $\bar{G}_\alpha$ by Theorem 1 and our very definition of G_α.

Finally, we show how to transfer the notions of supporting arcs and shortcut depth from contraction hierarchies to weak contraction hierarchies. For this purpose consider a shortcut uw of a weak contraction hierarchy H_α. Mimicking the construction (2.5) of the supporting vertex in G_α, we consider the set

$$S = \left\{ (uv, vw) \;\middle|\; uv \in B_\alpha^\vee, vw \in B_\alpha^\wedge \text{ and } \mathrm{len}_H^\alpha(uw) = \mathrm{len}_H^\alpha(uv) + \mathrm{len}_H^\alpha(vw) \right\}.$$

This set is nonempty by № 2.8 and our definition (2.26) of $\mathrm{len}_H^\alpha(-)$. The *supporting arcs* $\sup(uw) = (uv, vw)$ of uw are now defined to be that pair $(uv, vw) \in S$ with maximal $\alpha(v)$. The *supporting vertex* of uw is simply the vertex v occurring as target of the first and source of the second supporting arc of uw. Now, our previous definition (2.9) of the shortcut depth $\mathrm{scd}(uw)$ of an arc uw of G_α carries over to weak contraction hierarchies without any modifications. Spelled out, the *shortcut depth* $\mathrm{scd}(uw)$ of an arc of H_α is given by

$$\mathrm{scd}(uw) = \begin{cases} \mathrm{scd}(uv) + \mathrm{scd}(vw) & uw \text{ is a shortcut, } \sup(uw) = (uv, vw) \\ 1 & \text{otherwise.} \end{cases} \tag{2.27}$$

As it was the case with the shortcut depth of the arcs of G_α in Chapter 2.2, it is easy to convince oneself that $\mathrm{scd}(uw)$ is well-defined and that one is allowed to do induction on $\mathrm{scd}(-)$, i.e. that the shortcut depths of the supporting arcs of some shortcut uw are strictly less than $\mathrm{scd}(uw)$. Further observe that this notion of shortcut and shortcut depth coincides with our previous notions of shortcuts in $\bar{G}_\alpha$ and G_α and shortcut depth $\mathrm{scd}(-)$ in G_α when considering those contraction hierarchies as weak contraction hierarchies.

The notions of shortcut and supporting arcs in weak contraction hierarchies encompass the corresponding notions in $\bar{G}_\alpha$ and G_α. In addition, we argued above that the arc lengths $\mathrm{len}_H^\alpha(-)$ defined in (2.26) also coincide with the lengths $\mathrm{len}_G^\alpha(-)$ in $\bar{G}_\alpha$ and G_α. Altogether, weak contraction hierarchies in fact embrace both our previous definitions of contraction hierarchies. Moreover, carefully revisiting the proof of Theorem 2, we realise that it incorporates only very few properties of G_α and its arc lengths. To be precise, we utilised the fact that $\mathrm{len}_G^\alpha(uv) \geq \mathrm{dist}_G(u, v)$ in № 2.6 to be able to infer equality of $\mathrm{dist}_G^\wedge(s, u) + \mathrm{dist}_G^\vee(u, t)$ and $\mathrm{dist}_G(s, t)$ from just the inequality $\mathrm{dist}_G^\wedge(s, u) + \mathrm{dist}_G^\vee(u, t) \leq \mathrm{dist}_G(s, t)$. Furthermore, the base case of the induction used to settle Theorem 2 crucially depended on the equality of $\mathrm{len}_G^\alpha(uv)$ and $\mathrm{dist}_G(uv)$ for all arcs uv with $P_\alpha(u, v) = \{u, v\}$. Apart from these two properties, our proof made use only of the induction hypothesis and properties of the sets $P_\alpha(-, -)$, which do not depend on G_α, $\bar{G}_\alpha$ or H_α though. As demonstrated by the following remark, the arc lengths of H_α as defined above possess both these properties and Theorem 2 therefore holds for weak contraction hierarchies, too. Furthermore, both Algorithm 2.3 and Algorithm 2.4 remain correct when run with a weak contraction hierarchy as input.

2.9 Remark: Let H_α be a weak contraction hierarchy.

(a) $\mathrm{len}_H^\alpha(uw) \geq \mathrm{dist}_G(u, w)$ for all arcs uw of H_α.

(b) $\mathrm{len}_H^\alpha(uw) = \mathrm{dist}_G(u, w)$ for all arcs uw of H_α with $P_\alpha(u, w) = \{u, w\}$. †

Proof: We do induction on the shortcut depth $\mathrm{scd}(uw)$ of uw.

$\mathrm{scd}(uw) = 1$: Observe that $\mathrm{scd}(uw) = 1$ implies that uw is an arc of G and that additionally $\mathrm{len}_G(uw) = \mathrm{dist}_G(u, w)$. Given our definition (2.26) of $\mathrm{len}_H^\alpha(-)$, both conditions (a) and (b) are then trivially satisfied.

$\mathrm{scd}(uw) > 1$: In this case, uw is a shortcut of H_α and its length therefore equals the sum of the lengths of its supporting arcs $\mathrm{sup}(uw) = (uv, vw)$. The induction hypothesis then implies $\mathrm{len}_H^\alpha(uv) \geq \mathrm{dist}_G(u, v)$ and $\mathrm{len}_H^\alpha(vw) \geq \mathrm{dist}_G(v, w)$, which already gives $\mathrm{len}_H^\alpha(uw) \geq \mathrm{dist}_G(u, w)$ by the triangle inequality.

Let us now suppose $P_\alpha(u, w) = \{u, w\}$. In this case uw is an arc of G_α and № 2.3 and № 2.4 hence imply that G_α contains supporting arcs $\mathrm{sup}(uw) = (uv, vw)$ satisfying $\mathrm{len}_G^\alpha(uw) = \mathrm{len}_G^\alpha(uv) + \mathrm{len}_G^\alpha(vw)$. Since both uv and vw are arcs of G_α, it follows from the induction hypothesis that $\mathrm{len}_H^\alpha(uv) = \mathrm{dist}_G(u, v)$ and $\mathrm{len}_H^\alpha(vw) = \mathrm{dist}_G(v, w)$. Employing these equalities in the definition of $\mathrm{len}_H^\alpha(-)$ finally gives $\mathrm{len}_H^\alpha(uw) = \mathrm{dist}_G(u, w)$ since the arc lengths in G_α are by definition nothing but the respective distances in G. □

Although a bit terse, the above discussion should have convinced the reader that weak contraction hierarchies are the "right" definition to embrace both G_α and $\bar{G}_\alpha$ while retaining the most desirable properties of both. As a side note, we remark that Algorithm 2.2, which facilitated our definition of $\bar{G}_\alpha$, is not quite the algorithm given by Geisberger et al. in [Gei07; GSSD08]. Actually, they considered it too costly to check if a given path (u, v, w) is a unique shortest path upon contraction of the vertex v in an actual implementation and therefore devised an algorithm that inserts more shortcuts than necessary. In fact, the algorithms in [Gei07; GSSD08] compute weak contraction hierarchies.

In view of condition (w1) it is clear that G_α is the unique smallest weak contraction hierarchy. Utilising shortest path queries in H_α, it is even possible to efficiently compute G_α from any given weak contraction hierarchy H_α. To this end, consider the following proposition, which characterises the arcs of G_α as a subset of the arcs of H_α in terms of a criterion that is suitable for an actual algorithm.

2.10 Proposition: Let H_α be a weak contraction hierarchy and let u and v be two vertices of H_α, such that $\alpha(u) < \alpha(v)$. The conditions

(i) $P_\alpha(u, v) = \{u, v\}$

(ii) (u, v) is the only shortest u-v-path in $H_\alpha^\wedge$ and v is additionally the only vertex in $P_\alpha(u, v)$ that satisfies $\mathrm{dist}_G(u, v) = \mathrm{dist}_H^\wedge(u, x) + \mathrm{dist}_H^\vee(x, v)$

as well as

(i′) $P_\alpha(v, u) = \{v, u\}$

(ii′) (v, u) is the only shortest v-u-path in $H_\alpha^\vee$ and v is additionally the only vertex in $P_\alpha(v, u)$ that satisfies $\mathrm{dist}_G(v, u) = \mathrm{dist}_H^\wedge(v, x) + \mathrm{dist}_H^\vee(x, u)$

are then equivalent. †

Proof: Note that the equivalence of (i′) and (ii′) follows from that of (i) and (ii) by symmetry. We therefore prove the equivalence of (i) and (ii) only.

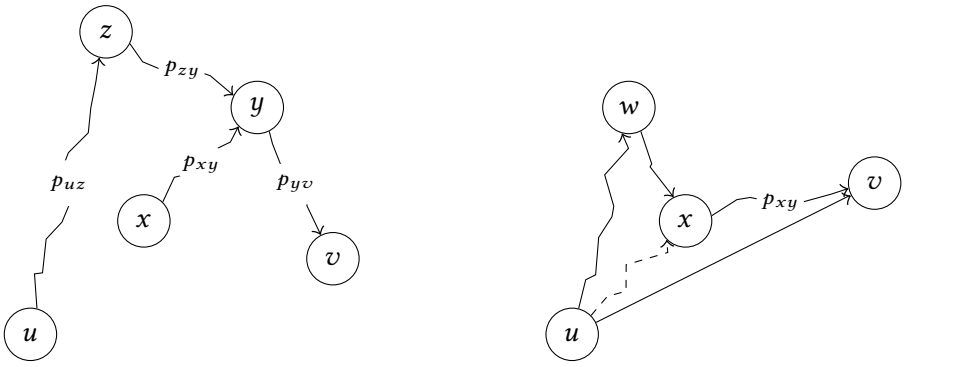
"(i) ⇒ (ii)": Note that $\alpha(x) \leq \alpha(v)$ for all $x \in P_\alpha(u, v)$. This implies in particular that there cannot possibly exist any vertex $x \in P_\alpha(u, v)$ that is distinct from v and still satisfies $\mathrm{dist}_G(u, v) = \mathrm{dist}^\wedge_H(u, x) + \mathrm{dist}^\vee_H(x, v)$, for any such vertex would have to satisfy $\alpha(x) > \alpha(v)$.

Now consider an arbitrary shortest path p from u to v in H_α. Since (u, v) is a shortest path by assumption and since $\mathrm{len}^\alpha_H(uv) = \mathrm{dist}_G(u, v)$ by № 2.9, it follows that $\mathrm{len}(p) = \mathrm{dist}_G(u, v)$. Let us assume for the sake of contradiction that p contains some vertex $x \notin \{u, v\}$. This vertex x would necessarily satisfy $\alpha(x) > \alpha(u)$ and $\mathrm{dist}^\wedge_H(u, x) + \mathrm{dist}^\wedge_H(x, v) = \mathrm{dist}_G(u, v)$. This latter equality can be satisfied only if $\mathrm{dist}_G(u, x) + \mathrm{dist}_G(x, v) = \mathrm{dist}_G(u, v)$ since $\mathrm{dist}^\wedge_H(u, x) \geq \mathrm{dist}_G(u, x)$ and $\mathrm{dist}^\vee_H(x, v) \geq \mathrm{dist}_G(x, v)$ by № 2.9. Note that $\mathrm{dist}_G(u, x) + \mathrm{dist}_G(x, v) = \mathrm{dist}_G(u, v)$ implies $x \in P_\alpha(u, v)$, which contradicts our assumptions. Altogether, we conclude that $p = (u, v)$, which was to be shown.

"'(ii) ⇒ (i)": This part of the proof is technically more involved than the first part and we therefore begin with a brief sketch of our argument. The proof consists of two steps, where each step utilises one of the two statements of condition (i). Assuming $P_\alpha(u, v) \neq \{u, v\}$, we choose $x \in P_\alpha(u, v) \setminus \{u, v\}$, such that $\alpha(x)$ is maximal. We then employ Theorem 2 twice to obtain vertices $y \in P_\alpha(x, v)$ and $z \in P_\alpha(u, y)$ together with shortest paths p_{xy} from x to y and p_{uz} from u to z in $H^\wedge_\alpha$ and p_{zy} from z to y and p_{yv} from y to v in $H^\vee_\alpha$. A sketch of these paths may also be seen in Figure 2.9a. By concatenating all these paths, we then deduce that the vertex z is actually one of the vertices $z \in P_\alpha(u, v)$, such that $\mathrm{dist}^\wedge_H(u, z) + \mathrm{dist}^\vee_H(z, v) = \mathrm{dist}_G(z, v)$, which gives $z = v = y$ by our assumptions.

As can also be seen in Figure 2.9b, we then find x somewhere between u and v. Since any vertex in $P_\alpha(u, x)$ is contained in $P_\alpha(u, v)$ and since x was chosen from $P_\alpha(u, v)$ such that $\alpha(x)$ is maximal, $w = x$ is the only vertex $w \in P_\alpha(u, x)$ together with paths from u to w and from w to x in $H^\wedge_\alpha$ and $H^\vee_\alpha$, respectively. However, such paths are guaranteed to exist by Theorem 2, which implies the existence of a path from u to x and hence a path from u via x to v. We are then able to prove this path to be a shortest path, which contradicts our second assumption and therefore finishes the proof.

Let us now fill in the technical details. Assume $P_\alpha(u, v) \neq \{u, v\}$ and choose the vertex $x \in P_\alpha(u, v) \setminus \{u, v\}$ with $\alpha(x)$ maximal. By Theorem 2, there is then



(a) Sketch of all the paths in H_α that occur during the first step of the proof.

(b) Sketch of the relevant parts of H_α during the second step of the proof.

Figure 2.9: Visualisation of the strategy used in the proof of № 2.10.

some vertex $y \in P_\alpha(x, v)$, such that

$$\begin{aligned} \text{dist}^\wedge_H(x, y) &= \text{dist}_G(x, y) \\ \text{and } \text{dist}^\vee_H(y, v) &= \text{dist}_G(y, v)\,. \end{aligned} \tag{2.28}$$

We denote the paths in $H^\wedge_\alpha$ and $H^\vee_\alpha$ that correspond to the above distances by p_{xy} and p_{yv}, respectively. Another application of Theorem 2 now guarantees the existence of some $z \in P_\alpha(u, y)$, such that

$$\text{dist}^\wedge_H(u, z) = \text{dist}_G(u, z) \tag{2.29}$$

$$\text{and } \text{dist}^\vee_H(z, y) = \text{dist}_G(z, y)\,. \tag{2.30}$$

Again, we denote by p_{uz} and p_{zy} the paths in H_α that correspond to these two equations. Utilising the equations (2.29), (2.30) and (2.28), we now compute

$$\begin{aligned} \text{dist}^\wedge_H(u, z) + \text{dist}^\vee_H(z, v) &\leq \text{dist}^\wedge_H(u, z) + \text{dist}^\vee_H(z, y) + \text{dist}^\vee_H(y, v) \\ &= \text{dist}_G(u, z) + \text{dist}_G(z, y) + \text{dist}_G(y, v) \\ &= \text{dist}_G(u, v) \end{aligned}$$

where the last equality follows from $z \in P_\alpha(u, y)$ and $y \in P_\alpha(x, v) \subseteq P_\alpha(u, v)$. Note that our assumptions now imply $z = v$. Moreover, since y lies on the path $p_{zy} \cdot p_{yv}$ from $z = v$ to v in $H^\vee_\alpha$, we may also conclude that $y = v$.

We now claim that $H^\wedge_\alpha$ contains a shortest path from u via x to v. To this end, note that Theorem 2 implies the existence of some $w \in P_\alpha(u, x) \subseteq P_\alpha(u, v)$ such that $\alpha(w) > \min\{\alpha(u), \alpha(x)\}$ and such that there is a path q from u to w in $H^\wedge_\alpha$ of length $\text{dist}_G(u, w)$ and a path r from w to x in $H^\vee_\alpha$ of length $\text{dist}_G(w, x)$. Observe that each such vertex w is an element of $P_\alpha(u, v)$ distinct from both u and v

and still satisfies $\alpha(w) \geq \alpha(x)$. Our initial choice of x therefore implies $w = x$. The concatenation $q \cdot p_{xy}$ of the path q from u to $w = x$ and the path p_{xy} from x to $y = v$ from above is then a path in $H_\alpha^\wedge$ of length

$$\begin{aligned} \operatorname{len}(q \cdot p_{xy}) &= \operatorname{dist}_H^\wedge(u, x) + \operatorname{dist}_H^\wedge(x, v) \\ &= \operatorname{dist}_G(u, x) + \operatorname{dist}_G(x, v) \\ &= \operatorname{dist}_G(u, v)\,, \end{aligned}$$

where the last equality follows from $x \in P_\alpha(u, v)$. As $\operatorname{dist}_H^\wedge(u, v) \geq \operatorname{dist}_G(u, v)$, we find that $q \cdot p_{xy}$ is a shortest path in $H_\alpha^\wedge$, which contradicts our assumption that (u, v) is the only shortest path from u to v in $H_\alpha^\wedge$. ▫

The above proposition facilitates an algorithm that determines whether a given arc st of a weak contraction hierarchy H_α is an arc of the contraction hierarchy G_α contained in H_α. To this end, it suffices to adapt the distance query Algorithm 2.3, which is correct for weak contraction hierarchies by our previous discussions, such that it does not only compute some vertex x with $\operatorname{dist}_G(s, t) = \operatorname{dist}_H^\wedge(s, x) + \operatorname{dist}_H^\vee(x, t)$ but also determines whether both this vertex and the accompanying shortest path from s to x are unique. According to № 2.10, st is an arc of G_α if and only if the above conditions are met and if furthermore $x = t$. Recall from Chapter 2.3 that Algorithm 2.3 performs two concurrent Dijkstra queries with respective source vertices s in $H_\alpha^\wedge$ and t in $H_\alpha^\vee$ with reversed arcs. Further recall that we denoted by $d_s(x)$ and $d_t(x)$ the tentative distances of x in the instance of Dijkstra's algorithm with source s and source t, respectively. If, during the course of this algorithm, one does not only maintain the length $\min d_s(x) + d_t(x)$ of the as yet shortest path from s to t but also a set M containing all the vertices x that minimise $d_s(x) + d_t(x)$, then it is certainly possible to decide whether the vertex x with $\operatorname{dist}_G(s, t) = \operatorname{dist}_G^\wedge(s, x) + \operatorname{dist}_G^\vee(x, t)$ is unique and equal to v. Utilising for example linked lists, this is obviously possible with only a constant overhead per arc relaxation.

The only remaining issue is to determine whether a shortest path from u to x in $H_\alpha^\wedge$ is unique. For this purpose, one maintains a boolean flag $\mathsf{unique}(x)$ for each vertex $x \in V$. Initially, this flag is set to **True**. Upon relaxation of an arc xy in the instance of Dijkstra's algorithm with source s, one updates the flag $\mathsf{unique}(y)$ by

$$\mathsf{unique}(y) \leftarrow \begin{cases} \textbf{False} & \text{if } d_s(x) + \operatorname{len}_G(xy) = d_s(y) \\ \mathsf{unique}(x) & \text{if } d_s(x) + \operatorname{len}_G(xy) < d_s(y)\,. \end{cases}$$

Stated less formally, $\mathsf{unique}(y)$ is set to **False** whenever Dijkstra's algorithm discovers a path from s to y that is of the same length as some previously discovered path and it is set to $\mathsf{unique}(x)$ when the algorithm discovers a path – or possibly many? – from s via x to y of length strictly less than all previously discovered such paths. Since all the flags $\mathsf{unique}(y)$ are **True** initially, an easy induction on the number of iterations

proves unique(y) to be **True** when y gets settled if and only if there is a unique shortest path from s to y in $H_\alpha^\wedge$. That is, the algorithm is correct indeed. Moreover, the situation is completely symmetric in $H_\alpha^\vee$ and the flags unique$(-)$ can obviously be implemented with constant overhead per arc relaxation. Altogether, we thus have the following corollary.

2.11 Corollary: Given an arc st of $H_\alpha^\wedge$ or $H_\alpha^\vee$, there is an algorithm that decides in time $\mathcal{O}(\mathsf{T}_{\text{query}})$ whether st is an arc of G_α. Successive application of this algorithm computes G_α from H_α in time $\mathcal{O}(m_\alpha \cdot \mathsf{T}_{\text{query}})$, where m_α denotes the number of arcs in H_α. †

After being able to reduce a given weak contraction hierarchy to G_α, let us investigate the reverse process, i.e. enlarging a weak contraction hierarchy. To this end, observe that the union of two weak contraction hierarchies H_α and K_α, i.e. the pair $(H_\alpha^\wedge \cup K_\alpha^\wedge, H_\alpha^\vee \cup K_\alpha^\vee)$, is again a weak contraction hierarchy since (w1), (w2) and (w3) are all trivially satisfied. Thus, it turns out that there is not only a smallest but also a unique largest weak contraction hierarchy: The union of all weak contraction hierarchies $H_\alpha \supseteq G_\alpha$. We denote this largest weak contraction hierarchy by M_α. The existence and structure of M_α are not only of mere theoretical interest but turn out to be central to our study of search space sizes in contraction hierarchies in Chapter 3. The following remark might be understood as a first indication of the importance of M_α, for it allows us to infer upper bounds on the search space sizes in G_α or H_α from upper bounds on the search space sizes in the maximal weak contraction hierarchy M_α.

2.12 Remark: Let G be a subgraph of an acyclic digraph H. Then $\mathcal{S}(u,G) \subseteq \mathcal{S}(u,H)$ for all $u \in V$. This means in particular, that

$$\begin{aligned} \mathcal{S}(u,G_\alpha^\wedge) &\subseteq \mathcal{S}(u,H_\alpha^\wedge) \subseteq \mathcal{S}(u,M_\alpha^\wedge), \\ \mathcal{R}(u,G_\alpha^\vee) &\subseteq \mathcal{R}(u,H_\alpha^\vee) \subseteq \mathcal{R}(u,M_\alpha^\vee) \end{aligned}$$

and hence

$$\begin{aligned} \#\mathcal{S}(u,G_\alpha^\wedge) &\leq \#\mathcal{S}(u,H_\alpha^\wedge) \leq \#\mathcal{S}(u,M_\alpha^\wedge), \\ \#\mathcal{R}(u,G_\alpha^\vee) &\leq \#\mathcal{R}(u,H_\alpha^\vee) \leq \#\mathcal{R}(u,M_\alpha^\vee) \end{aligned}$$

for all weak contraction hierarchies H_α and all vertices $u \in V$. Consequently, we also have

$$\mathcal{S}_{\max}(G_\alpha) \leq \mathcal{S}_{\max}(H_\alpha) \leq \mathcal{S}_{\max}(M_\alpha)$$

and

$$\mathcal{S}_{\text{avg}}(G_\alpha) \leq \mathcal{S}_{\text{avg}}(H_\alpha) \leq \mathcal{S}_{\text{avg}}(M_\alpha).$$

for all weak contraction hierarchies H_α of G. †

Proof: Let $v \in \mathcal{S}(u, G)$. There is then a path from u to v in G. This path is of course also a path in $H \supseteq G$, which proves $v \in \mathcal{S}(u, H)$ and hence $\mathcal{S}(u, G) \subseteq \mathcal{S}(u, H)$. The implications for G_α, H_α and M_α follow immediately since $G_\alpha \subseteq H_\alpha \subseteq M_\alpha$ by the definition of weak contraction hierarchies and the definition of M_α. □

Admittedly, the definition of M_α as it stands is not very practical – neither for the deployment of actual algorithms nor for some of our arguments below. In what follows, we therefore develop another description of M_α akin to the definition (2.1) of $\bar{G}_\alpha$. This description additionally brings forth an algorithm to compute M_α, which closely resembles Algorithm 2.2. The following remark is a step in the direction of this very characterisation of M_α.

2.13 Remark: A weak contraction hierarchy H_α is maximal if and only if H_α possesses the following two properties.

(i) Each arc of G is contained in H_α.

(ii) There are no two arcs $uv \in B_\alpha^\vee$ and $vw \in B_\alpha^\wedge$, such that H_α does not contain the arc uw. †

Proof: It is clear that M_α satisfies conditions (i) and (ii). On the other hand, let us consider any weak contraction hierarchy H_α satisfying the conditions (i) and (ii). We want to show that H_α is in fact nothing but M_α, i.e. that H_α contains any other weak contraction hierarchy H'_α. To this end, we do induction on the shortcut depth $\operatorname{scd}(uw)$ of the arcs of H'_α.

$\operatorname{scd}(uw) = 1$: In this case uw is an arc of G, which is contained in H_α by (i).

$\operatorname{scd}(uw) > 1$: Consider the supporting arcs $\sup(uw) = (uv, vw)$ of uw. By construction, $\operatorname{scd}(uv) < \operatorname{scd}(uw)$ and $\operatorname{scd}(vw) < \operatorname{scd}(uw)$ and the induction hypothesis hence implies that both uv and vw are contained in H_α. According to (ii), uv is then an arc of H_α, too. □

Let us now reconsider Algorithm 2.2 from Chapter 2.1. This algorithm iteratively constructs the intermediate graphs $G(i)$ obtained by contraction of the first $(i-1)$ vertices of G, while collecting all the arcs uv of $\bar{G}_\alpha$. In view of the above remark, we may modify this algorithm to compute M_α by inserting not only the strictly necessary but all possible shortcuts. In order to make this precise, let us denote by $G\langle v\rangle = (V_{\langle v\rangle}, A_{\langle v\rangle})$ the digraph obtained from G by removing v and inserting an arc uw for each and every path (u, v, w) in G. We say that $G\langle v\rangle$ is obtained from G by *weak contraction* of v. Mimicking our definitions from Chapter 2.1, we write $G\langle i\rangle = (V_{\langle i\rangle}, A_{\langle i\rangle})$ to denote the digraph G after weak contraction of the vertices $\alpha^{-1}(1), \dots, \alpha^{-1}(i-1)$ in the given order. Let us now consider Algorithm 2.5, which is nothing but Algorithm 2.2 with ordinary contraction of vertices replaced by weak contraction. A close look at

this algorithm reveals that the digraphs $H_\alpha^\wedge = (V, B_\alpha^\wedge)$ and $H_\alpha^\vee = (V, B_\alpha^\vee)$ returned by this algorithm are given by

$$\begin{aligned} B_\alpha^\wedge &= \left\{ uv \in \bigcup_{i=1}^{n} A_{\langle i \rangle} \;\middle|\; \alpha(u) < \alpha(v) \right\} \\ \text{and } B_\alpha^\vee &= \left\{ uv \in \bigcup_{i=1}^{n} A_{\langle i \rangle} \;\middle|\; \alpha(u) > \alpha(v) \right\}. \end{aligned} \tag{2.31}$$

The reader should note the similarity of (2.31) to our definition (2.1) of $\bar{G}_\alpha$, where we defined $A_\alpha^\wedge$ and $A_\alpha^\vee$ to contain precisely those arcs of $\bigcup G(i)$ pointing upwards and pointing downwards, respectively.

Algorithm 2.5: Computation of the maximal weak contraction hierarchy $M_\alpha = (M_\alpha^\wedge, M_\alpha^\vee)$

Input: Weighted digraph $G = (V, A)$, an order α on V
Output: Maximal weak contraction hierarchy $M_\alpha = (M_\alpha^\wedge, M_\alpha^\vee)$

$B_\alpha^\wedge \leftarrow \varnothing$
$B_\alpha^\vee \leftarrow \varnothing$
for $i = 1$ **to** n **do**
  /* $V = V_{\langle i \rangle}$ and $A = A_{\langle i \rangle}$ */
  $B_\alpha^\wedge \leftarrow B_\alpha^\wedge \cup \left\{ uv \in A \mid \alpha(u) < \alpha(v) \right\}$
  $B_\alpha^\vee \leftarrow B_\alpha^\vee \cup \left\{ uv \in A \mid \alpha(u) > \alpha(v) \right\}$
  $v \leftarrow \alpha^{-1}(i)$
  Weakly contract v
return $M_\alpha^\wedge = (V, B_\alpha^\wedge)$ and $M_\alpha^\vee = (V, B_\alpha^\vee)$

We now show that Algorithm 2.5 actually computes M_α. To this end, let us first assure that $H_\alpha = (H_\alpha^\wedge, H_\alpha^\vee)$ is indeed a weak contraction hierarchy

2.14 Remark: The digraphs $H_\alpha = (H_\alpha^\wedge, H_\alpha^\vee)$ as computed by Algorithm 2.5 form a weak contraction hierarchy. †

Proof: Observe that $G(i) \subseteq G\langle i \rangle$ follows from an easy inductive argument employing the fact that each shortcut inserted upon contraction of some vertex v of G is also inserted upon weak contraction of v. Comparing (2.31) to the definition (2.1) of $\bar{G}_\alpha$, we therefore find $\bar{G}_\alpha \subseteq H_\alpha$ and hence $G_\alpha \subseteq H_\alpha$. Moreover, all the arcs of H_α do have supporting arcs by the construction of H_α. As (w2) is also trivially satisfied by H_α, it is indeed a weak contraction hierarchy. □

Note that $G = G\langle 1\rangle$ and that H_α hence contains all the arcs of G. Furthermore, weak contraction of v causes the insertion of uw for any two arcs $uv \in B_\alpha^\vee$ and $vw \in B_\alpha^\wedge$ since these arcs have to be contained in $G\langle \alpha^{-1}(v)\rangle$ if they are contained in H_α. Altogether, we may conclude that H_α satisfies both conditions (i) and (ii) of № 2.13 and we thus have proven $H_\alpha = M_\alpha$ as claimed.

There is yet one subtlety we did not touch upon. Up to this point, we have only shown that Algorithm 2.5 computes the digraph M_α but completely ignored the arc lengths $\mathrm{len}_M^\alpha(-)$ therein. Luckily, only a slight alteration of our notion of weak contraction is necessary to sort out this last problem. Recall from Chapter 2.1 that the arc length $\mathrm{len}_G^\alpha(uw)$ of an arc uw of $\bar{G}_\alpha$ is given by the minimum length $\mathrm{len}_{G(i)}(uw)$ of uw in the graphs $G(i)$. With a little abuse of notation, our definition (2.26) of $\mathrm{len}_M^\alpha(-)$ may also be written as

$$\mathrm{len}_M^\alpha(uw) = \min\left\{\, \mathrm{len}_G(uw), \min\Big\{ \mathrm{len}_M^\alpha(uv) + \mathrm{len}_M^\alpha(vw) \;\Big|\; uv \in B_\alpha^\vee, vw \in B_\alpha^\wedge \Big\} \right\},$$

so that we may actually adopt this approach, i.e. determine $\mathrm{len}_M^\alpha(uw)$ by taking the minimum over the lengths of all the paths (u, v, w) that get removed from G prior to u or v during the course of Algorithm 2.5. In terms of the graphs $G\langle i\rangle$, this is equivalent to defining the length $\mathrm{len}_{G\langle v\rangle}(uw)$ of a shortcut $uw \in A_{\langle v\rangle}$ that is inserted upon weak contraction of v by

$$\mathrm{len}_{G\langle v\rangle}(uw) = \min\Big\{ \mathrm{len}_G(uw), \mathrm{len}_G(uv) + \mathrm{len}_G(vw) \Big\}$$

and letting $\mathrm{len}_M^\alpha(uw)$ be given by the minimum length $\mathrm{len}_{G\langle i\rangle}(uw)$ over all graphs $G\langle i\rangle$ that actually contain uw. An easy induction on the shortcut depth $\mathrm{scd}(uw)$ indeed proves these arc lengths to coincide with the ones we defined in (2.26) above. A properly adapted implementation of weak contraction in pseudo code may also be seen in Algorithm 2.6. Altogether, Algorithm 2.5 thus computes M_α and it is not hard to see that its running time is bounded by $\mathcal{O}(n + m_\alpha)$, where m_α denotes the number of arcs of M_α. We have thus proven the following proposition.

2.15 Proposition: Algorithm 2.5 determines the maximal weak contraction hierarchy M_α of a given weighted digraph $G = (V, A)$ in time $\mathcal{O}(n + m_\alpha)$, where m_α denotes the arcs of M_α. †

We remark that Algorithm 2.5 may be adapted to dynamically update the edge lengths of a given weak contraction hierarchy in time $\mathcal{O}(\mathsf{T}_{\text{query}})$ per update. As all our upper bounds on $\mathcal{S}_{\max}(G_\alpha)$ and $\mathcal{S}_{\max}(H_\alpha)$ that we prove in Chapter 3 are actually bounds on $\mathcal{S}_{\max}(M_\alpha)$, these dynamic updates retain our guarantees on the maximum search space size in the contraction hierarchy in question.

Being able to compute M_α in time $\mathcal{O}(n + m_\alpha)$ also allows us to compute G_α in time $\mathcal{O}(n + m_\alpha \cdot \mathsf{T}_{\text{query}})$ by № 2.11, hence the following corollary.

2.16 Corollary: There is an algorithm that computes G_α in time $\mathcal{O}(n + m_\alpha \cdot \mathsf{T}_{\text{query}})$, where m_α denotes the number of arcs in M_α and where $\mathsf{T}_{\text{query}}$ is the running time of Algorithm 2.3 in M_α. †

This corollary is in fact worth stating, for there are bounds on and m_α in terms of parameters depending on G and not on M_α. As a step towards these bounds, the following remarks relates m_α to the average search space size $\mathcal{S}_{\text{avg}}(M_\alpha)$.

2.17 Remark: The number of arcs in some weak contraction hierarchy H_α is bounded from above by $n \cdot \mathcal{S}_{\text{avg}}(H_\alpha)$. †

Proof: As above, we denote by $B_\alpha^\wedge$ and $B_\alpha^\vee$ the arcs of H_α. Observe that the number of arcs $uv \in B_\alpha^\wedge$ with source u is certainly less than or equal to $\#\mathcal{S}(u, H_\alpha^\wedge)$. Likewise, the number of arcs $uv \in B_\alpha^\vee$ with prescribed target v is less than or equal to $\#\mathcal{R}(v, H_\alpha^\vee)$. Utilising these observations, we compute

$$\begin{aligned}\left|B_\alpha^\wedge\right| + \left|B_\alpha^\vee\right| &= \sum_{u \in V} \left|\left\{ uv \in B_\alpha^\wedge \right\}\right| + \sum_{v \in V} \left|\left\{ uv \in B_\alpha^\vee \right\}\right| \\ &\leq \sum_{u \in V} \#\mathcal{S}(u, H_\alpha^\wedge) + \sum_{v \in V} \#\mathcal{S}(v, H_\alpha^\vee) = n \cdot \mathcal{S}_{\text{avg}}(H_\alpha)\,,\end{aligned}$$

which finishes the proof. □

Algorithm 2.6: Weak contraction of a single vertex

Input: Weighted digraph $G = (V, A)$, a vertex $v \in V$
Output: $G\langle v \rangle = (V_v, A_v)$, which is G after weak contraction of v

1 $V_{\langle v \rangle} \leftarrow V \setminus \{v\}$
2 $A_{\langle v \rangle} \leftarrow \left\{ xy \in A \,\middle|\, v \notin \{x, y\} \right\}$
3 **foreach** $uv, vw \in A$ **do**
4 $\quad A_{\langle v \rangle} \leftarrow A_{\langle v \rangle} \cup \{uw\}$
5 $\quad \text{len}_{G\langle v \rangle}(uw) \leftarrow \min\left\{ \text{len}_G(uw), \text{len}_G(uv) + \text{len}_G(vw) \right\}$
6 **return** $G\langle v \rangle = (V_{\langle v \rangle}, A_{\langle v \rangle})$

3 Upper Bounds on Search Space Size via Nested Dissection

This chapter is organised as follows. In Chapter 3.1 we recall the definition of the filled graph G^α that can be associated with any undirected graph G and an order α on its vertices. Given our previous work on contraction hierarchies, it appears that the maximal weak contraction hierarchy M_α is nothing but a directed variant of the filled graph G^α. Motivated by this similarity of G^α and M_α, we then explore how certain invariants of G^α, that is, the elimination tree $T(G^\alpha)$ and its height $\mathrm{ht}(G^\alpha)$, relate to the search spaces in M_α. More explicitly, we show that the height $\mathrm{ht}(G^\alpha)$ of the elimination tree $T(G^\alpha)$ is an upper bound on the maximum search space size $\mathcal{S}_{\max}(M_\alpha)$ and hence also on $\mathcal{S}_{\max}(G_\alpha)$ by № 2.12. We also touch on secondary aspects like the computational complexity of finding the elimination tree of minimum height and the feasibility of a lower bound on $\mathcal{S}_{\max}(G_\alpha)$ in terms of the height of an elimination tree. Finding elimination trees of low height is a well studied problem and in Chapter 3.2 we investigate a particular heuristic known as nested dissection. Utilising results about the performance of this heuristic, we obtain an upper bound of $\log(n) \cdot \mathrm{tw}(G)$ on $\mathcal{S}_{\max}(G)$, where n denotes the number of vertices of G and $\mathrm{tw}(G)$ its treewidth. In many cases of practical relevance, this upper bound is further accompanied by an algorithm to compute orders α that actually attain this bound. By virtue of our findings in Chapter 3.1, these algorithms may be used without any further modification to compute orders α such that $\mathcal{S}_{\max}(M_\alpha) \leq \log(n) \cdot \mathrm{tw}(G)$. We further show how to improve on this upper bound on $\mathcal{S}_{\max}(M_\alpha)$ for a minor-closed class of graphs admitting balanced separators of size $\mathcal{O}(\sqrt{n})$ in Chapter 3.3.

Finally, we try to relate our results to previous work of Abraham et al. in Chapter 3.4. Abraham et al. study the performance of contraction hierarchies in terms of a parameter called highway dimension. We show that there are edge lengths on any undirected graph G, such that nested dissection orders α and the orders β as computed by the algorithms of Abraham et al. yield contraction hierarchies of comparable performance.

3.1 Contraction Hierarchies and Filled Graphs

In this paragraph we recall the definition of the filled graph G^α, the associated elimination tree $T(G^\alpha)$ and its height $\mathrm{ht}(G^\alpha)$. Since our definitions deviate slightly from the ones commonly found in the literature and since we explicitly carve out similarities to contraction hierarchies, more than a quick skimming might be worth it even for

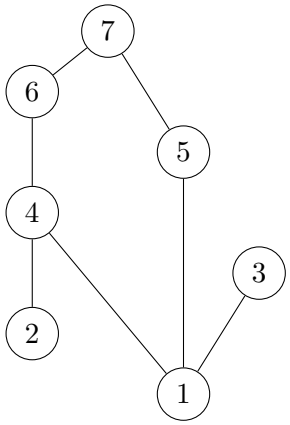
the knowledgeable reader. The filled graph G^α was introduced by Parter in his analysis [Par61] of Gaussian elimination. The related notion of an elimination tree was introduced only two decades later by Schreiber [Sch82]. Both the elimination tree and the filled graph have been studied extensively by numerous authors. A comparatively old but still valuable introduction to the various applications of elimination trees in sparse factorisation algorithms is the survey article by Liu [Liu90]. The filled graph and the elimination tree are also closely related to chordal graphs, chordal completion and treewidth. The article of Bodlaender et al. [BGKH91] includes a good survey of the interrelation of all these notions. A prominent application of the filled graph in this context is their use in linear-time algorithms for chordal graph recognition based on a characterisation of chordal graphs by Fulkerson and Gross [FG65] in terms of elimination orders.

Let us fix an undirected graph $G = (V, E)$ with $n = |V|$ vertices and an order $\alpha\colon V \to [n]$. We consider the so-called *elimination game* played on G. Beginning with $G^1 = G$, one removes in each step $i \in [n]$ the vertex $v_i = \alpha^{-1}(i)$ and all its incident edges from G^i. Afterwards, one obtains the graph G^{i+1} by inserting a *fill-edge* between each pair of remaining neighbours of v_i. That is, the remaining neighbourhood of v_i becomes a clique. Let F^i be the set of edges inserted upon removal of v_i and denote by $F = \bigcup_{i=1}^{n} F^i$ the whole of fill-edges during the course of this game. The filled graph G^α is now commonly defined to be the undirected graph with edges $E \cup F$. For our purposes it is more convenient to define the filled graph as the accordant directed graph with all arcs pointing upwards with respect to α. That is, the *filled graph* G^α is the acyclic digraph $G^\alpha = (V, A^\alpha)$ given by

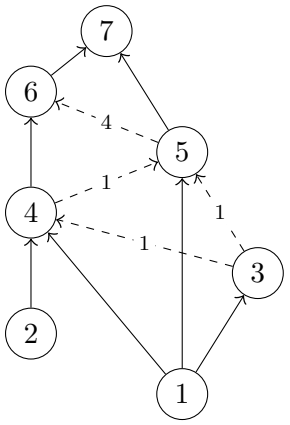
$$A^\alpha = \left\{ uv \;\middle|\; \{u, v\} \in E \cup F \text{ and } \alpha(u) < \alpha(v) \right\}. \tag{3.1}$$

An example of an undirected graph $G = (V, E)$ and one of its filled graphs G^α may also be seen in Figure 3.1. The reader should note that the construction of the filled graph G^α bears a strong resemblance to Algorithm 2.5 from Chapter 2.4 that computes the maximal weak contraction hierarchy M_α from a given digraph and an order on its vertices. More precisely, during both the elimination game and Algorithm 2.5 the vertices $\alpha^{-1}(1), \ldots, \alpha^{-1}(n)$ of G are successively removed from G and an additional arc uw or edge $\{u, w\}$ is inserted for each path (u, v, w) containing the most recently deleted vertex v. The only difference is that Algorithm 2.5 operates on directed graphs while the elimination game is played on undirected graphs.

In order to explore the relation between the maximal weak contraction hierarchy M_α and the filled graph G^α in some greater detail, let us fix a digraph $G = (V, A)$ and let ${}^*G = (V, E)$ denote the associated undirected graph. Further let α be an order on the vertices V. It is clear that there is an undirected path (u, v, w) in ${}^*G$ for each directed path (u, v, w) in G. Any shortcut uw inserted during the first step of Algorithm 2.5 due to the presence of a directed path (u, v, w) therefore gives rise to a fill-edge $\{u, w\}$ that is inserted during the very first step of the elimination game played on ${}^*G$. Again,



(a) A graph $G = (V, E)$ with 7 vertices. The numbers shown inside the nodes correspond to the respective values $\alpha(v)$ of an order α on V.



(b) The filled graph G^α associated with the graph G and the map α. Arcs corresponding to fill-edges are drawn dashed and labelled with $\alpha(v)$ where v is the vertex whose removal led to their insertion.

Figure 3.1: A graph $G = (V, E)$ and one of its filled graphs.

we find a corresponding undirected path in the remainder ${}^*G^2$ of ${}^*G$ for each directed path in the remainder $G\langle 2\rangle$ of G. We now see, by induction, that $M_\alpha^\wedge$ and $M_\alpha^\vee$ with reversed arcs are subgraphs of ${}^*G^\alpha$. Even though our previous work renders the proof trivial, the consequences of this very observation are of such significance, that we state it as a genuine theorem.

Theorem 3 *Maximal weak contraction hierarchies and filled graphs*
Let $G = (V, A)$ be a directed graph and let α be an order on its vertices. Further let ${}^*G$ be the undirected graph associated with G and let $M_\alpha = (M_\alpha^\wedge, M_\alpha^\vee)$ be the maximal weak contraction hierarchy. Then both $M_\alpha^\wedge$ and $M_\alpha^\vee$ with reversed arcs are subgraphs of the filled graph ${}^*G^\alpha$. If G was undirected in the first place, that is, if G contained for each arc $uv \in A$ also the reverse arc vu, then all three graphs $M_\alpha^\wedge$, $M_\alpha^\vee$ with reversed arcs and the filled graph G^α are even equal. ‡

The above theorem has many far-reaching consequences of which we are only able to explore very few in this thesis. Filled graphs are an extensively studied subject and a vast amount of work has been invested in their analysis since their introduction by Parter [Par61]. It even turns out that the maximal weak contraction hierarchy M_α is nothing essentially new but has already been defined and studied by Rose and Tarjan [RT78] in 1978. However, much of the work on filled graphs is primarily concerned with the problem of minimising the number of arcs in G^α – a problem commonly known as *minimum fill-in*. The beautiful survey by Heggernes [Hego6] contains an overview of many classic and recent developments concerning this very

problem. In view of Theorem 3, the question how to minimise the number of arcs in G^α is also of importance in the context of contraction hierarchies, for the number of fill-arcs is certainly an upper bound on the number of shortcuts in G_α and hence on its space requirements. Although the arguments concerning the computational complexity of the minimum fill-in problem due to Yannakakis [Yan81] do not carry over to contraction hierarchies, it is already known that deciding whether there is a contraction hierarchy G_α with less than k arcs is NP-complete [Colo9; BCKKW10]. Given these hardness results, it seems quite appropriate to take known heuristics for the minimum fill-in problem as a starting point for the design of algorithms minimising the space requirements of contraction hierarchies. A quite recent development seems to be of particular interest also with respect to our findings concerning the maximum search space size. Instead of trying to find an order α that minimises the number of arcs in G^α, researchers have been studying the easier problem of determining an order β minimising the number of arcs in G^β while retaining the condition $G^\beta \subseteq G^\alpha$ for an initially given order α. This problem is known as the *minimal triangulation sandwich problem* and there are efficient and practical algorithms due to Peyton [Peyo1] and Heggernes and Peyton [HPo8] among others. Beginning with a maximal weak contraction hierarchy M_α that has small maximum search space size, these algorithms can be used to reduce the space requirements of M_α retroactively without increasing their search space sizes. It seems definitely worth the effort to investigate the effect of these algorithms on M_α and their "compatibility" with the computation of G_α from M_α from № 2.16.

Instead of the above ramifications concerning the space requirements of contraction hierarchies, we rather concentrate on the implications of Theorem 3 regarding the search spaces and their size. To this end, consider the following corollary of Theorem 3.

3.1 Corollary: Let $G = (V, A)$ be a weighted directed graph, let α be an order on its vertices and let ${}^*G$ be the undirected graph associated with G. Then

$$\mathcal{S}(u, H_\alpha^\wedge) \subseteq \mathcal{S}(u, {}^*G^\alpha) \qquad \text{and} \qquad \mathcal{R}(u, H_\alpha^\vee) \subseteq \mathcal{S}(u, {}^*G^\alpha)$$

for all $u \in V$ and all weak contraction hierarchies $H_\alpha \supseteq G_\alpha$. Consequently

$$\#\mathcal{S}(u, H_\alpha^\wedge) \leq \#\mathcal{S}(u, {}^*G^\alpha) \qquad \text{and} \qquad \#\mathcal{R}(u, H_\alpha^\vee) \leq \#\mathcal{S}(u, {}^*G^\alpha)$$

for all u and H_α as above. Summation over all the vertices $u \in V$ finally gives

$$\mathcal{S}_{\max}(H_\alpha) \leq \mathcal{S}_{\max}({}^*G^\alpha) \qquad \text{and} \qquad \mathcal{S}_{\text{avg}}(H_\alpha) \leq \mathcal{S}_{\text{avg}}({}^*G^\alpha)$$

for all weak contraction hierarchies H_α. †

Proof: Follows immediately from № 2.12 and Theorem 3. □

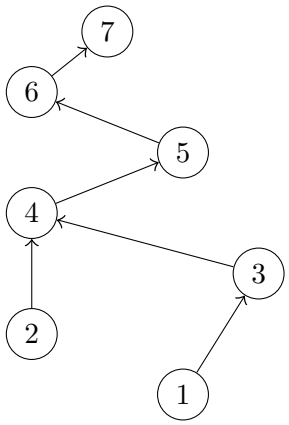


Figure 3.2: The elimination tree $T(G^\alpha)$ associated with the filled graph shown in Figure 3.1. It is rooted in the vertex 7 and has height $\mathrm{ht}(G^\alpha) = 5$.

We now give an alternative description of $\mathcal{S}_{\max}({}^*G^\alpha)$ known as the height of the elimination tree of ${}^*G^\alpha$. For this purpose, consider an undirected graph $G = (V, E)$ and one of its filled graphs G^α again. Associated with G^α is the so-called *elimination tree* $T(G^\alpha)$ of G. The elimination tree has the same vertices as G^α but contains only those arcs uv of G^α with minimal $\alpha(v)$. More formally, the arcs of $T(G^\alpha)$ are given by

$$\left\{ uv \;\middle|\; uv \in A^\alpha \text{ and } \alpha(v) \leq \alpha(w) \text{ for all } uw \in A^\alpha \right\}.$$

As it was the case with the filled graph G^α, the elimination tree, too, is usually defined as an undirected tree. There is a natural choice of root in $T(G^\alpha)$, namely $\alpha^{-1}(n)$. The height of $T(G^\alpha)$ with respect to this root is commonly known as *elimination tree height* and we denote it by $\mathrm{ht}(G^\alpha)$. The minimum of $\mathrm{ht}(G^\alpha)$ over all orders α is called *minimum elimination tree height* of G and is denoted by $\mathrm{ht}(G)$. The elimination tree associated with the filled graph of Figure 3.1 is depicted in Figure 3.2 as an example. Deciding for a given graph G whether there is an elimination tree of G of height less than or equal to some given $k \in \mathbb{N}$ is known to be NP-complete even when restricted to co-bipartite graphs due to a result of Pothen [Pot88]. Later, this problem was shown to be NP-complete on other families of graphs like bipartite [BDJKKMT98] and chordal graphs [DN06]. Nevertheless, there are also several classes of graphs for which polynomial-time algorithms determining the elimination tree $T(G^\alpha)$ of minimum height are known. Among these classes are for example trees [Sch89; IRV88; HPS07], interval graphs [AH94], trapezoid, circular-arc or permutation graphs [DKKM94; DKKM99]. Many of these results are actually not explicitly concerned with minimising the height of elimination trees but with an equivalent problem known as *ordered colouring* or *vertex ranking*. There is also an attempt to generalise these problems to directed graphs due to Kratochvíl and Tuza [KT99], which is further explored in the article [Der07] of Dereniowski. Apart from the optimal

vertex ranking algorithms for trees due to Schäffer [Sch89], Iyer, Ratliff and Vijayan [IRV88] and Hsu, Peng and Shi [HPS07] that we shortly touch upon at the end of the next paragraph, we do not investigate any of these issues further, but suggest the above articles as a starting point for more profound studies of vertex ranking and elimination tree height.

Let us now return to our investigation of search spaces in G^α and their claimed relation to the height of the elimination tree $T(G^\alpha)$. For we chose each arc uv in $T(G^\alpha)$ to be directed towards the root, the height of $T(G^\alpha)$ is one less than the maximum number of vertices reachable from any vertex u in $T(G^\alpha)$. Similarly, the search space $\mathcal{S}(u, T(G^\alpha))$ of some vertex u in $T(G^\alpha)$ is the set of vertices in $T(G^\alpha)$ reachable from u. This suggests the following reinterpretation of $\mathrm{ht}(G^\alpha)$ in our framework of search spaces and their size.

3.2 Remark: Let $G = (V, E)$ be a connected graph, α an order on V and let $T = T(G^\alpha)$ be the elimination tree of G with respect to α. Then $\mathcal{S}_{\max}(T) = 1 + \mathrm{ht}(G^\alpha)$. †

Proof: Let r denote the root of $T = T(G^\alpha)$ and denote by $p(u)$ the vertices lying on the path from u to r. Then

$$\mathrm{ht}(G^\alpha) = \max_{u \in V} |p(u)| - 1$$

and it therefore suffices to show $\mathcal{S}(u, T) = p(u)$ for all $u \in V$. This latter equality is trivially satisfied though, for G is connected, hence each vertex $u \in V$ is the source of precisely one arc uv. □

Let us stress one particular property of $T(G^\alpha)$ that is of great interest to us: If u and v are two vertices connected by a path p in the filled graph G^α of G, then there is a path p' from u to v in the elimination tree $T(G^\alpha)$. It obviously suffices to prove this statement when p is an arc, as the general case then follows by induction. This is precisely the subject of the following lemma.

3.3 Lemma: If uv is an arc of the filled graph G^α, then there is a path p with source u and target v in $T = T(G^\alpha)$. †

Proof: Let us denote by $p(u)$ the unique path in $T = T(G^\alpha)$ from u to the root r. Recall that each vertex $u \in V$ is the source of at most one arc uv in T, which is also the first arc of $p(u)$ of course. We show by descending induction on $\alpha(u)$ that we have $p(v) \subseteq p(u)$ for all arcs uv of G^α. Note that this implies our claim, for it then follows that $p(u) = q \cdot p(v)$, where q is a path from u to v.

$\alpha(u) = n - 1$: If $\alpha(u) = n - 1$, then $\alpha(v) = n$ and it follows that $v = r$. There is nothing to show, for the root r is a subpath of $p(u)$ by definition of $p(u)$.

$\alpha(u) < n - 1$: Let uw be the unique arc of T with source u. Note that our choice of w implies $\alpha(w) \leq \alpha(v)$, since T contains only those arcs uw where w is the α-minimal neighbour of u in G^α. The path $p(u)$ is, according to our preliminary remark, nothing but the concatenation $p(u) = uw \cdot p(w)$. If $v = w$, there is again little to show as $p(w) \subseteq p(u)$ is trivially satisfied. If $v \neq w$, then G^α contains the arc vw as both v and w are neighbours of u when it gets removed during the elimination game. The induction hypothesis therefore implies $p(v) \subseteq p(w)$, hence $p(v) \subseteq p(u)$. □

This lemma has one important implication concerning search spaces in the filled graph G^α and in the elimination tree $T = T(G^\alpha)$.

3.4 Corollary: Let $G = (V, E)$ be a graph, α an order on V and $T = T(G^\alpha)$ the corresponding elimination tree. Then $\mathcal{S}(u, T) = \mathcal{S}(u, G^\alpha)$ for all $u \in V$. †

Proof: As T is a subgraph of G^α, it follows that $\mathcal{S}(u, T) \subseteq \mathcal{S}(u, G^\alpha)$ for all $u \in V$ by № 2.12. Conversely, if $v \in \mathcal{S}(u, G^\alpha)$, there is some path $p = (x_1, \ldots, x_\ell)$ with source $x_1 = u$ and target $x_\ell = v$ in G^α. By № 3.3 there exists for each arc $x_i x_{i+1}$ in p some path q_i from x_i to x_{i+1} in T. The concatenation of all these paths q_i is a path in T that has source $x_1 = u$ and target $x_\ell = v$. □

We already saw in № 3.2 that the maximum search space size in the elimination tree T is nothing but its height. As search spaces in the elimination tree and the filled graph coincide by the above corollary, we find that studying elimination tree height is the same as studying maximum search space size in G^α. Moreover, we know from Theorem 3 – or more precisely from № 3.1 – that $\mathcal{S}_{\max}(G_\alpha)$ is at most $\mathcal{S}_{\max}(^*G^\alpha)$. Altogether, we find $\mathcal{S}_{\max}(G_\alpha) \leq \operatorname{ht}(^*G^\alpha) + 1$ and have thus proven the following corollary.

3.5 Corollary: Let $G = (V, E)$ be a connected weighted digraph and let $^*G$ be the associated undirected graph. Then

$$\mathcal{S}_{\max}(H_\alpha) \leq \operatorname{ht}(^*G^\alpha) + 1$$

for any order α on the vertices of G and any weak contraction hierarchy $H_\alpha \supseteq G_\alpha$. †

Despite its innocent appearance, the above corollary is central to our analysis of search spaces in contraction hierarchies, for it enables us to translate upper bounds on $\operatorname{ht}(^*G)$ into upper bounds on $\mathcal{S}_{\max}(G)$. These upper bounds are not seldom accompanied by algorithms to determine orders α so that $\operatorname{ht}(^*G^\alpha)$ is less than the upper bound at hand. Without any further modification these algorithms may be used to compute contraction orders α with good upper bounds on $\mathcal{S}_{\max}(G_\alpha)$.

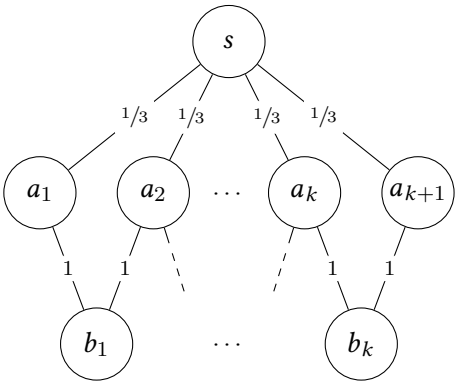
Given № 3.5 and the apparently strong connection between weak contraction hierarchies H_α, filled graphs $^*G^\alpha$ and their elimination trees $T(^*G^\alpha)$, the reader

might wonder if there also exists a lower bound on $\mathcal{S}_{\max}(G_\alpha)$ or $\mathcal{S}_{\max}(M_\alpha)$ in terms of $\mathrm{ht}({}^*G^\alpha)$. In the remainder of this paragraph, we briefly sketch examples rendering such lower bounds impossible if one is not willing to impose further restrictions on either the arc lengths or the connectivity of G. Let us first discuss the feasibility of a lower bound on $\mathcal{S}_{\max}(G_\alpha)$ in terms of $\mathrm{ht}({}^*G^\alpha)$. For each positive integer k, we construct an undirected graph G as follows. Its vertices are composed of a single vertex s and two rows $a_1, a_2, \ldots, a_{k+1}$ and $b_1, b_2, \ldots, b_k$ of vertices. Each a_i is connected to s, b_{i-1} and b_i. If $i - 1 < 1$ or $i > k$, then G does simply not include the corresponding edge $\{a_i, b_{i-1}\}$ or $\{a_i, b_i\}$. The edges have lengths $\mathrm{len}(sa_i) = 1/3$ and $\mathrm{len}(a_i b_{i-1}) = \mathrm{len}(a_i b_i) = 1$. The construction of G is also schematically depicted in Figure 3.3a. We consider the order α on V given by

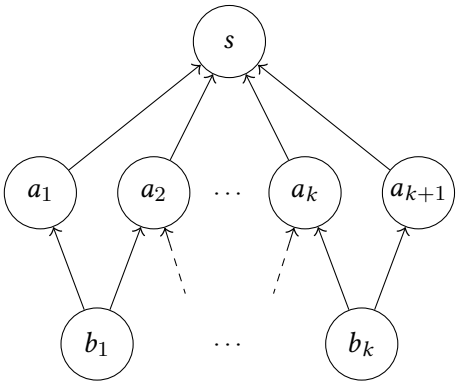
$$\alpha(x) = \begin{cases} i & \text{if } x = b_i \\ k + i & \text{if } x = a_i \\ 2k + 2 & \text{if } x = s\,. \end{cases} \tag{3.2}$$

Less formally, the vertex s gets contracted last and all the vertices b_i are contracted prior to the vertices a_i. Furthermore, the vertices a_i and b_i are ordered by their indices. It is not hard to convince oneself that contraction of G in this order leads to no shortcut at all as each shortest path in G necessarily passes the vertex s that is contracted last. The contraction hierarchy G_α of G hence looks like the one shown in Figure 3.3b. Observe that $\mathcal{S}_{\max}(G_\alpha) = 4$. The filled graph G^α, on the other hand, contains all the fill-arcs $a_i a_{i+1}$ caused by the removal of the vertices b_i during the course of the elimination game. As can also be seen in Figure 3.3c and Figure 3.3d this results in an elimination tree of height $k + 2$. All in all, the gap between $\mathcal{S}_{\max}(G_\alpha)$ and $\mathrm{ht}(G^\alpha)$ is linear in the number of vertices of G and this example renders a lower bound on $\mathcal{S}_{\max}(G_\alpha)$ in terms of $\mathrm{ht}(G^\alpha)$ obviously infeasible. As a side note, we remark that all the graphs G constructed above are fairly elementary in that they are planar and bipartite. The suspicious reader might object that the degree of s is unbounded in this example, but replacing the vertex s and its incident edges with e.g. a rooted binary tree, a similar argument as above shows that the gap between $\mathcal{S}_{\max}(G_\alpha)$ and $\mathrm{ht}(G^\alpha)$ is still of the order of $\Omega(n/\log(n))$. We leave the details to the reader but have sketched the corresponding construction in Figure 3.4.

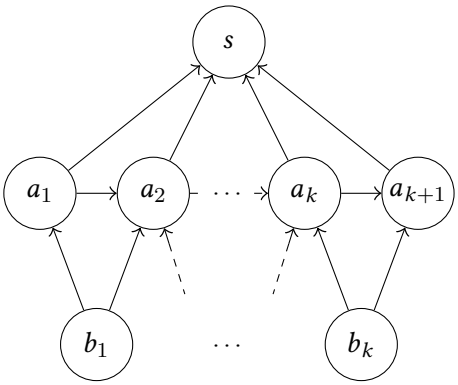
Let us now consider the difference between $\mathcal{S}_{\max}(M_\alpha)$ and $\mathrm{ht}({}^*G^\alpha)$. At first glance, it might seem surprising that there is any gap at all between $\mathcal{S}_{\max}(M_\alpha)$ and $\mathrm{ht}({}^*G^\alpha)$, for we have shown above that M_α and G^α do coincide if G is undirected. Moreover, the preceding examples can obviously not be extended to M_α since they exploited the fact that G_α depends on the edge lengths and G^α does not. However, the weak contraction hierarchy M_α is constructed from the directed graph G, while the filled graph ${}^*G^\alpha$ is built only from the undirected graph ${}^*G$ associated with G. If we choose G to be e.g. a directed acyclic graph whose vertices are topologically sorted by α, then M_α contains no shortcut at all as there occurs no directed path (u, v, w) during contrac-



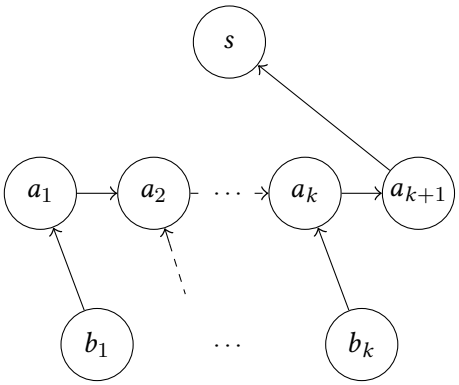
(a) The undirected graph G with its edge lengths drawn on the respective edges.



(b) The contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$ associated with the graph shown to the left and the order α defined in (3.2). Note that $\mathcal{S}_{\max}(G_\alpha) = 4$.

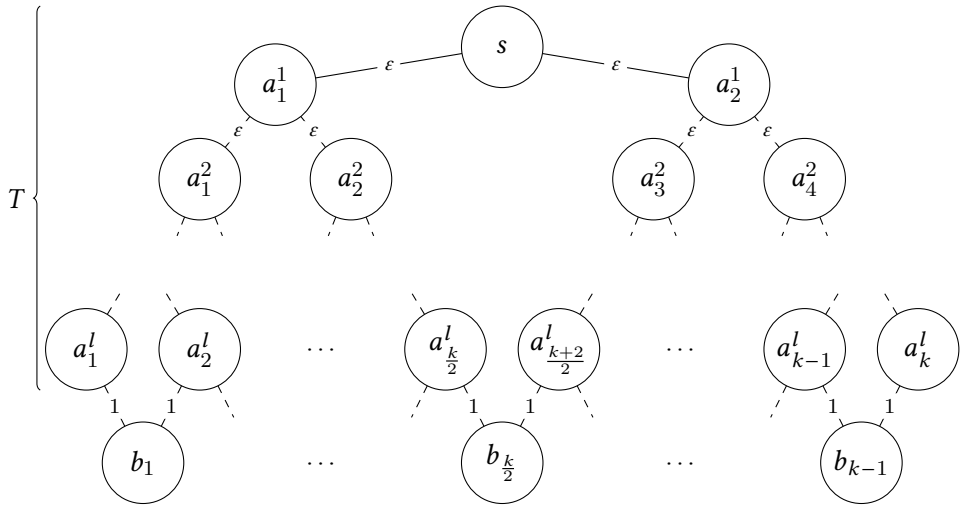


(c) The filled graph G^α associated with the graph shown in Figure 3.3a and the order α defined in (3.2).

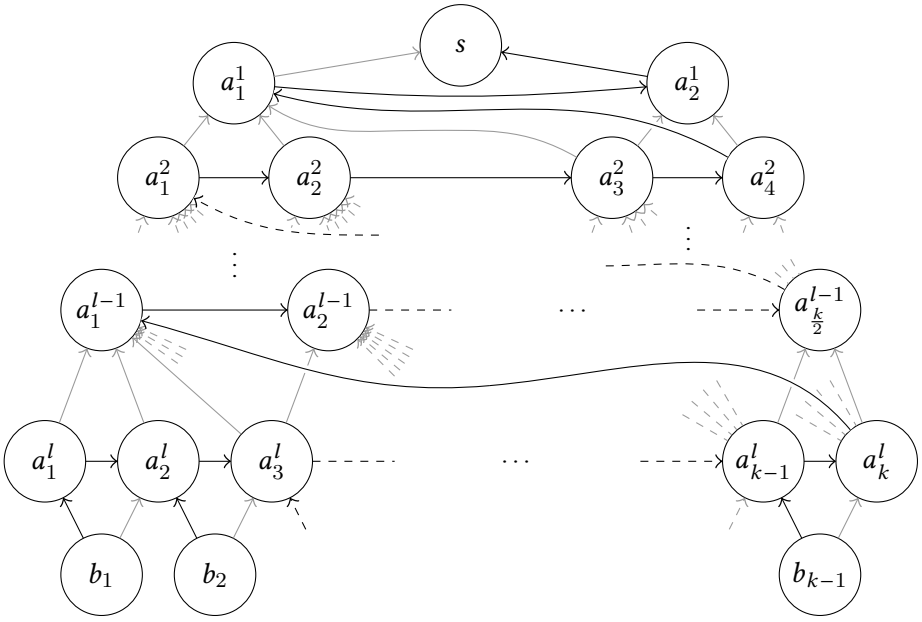


(d) The elimination tree $T(G^\alpha)$ associated with the filled graph G^α shown to the left. Note that $\mathrm{ht}(G^\alpha) = k+2$ as the longest path with target s in $T(G^\alpha)$ is the one with source b_1 of length $k+2$.

Figure 3.3: An example of a family of weighted graphs $G = (V, E)$ and orders α on their vertices, such that the gap between $\mathcal{S}_{\max}(G_\alpha)$ and $\mathrm{ht}(G^\alpha)$ is linear in the number of vertices of G. Note that $G_\alpha^\wedge$ and $G_\alpha^\vee$ with reversed arcs coincide since G is undirected and since $\mathrm{len}_G(xy) = \mathrm{len}_G(yx)$. The single graph $G_\alpha^\wedge$ depicted in Figure 3.3b therefore suffices to completely describe G_α.

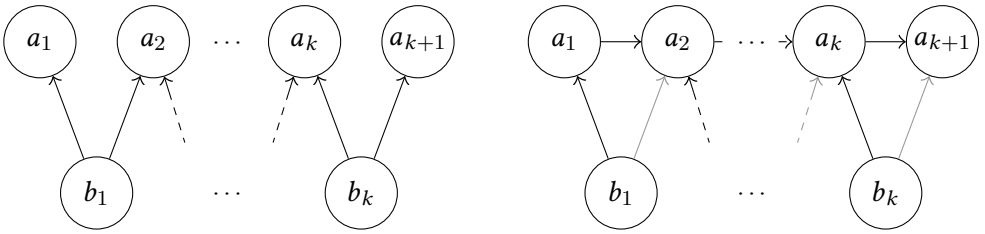


(a) A graph $G = (V, E)$ consisting of a complete binary tree T of height l with $k = 2^l$ leaves $a_1^l, \ldots, a_k^l$ and an additional row $b_1, \ldots, b_{k-1}$. Each b_i is connected to a_i^l and a_{i+1}^l by an edge of length 1. All edges of T have length $\varepsilon < 1/2l$, so that each shortest path of G lies inside of T. If contraction takes place bottom-up and from left to right in each row, there is no shortcut at all in the corresponding contraction hierarchy G_α of G. The maximum search space size $\mathcal{S}_{\max}(G_\alpha)$ hence equals $l + 2$.



(b) Relevant parts of the filled graph G^α corresponding to the elimination game played on the above graph G in bottom-up and left to right order. The arcs also occurring in the elimination tree $T(G^\alpha)$ are drawn black. All other arcs are shaded in grey. Note that $\mathrm{ht}(G^\alpha) = 2^{l+1} - 1$ since the longest path with target s is essentially a zig-zag path starting in b_1 and passing all the vertices a_i^j.

Figure 3.4: A family of undirected graphs $G = (V, E)$ with $n = 3 \cdot 2^l - 2$ vertices, bounded degree and a gap between $\mathcal{S}_{\max}(G_\alpha)$ and $\mathrm{ht}(G^\alpha)$ of order $\Omega(n/\log(n))$.



(a) The directed acyclic graph G. We suppose that vertices are contracted bottom-up and from left to right in each row. Note that the downwards directed graph $M_\alpha^\vee$ of the maximal weak contraction hierarchy M_α contains no arcs at all. Furthermore, $M_\alpha^\wedge = G$ since there are no shortcuts in M_α. Therefore, $\mathcal{S}_{\max}(M_\alpha) = 3$.

(b) The filled graph ${}^*G^\alpha$ resulting from the elimination game played on the undirected graph associated with the graph G shown to the left. Vertices are removed bottom-up and from left to right in each row. The arcs also occurring in the elimination tree are drawn black and all other arcs are drawn in shades of grey. Observe that $\mathrm{ht}({}^*G^\alpha) = k + 1$, which is linear in the number of vertices of G.

Figure 3.5: An example of a family of weighted digraphs $G = (V, E)$ and orders α on their vertices, such that the gap between $\mathcal{S}_{\max}(M_\alpha)$ and $\mathrm{ht}({}^*G^\alpha)$ is linear in the number of vertices of G.

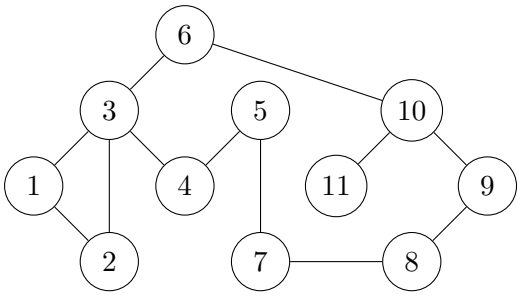
tion of some vertex v. The filled graph G^α may nevertheless contain many fill-arcs. The height $\mathrm{ht}(G^\alpha)$ of the corresponding elimination tree may thus be significantly larger than $\mathcal{S}_{\max}(M_\alpha)$. This phenomenon may also be seen in Figure 3.5. For a particular choice of α, the digraph G depicted therein has a maximal weak contraction hierarchy M_α with $\mathcal{S}_{\max}(M_\alpha) = 3$ but the elimination tree height $\mathrm{ht}(G^\alpha)$ for this particular order α is linear in the number of vertices.

Although the preceding examples render a lower bound on $\mathcal{S}_{\max}(G_\alpha)$ or $\mathcal{S}_{\max}(M_\alpha)$ in terms of $\mathrm{ht}({}^*G^\alpha)$ impossible in general, some unanswered questions remain that are worth mentioning. It is not clear whether appropriate conditions on the arc lengths of G exist, such that the gap between $\mathcal{S}_{\max}(G_\alpha)$ and $\mathrm{ht}({}^*G^\alpha)$ may be bounded by some slowly growing function or a constant. Similarly, finding a relation between the connectivity of a digraph G and the gap between $\mathcal{S}_{\max}(M_\alpha)$ and $\mathrm{ht}(G^\alpha)$ seems to be a difficult but interesting task. Moreover, there is also a more conceptual issue with our above examples. We always consider the filled graph ${}^*G^\alpha$ and the contraction hierarchy G_α with respect to the same order α. We leave it as an open question, whether orders α with contraction hierarchies G_α that have small maximum search space size $\mathcal{S}_{\max}(G_\alpha)$ may be transformed into orders β, such that the resulting filled graphs ${}^*G^\beta$ have small elimination tree height. A positive answer to this question would automatically imply the existence of approximation algorithms for the problem of finding a contraction hierarchy with minimum maximum search space size $\mathcal{S}_{\max}(G)$ as we shall see below.

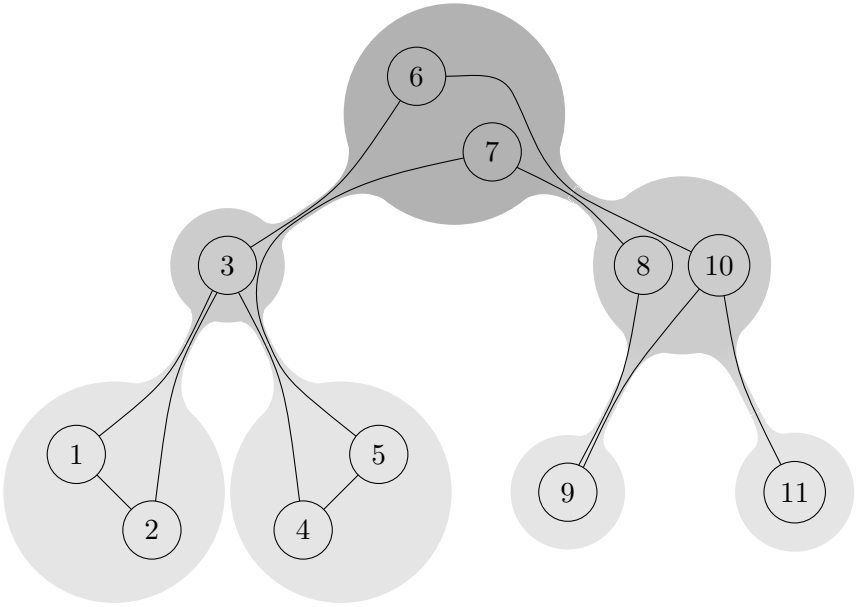
3.2 Nested Dissection

As the computation of an order α that minimises $\mathrm{ht}(G^\alpha)$ is rendered computationally infeasible for a large class of graphs by virtue of the above-mentioned hardness results [Pot88; BDJKKMT98; DN06], one must be content with heuristics or approximation algorithms. In this paragraph, we revisit a particular heuristic going back to the work of George [Geo73] that is commonly known as *nested dissection*. The basic idea of nested dissection is to separate a given graph $G = (V, E)$ into pieces $G_1, \ldots, G_d$ of approximately equal size by the removal of a subset $S \subseteq V$, place the elements of S last with respect to the prospective order α on V and recursively order the vertices contained in the subgraphs $G_1, \ldots, G_d$ of G. Employing approximate balanced minimum separators in this very algorithm, Bodlaender et al. [BGKH91] show that nested dissection gives in fact a $r \cdot \log(n)$-approximation to $\mathrm{ht}(G)$, where r denotes the approximation ratio of the algorithm computing the separators. To the author's knowledge, the current best approximation ratio of $r = \sqrt{\log \mathrm{opt}}$, where opt denotes the size of a minimum balanced separator, is achieved by the algorithm of Feige, Hajiaghayi and Lee [FHL05]. Their algorithm even turns out to approximate balanced minimum separators within only a constant factor on families of graphs that exclude a fixed minor, such as the family of planar graphs. However, we have seen above that our current techniques do not suffice to transform approximations to $\mathrm{ht}(G)$ into approximations to $\mathcal{S}_{\max}(G)$. As all known approximation algorithms for balanced minimum separators are additionally not able to cope with graphs as large as those one typically wants to compute a contraction hierarchy of, we do not pursue this line of thought any further.

In order to make the notion of nested dissection precise, we suppose to be given a (b, f)-*balanced separator decomposition* of G, that is, a rooted tree $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ whose *nodes* $X \in \mathcal{X}$ are subsets of the vertices V of G and that is recursively defined as follows. Let n denote the number of vertices in G. If $n \leq n_0$ for some fixed constant n_0, then a (b, f)-balanced separator decomposition of $G = (V, E)$ consists of a single node $X = V$. If $n > n_0$, then a (b, f)-balanced separator decomposition of G consists of a root $X \subseteq V$ of size at most $f(n)$ whose removal separates G into at least two subgraphs $G_1, \ldots, G_d$ with at most bn vertices each. The children of X in $\mathcal{T}$ are then the roots of (b, f)-balanced separator decompositions of the connected components $G_1, \ldots, G_d$ of G after removal of X. We thereby assume that $0 < b < 1$ is a constant and that f is a not necessarily strictly increasing function that assumes its values in the positive reals. Note, that we deliberately call the vertices of $\mathcal{T}$ nodes and those of G vertices. To ease the following discussion, let us denote by $\mathcal{T}_X$ the subtree of $\mathcal{T}$ rooted in X and by G_X the connected subgraph of G induced by the vertices contained in the nodes of $\mathcal{T}_X$. Given some vertex $u \in V$, we refer to the unique node X of $\mathcal{T}$ with $u \in X$ also by X_u. A node X of $\mathcal{T}$ is further said to be of *level* i, if the unique simple path from X to the root of $\mathcal{T}$ has length i. An illustration of a graph G and one of its $(1/2, \sqrt{n})$-balanced separator decompositions may be seen in Figure 3.6. It



(a) A graph $G = (V, E)$ with vertices numbered 1 through 11.



(b) A $(1/2, \sqrt{n})$-separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of the graph shown in Figure 3.6a. The nodes $X \in \mathcal{X}$ are drawn as grey circles containing the vertices $u \in X$. The shades of grey become increasingly lighter with ascending levels of the nodes $X \in \mathcal{X}$. The root of $\mathcal{T}$ is the node drawn at the top.

Figure 3.6: A graph $G = (V, E)$ and a $(1/2, \sqrt{n})$-balanced separator-decomposition of G.

may also be seen in this example that each edge $\{u, v\}$ of G, where X_u is of greater or equal level than X_v, "follows" the unique simple path from X_u to the root R of $\mathcal{T}$. It is this very observation that we will exploit in what follows below and therefore record in terms of the following remark.

3.6 Remark: Let $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ be a (b, f)-balanced separator decomposition of some undirected graph G and let $\{u, v\}$ be an edge of G, such that X_u is of greater or equal level than X_v. The node X_v is then an ancestor of X_u. †

Proof: Let R denote the root of $\mathcal{T}$. Consider the lowest common ancestor X of X_u and X_v in $\mathcal{T}$, that is, the node X of $\mathcal{T}$ of highest level that simultaneously lies on both the unique simple path from X_u to R and the unique simple path from X_v to R. As X is the node of highest level that lies on both these paths, it follows that either $X = X_v$ or that X_u and X_v lie in distinct subtrees of $\mathcal{T}_X$. Observe that X_u and X_v cannot possibly lie in distinct subtrees of $\mathcal{T}_X$, for u and v are connected by an edge and vertices contained in distinct subtrees of $\mathcal{T}_X$ belong to distinct connected components of $G_X \setminus X$ by the definition of a (b, f)-balanced separator decomposition. It therefore follows that $X = X_v$, which obviously implies our claim. □

Given a (b, f)-balanced separator decomposition $\mathcal{T}$ of G, we associate a *nested dissection order* $\alpha(\mathcal{T})$ on the vertices of G with $\mathcal{T}$. For this purpose, we choose an arbitrary bijection $\alpha_X \colon X \to \{n - |X| + 1, \ldots, n\}$ and let

$$\alpha(u) = \begin{cases} \alpha_X(u) & \text{if } u \in X \\ \alpha(\mathcal{T}_i)(u) + \sum_{j<i} n_j & \text{otherwise}, \end{cases} \tag{3.3}$$

where $\mathcal{T}_i$ denotes the subtree of $\mathcal{T}$ rooted in the i-th child X_i of X and n_i denotes the number of vertices contained in the nodes of $\mathcal{T}_i$. The order $\alpha(\mathcal{T})$ hence assigns the maximum values to the vertices contained in the root X of $\mathcal{T}$, maps the vertices contained in the subtrees $\mathcal{T}_i$ of $\mathcal{T}$ rooted in X_i into pairwise disjoint intervals

$$[1, \ldots, n_1], \quad [1 + n_1, \ldots, n_1 + n_2], \quad \ldots, \quad [1 + \sum_{j<d} n_j, \ldots, \sum_{j \le d} n_j]$$

and induces a nested dissection order $\alpha(\mathcal{T}_i)$ in each such interval. We denote by d the degree of X in $\mathcal{T}$. The reader should be aware that it is not mandatory but merely convenient to require a nested dissection order α to map the subtrees $\mathcal{T}_i$ into disjoint intervals. If one assumes that the children of a node X in our previous example from Figure 3.6 are sorted from left to right, the nested dissection order $\alpha(\mathcal{T})$ associated with the $(1/2, \sqrt{n})$-balanced separator decomposition $\mathcal{T}$ of this example may be seen in Figure 3.7. Nested dissection orders $\alpha = \alpha(\mathcal{T})$ relate to our earlier observation № 3.6 in the following manner. Any edge $\{u, v\}$ with $\alpha(u) < \alpha(v)$ certainly satisfies that X_u is of greater or equal level than X_v, for otherwise u and v would have to be vertices

i	1	2	3	4	5	6	7	8	9	10	11
$\alpha^{-1}(i)$	1	2	4	5	3	9	11	8	10	6	7
Root X of the corresponding subtree $\mathcal{T}_X$ of $\mathcal{T}$	$\{1,2\}$		$\{4,5\}$			$\{9\}$	$\{11\}$				
	$\{3\}$					$\{8,10\}$					
	$\{6,7\}$										

Figure 3.7: The nested dissection order $\alpha(\mathcal{T})$ associated with the $(1/2, \sqrt{n})$-balanced separator decomposition $\mathcal{T}$ from Figure 3.6.

contained in disjoint subtrees of $\mathcal{T}$, which is excluded by the existence of the edge $\{u, v\}$. Therefore, № 3.6 implies that X_u is not only of greater or equal level than X_v, but that X_v is an ancestor of X_u. The following remark merely spells out this discussion.

3.7 Remark: Let $G = (V, E)$ be an undirected graph and $\alpha = \alpha(\mathcal{T})$ be the nested dissection order associated with a given (b, f)-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of G. Then X_v is an ancestor of X_u for any edge $\{u, v\}$ of G with $\alpha(u) < \alpha(v)$. †

Proof: We may assume without loss of generality that $X_u \neq X_v$, for otherwise our claim is trivially satisfied. Let us assume for the sake of contradiction that X_v is of greater level than X_u. Since $\alpha(u) < \alpha(v)$ by assumption, our construction (3.3) of $\alpha = \alpha(\mathcal{T})$ then implies that X_u and X_v are nodes of disjoint subtrees of $\mathcal{T}$. This is in fact a contradiction since u and v are connected by an edge in G but ought to lie in different connected components of $G_X \setminus X$, where X denotes the lowest common ancestor of X_u and X_v in $\mathcal{T}$. We may thus draw the conclusion that X_u is of greater or equal level than X_v and our claim hence follows from № 3.6. □

Let us now investigate how these findings relate to the elimination game played on G in order $\alpha(\mathcal{T})$. If the removal of $v = \alpha^{-1}(1)$ during the first step of the elimination game played on G causes the insertion of a fill-edge $\{u, w\}$, we find that both X_u and X_w are ancestors of X_v by virtue of the above remark. If we additionally assume $\alpha(u) < \alpha(w)$, i.e. that uw is an arc of the filled graph G^α, we see that X_w is an ancestor of X_u. Altogether, the graph G^2 obtained from $G = G^1$ by this first step of the elimination game contains only edges, such that the node of $\mathcal{T}$ containing one endpoint is an ancestor of the node containing the other endpoint. It now follows by induction that subsequent steps of the elimination game can only lead to the insertion of fill-arcs xy in G^α, where X_y is an ancestor of X_x in $\mathcal{T}$. The following remark makes this discussion precise.

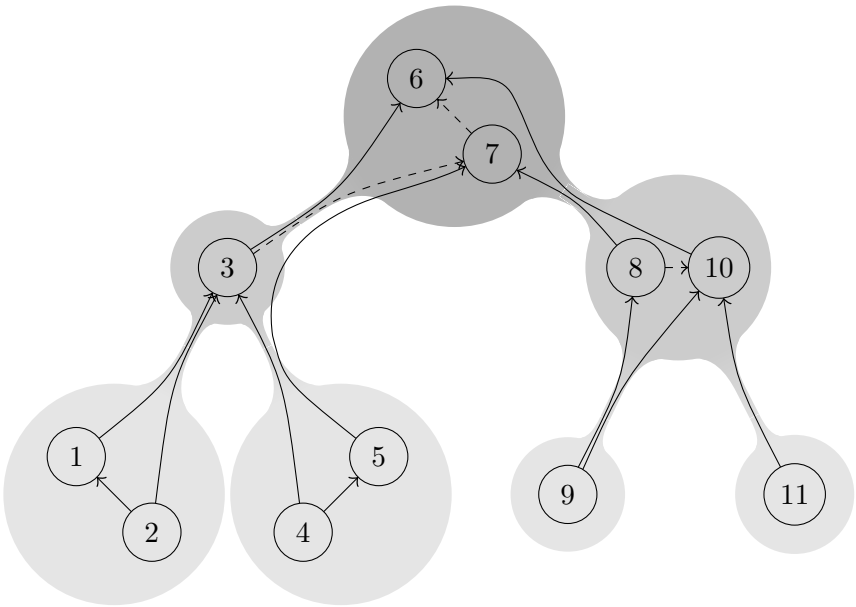


Figure 3.8: The filled graph G^α associated with the graph G shown in Figure 3.6a and the nested dissection order $\alpha(\mathcal{T})$ of the $(1/2, \sqrt{n})$-balanced separator decomposition from Figure 3.6. Fill arcs are drawn dashed.

3.8 Remark: Let $G = (V, E)$ be an undirected graph and $\alpha = \alpha(\mathcal{T})$ be the nested dissection order associated with a given (b, f)-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of G. Then X_v is an ancestor of X_u for any arc uv of the filled graph G^α. †

Proof: It suffices to show that X_v is an ancestor of X_u for each edge $\{u, v\}$ of G^i with $\alpha(u) < \alpha(v)$ and each step i of the elimination game. Observe that № 3.7 ensures precisely this property for $G = G^1$. We thus only need to show that G^{i+1} possesses the above property if G^i does, for our claim then follows by induction. To this end, consider the vertex $v_i = \alpha^{-1}(i)$ that is removed in the i-th step of the elimination game. Further let $\{u, w\}$ be a fill-edge with $\alpha(u) < \alpha(w)$ that is inserted during this step. There are then edges $\{v_i, u\}$ and $\{v_i, w\}$ in G^i. Our assumptions imply that both X_u and X_v are ancestors of X_{v_i} in $\mathcal{T}$ since $\alpha(v_i)$ is minimal among all the vertices of G^i. Note that this is possible only if either X_u is an ancestor of X_v or vice versa. However, our assumption $\alpha(u) < \alpha(v)$ and the construction (3.3) of $\alpha = \alpha(\mathcal{T})$ preclude the case that X_u is an ancestor of X_v. Altogether, we conclude that X_v is indeed an ancestor of X_u for any fill-edge $\{u, v\}$ with $\alpha(u) < \alpha(v)$ inserted during the i-th step of the elimination game. This obviously finishes the proof. □

Summarising our above discussion, we have just shown that the target vertex v of any arc uv of G^α is contained in a node X on the unique simple path from X_u to the root R

of $\mathcal{T}$. This can also be seen in Figure 3.8, where the filled graph G^α associated with the graph from Figure 3.6a and the nested dissection order $\alpha = \alpha(\mathcal{T})$ from Figure 3.6 and Figure 3.7 is depicted. An easy induction over the number of arcs of a path in G^α now gives the following corollary.

3.9 Corollary: Let $G = (V, E)$ be an undirected graph and $\alpha = \alpha(\mathcal{T})$ be the nested dissection order associated with a given (b, f)-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of G. Furthermore, let $u \in V$ be any vertex of G. The search space $\mathcal{S}(u, G^\alpha)$ of u in G^α is a subset of the set of vertices contained in all the nodes lying on the unique simple path from X_u to the root R of $\mathcal{T}$. In particular, this implies that $\#\mathcal{S}(u, G^\alpha)$ is less than or equal to the number of those vertices. †

As the elimination tree $T(G^\alpha)$ is a subgraph of G^α this same assertion holds true in $T(G^\alpha)$, too. We thus have the following corollary.

3.10 Corollary: Let $G = (V, E)$ be an undirected graph and $\alpha = \alpha(\mathcal{T})$ be the nested dissection order associated with a given (b, f)-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of G. Further let $u \in V$ be any vertex of G. The length of the unique simple path from u to the root r of the elimination tree $T(G^\alpha)$ is less than or equal to the number of vertices contained in all the nodes lying on the unique simple path from X_u to the root R of $\mathcal{T}$. †

Comparing the above two corollaries, we feel that the formulation in terms of search spaces and their size better reflects the actual reasoning we employed in № 3.6, № 3.7 and № 3.8 to draw these conclusions. In fact, our machinery of search spaces and their size renders the usual indirection over elimination trees and their height essentially superfluous and allows us to state the very same content naturally in terms of the filled graph G^α. However, we ought to stop philosophising and rather explore the consequences of the above corollaries concerning search space sizes in contraction hierarchies. To this end, the following theorem is an immediate consequence of our previous results.

Theorem 4 *Contraction hierarchies with respect to nested dissection orders*
Let $G = (V, E)$ be a weighted digraph and $\alpha = \alpha(\mathcal{T})$ be the nested dissection order associated with a given (b, f)-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of ${}^*G$. The search space $\mathcal{S}(u, H_\alpha^\wedge)$ as well as the reverse search space $\mathcal{R}(u, H_\alpha^\vee)$ of any vertex u in any weak contraction hierarchy H_α of G is a subset of the set of vertices contained in all the nodes lying on the unique simple path from X_u to the root R of $\mathcal{T}$. This implies in particular, that $\#\mathcal{S}(u, H_\alpha^\wedge)$ and $\#\mathcal{R}(u, H_\alpha^\vee)$ are bounded from above by the number of those vertices. ‡

In order to finally obtain an upper bound on $\mathcal{S}_{\max}(u, H_\alpha)$, we only need to bound the number of vertices contained in the nodes on the simple path from X_u to the root R

of $\mathcal{T}$. This issue can not be handled simultaneously for all families of graphs, but needs special treatment depending on the properties of G. In the remainder of this paragraph, we give a rather general such treatment, but devote Chapter 3.3 to the investigation of the issue at hand for some particular family of graphs.

Recall that a (b, f)-balanced separator decomposition $\mathcal{T}$ of a graph G with n vertices is recursively defined as a tree with root $X \subseteq V$ of size at most $f(n)$, such that $G \setminus X$ consists of at least two connected components $G_1, \ldots, G_d$ with at most bn vertices each. The children X_i of X in $\mathcal{T}$ are then the roots of (b, f)-balanced separator decompositions of $G_1, \ldots, G_d$. As we required f to be increasing and as $0 < b < 1$, it follows that the children X_i of X in $\mathcal{T}$ have size at most $f(bn) \leq f(n)$. Iterating this argument, we find that all the internal nodes X of $\mathcal{T}$ have size at most $f(n)$, while the leaves have size n_0 for some constant $n_0 \geq 1$. The number of vertices contained in all the nodes on a path from a leaf of $\mathcal{T}$ to its root is therefore bounded from above by

$$n_0 + \sum_{i=1}^{h} f(n) = n_0 + h \cdot f(n)\,, \tag{3.4}$$

where h denotes the height of $\mathcal{T}$. However, the height h of $\mathcal{T}$ may also bounded in terms of the following remark.

3.11 Remark: The height of a (b, f)-balanced separator decomposition $\mathcal{T}$ of G is bounded from above by $\log_a n$, where n denotes the number of vertices of G and where $a = 1/b$. †

Proof: The proof is by induction on n.

$n \leq n_0$: If $n \leq n_0$, then $\mathcal{T}$ is a tree of height 0 and $\log_a n \geq 0$ since $n \geq 1$. In this case, our claim is thus satisfied.

$n > n_0$: If $n > n_0$, the height h of $\mathcal{T}$ is given by $h = 1 + \max_i h_i$, where h_i denotes the height of the subtree $\mathcal{T}_i$ rooted at the i-th child X_i of X. Recall that $\mathcal{T}_i$ contains at most $bn < n$ vertices. Our induction hypothesis hence implies $h_i \leq \log_a(bn)$. Note that the right hand side of the latter inequality equals $\log_a(n) - 1$ since $a = 1/b$. This gives the desired result. □

Combining the above remark with Theorem 4 and (3.4), we obtain the following corollary.

3.12 Corollary: Let G be a weighted digraph and denote by ${}^*G$ the undirected graph associated with G. Then

$$\mathcal{S}_{\max}(H_\alpha) \leq n_0 + f(n) \cdot \log_{1/b}(n)$$

for any weak contraction hierarchy H_α of G and any nested dissection order α associated with a (b, f)-balanced separator decomposition $\mathcal{T}$ of ${}^*G$. †

Albeit the above corollary finally gives the desired upper bound on $\mathcal{S}_{\max}(H_\alpha)$, an open question that we did not touch upon at all still remains to be answered: Are there any suitable (b, f)-balanced separator decompositions of a given graph G and how can they be computed in case they exist? We will answer this very question for a particularly interesting case in Chapter 3.3. However, in this paragraph, we committed ourselves to the most general phrasing of our results and deemed the notions of *treewidth* and *tree decomposition* appropriate for a sufficiently general yet meaningful summary of our findings so far. Essentially, we only make use of the fact that a $(1/2, k+1)$-balanced separator decomposition is computable from a suitable tree decomposition of width at most k in time $\mathcal{O}(n \cdot \log(n))$. The constant n_0 that we kept throughout this whole paragraph turns out to equal $k+1$ in this particular case. We remark that we require the tree decomposition of G to be "suitable" only to exclude too large tree decompositions; This is no serious restriction though. The reader may find all the relevant definitions in the article of Bodlaender et al. [BGKH91] or in the book of Diestel [Die06]. Given the definition of a tree decomposition and its width, the well-known algorithm to compute $(1/2, 1)$-balanced separator decompositions in trees facilitates the computation of $(1/2, k+1)$-balanced separator decomposition from a given tree decomposition of width k. Let us take these notions and the said result for granted. We then obtain the following theorem.

Theorem 5 *Contraction hierarchies of graphs with bounded treewidth*
For any weighted digraph G of treewidth at most k, there exists an order α on the vertices of G, such that

$$\mathcal{S}_{\max}(H_\alpha) \leq (k+1) \cdot (1 + \log(n))$$

for any weak contraction hierarchy H_α of G. The number of arcs in H_α is thereby less than or equal to $n \cdot (k+1) \cdot (1 + \log(n))$.

Moreover, given a suitable tree decomposition of width at most k, there are algorithms that compute this order α in time $\mathcal{O}(n \log(n))$, the corresponding maximal weak contraction hierarchy M_α in time $\mathcal{O}(kn \log(n))$ and the contraction hierarchy G_α in time $\mathcal{O}(kn \log(n) \cdot \mathsf{T}_{\text{query}})$. ‡

Proof: The order α is of course the nested dissection order $\alpha(\mathcal{T})$ associated with the $(1/2, k+1)$-balanced separator decomposition $\mathcal{T}$ that one obtains from any tree decomposition of G of width at most k. The upper bound on $\mathcal{S}_{\max}(H_\alpha)$ then follows from № 3.12. Moreover, this bound also applies to the maximal weak contraction hierarchy M_α. This implies in particular that the *average* search space size in M_α is bounded by the same term as is the maximum search space size $\mathcal{S}_{\max}(M_\alpha)$. The claimed bound on the number of arcs in any weak contraction hierarchy $H_\alpha \subseteq M_\alpha$ now follows from № 2.17.

Since $\mathcal{T}$ can be computed from a suitable tree decomposition in time $\mathcal{O}(n \log(n))$ and since the nested dissection order $\alpha(\mathcal{T})$ can then be constructed in linear time,

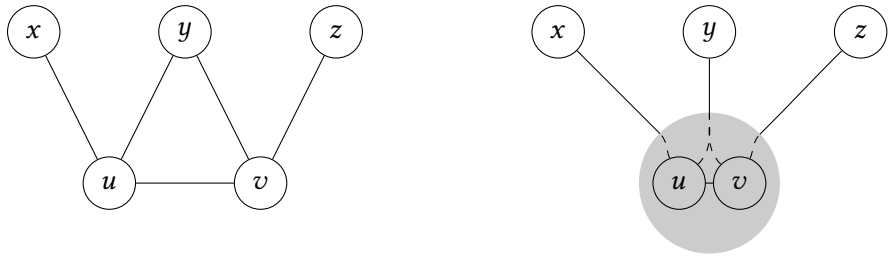
the claimed running time for the computation of α follows. As the number of arcs in M_α is of the order of $\mathcal{O}(nk\log(n))$, it follows that the algorithms from № 2.15 to compute M_α and from № 2.16 to compute G_α in fact achieve the claimed running times. ▫

The simplest class of graphs to which the above theorem applies are trees having constant treewidth one. For them, Theorem 5 states that the maximum search space size in a contraction hierarchy with respect to a nested dissection order is at most $\mathcal{O}(\log(n))$ while its space consumption is of the order of $\mathcal{O}(n\log(n))$. It is also the class of trees, where explicit examples for an exponential gap between the minimum elimination tree height and the height of an elimination tree associated with a nested dissection order are known. More precisely, Heggernes constructs in her master's thesis [Heg92] a family of trees containing $n = 2^k$ vertices that have minimum elimination tree height $\mathrm{ht}(G) \leq \log(k)$ but for which there is a nested dissection order resulting in an elimination tree of height k. Note that this does not contradict the approximation results of Bodlaender et al. [BGKH91] as the gap between the minimum elimination tree height and the height of the elimination tree with respect to the nested dissection order is still of the order of $\mathcal{O}(\log(n)) = \mathcal{O}(k)$. However, in the case that one is really concerned with contraction hierarchies of trees, there are superior methods. Algorithms due to Schäffer [Sch89], Iyer, Ratliff and Vijayan [IRV88] and Hsu, Peng and Shi [HPS07] compute orders α on the vertices of an undirected tree, such that the corresponding elimination tree is of minimum height. Moreover, utilising the fact that paths in trees are unique, it is easy to see that the height $\mathrm{ht}(T^\alpha)$ of an elimination tree and the maximum search space size $\mathcal{S}_{\max}(T_\alpha)$ in the contraction hierarchy T_α of T coincide indeed. Thus, said algorithms also solve the problem to determine optimal contraction hierarchies of trees. Despite the fact that nested dissection orders are not suitable for an approximation of the optimal maximum search space size, an easy argument proves nested dissection to yield a 2-approximation to the minimum average search space size in contraction hierarchies of trees – a problem that has not been previously studied.

3.3 Nested Dissection in Graphs with $\mathcal{O}(\sqrt{n})$-Separators

In this paragraph we study the specifics of nested dissection in graphs admitting balanced $\mathcal{O}(\sqrt{n})$-separators. In this particular situation, we can improve on our results from Chapter 3.2 in that we obtain smaller upper bounds on both the number of arcs in G_α and the maximum search space size $\mathcal{S}_{\max}(G_\alpha)$ for nested dissection orders α.

In order to keep our exposition as self-contained as possible, we briefly recall the notion of edge contraction and that of a minor of a graph. For this purpose, fix an undirected graph $G = (V, E)$ and let $\{u, v\} \in E$ be an edge of G. By the *contraction of* $\{u, v\}$ we mean the removal of one of its endpoints, say v, from G and the subsequent insertion of edges $\{u, w\}$ for each neighbour w of the removed endpoint v. The reader



(a) An undirected graph $G = (V, E)$.

(b) The undirected graph G shown to the left after contraction of the edge $\{u, v\}$.

Figure 3.9: Contraction of an edge $\{u, v\}$. The graph obtained from contraction of the edge $\{u, v\}$ of the graph shown in Figure 3.9a may be seen in Figure 3.9b. The vertex corresponding to $\{u, v\}$ is drawn as grey circle containing both u and v.

might also envision the contraction of an edge $\{u, v\}$ as the process of continuously shrinking the edge $\{u, v\}$ until both endpoints u and v meet and thus become a single vertex of G. Formally, the graph $H = (V', E')$ obtained from G by contraction of $\{u, v\}$ is given by $V' = V \setminus \{v\}$ and

$$E' = \left(E \setminus \left\{ \{x, y\} \;\middle|\; v \in \{x, y\} \right\}\right) \cup \left\{ \{u, w\} \;\middle|\; \{v, w\} \in E \text{ and } w \neq u \right\}.$$

An example of an edge contraction may also be seen in Figure 3.9. Given the notion of edge contractions, it is now straightforward to define what is meant by a minor of some graph G: A graph H is said to be a *minor* of some graph G, if H may be obtained from an arbitrary subgraph of G by successive contraction of edges. For the remainder of this paragraph, we fix a class $\mathcal{C}$ of graphs that is closed under taking minors and of which every graph on at least n_0 vertices admits a $(b, a\sqrt{n})$-balanced separator. That is, any minor H of some $G \in \mathcal{C}$ is also contained in $\mathcal{C}$ and each $G \in \mathcal{C}$ with $n \geq n_0$ vertices contains a set S of vertices of size at most $a\sqrt{n}$ whose removal separates G into connected components containing at most bn vertices each. Recall from [Mad67] that any such class $\mathcal{C}$ of graphs is sparse in the sense that there is a constant $\delta > 0$, such that the number of edges of a graph $G \in \mathcal{C}$ with n vertices is at most δn. By the infamous planar separator theorem of Lipton and Tarjan [LT79] and its generalisation due to Gilbert, Hutchinson and Tarjan [GHT84], both the family of planar graphs and the family of graphs embeddable on a surface of fixed genus g are examples of such classes $\mathcal{C}$. Moreover, there are algorithms to compute such balanced separators in linear time [LT79; AD96]. Inspired by these results, Reed and Wood and Kawarabayashi and Reed among others have studied the existence of balanced $\mathcal{O}(\sqrt{n})$-separators in classes of graphs that exclude a fixed graph as minor. For these families of graphs, there are $\mathcal{O}(n^{2/3})$-separators, which can be computed in linear time [RW09] and $\mathcal{O}(\sqrt{n})$-separators that can be computed in time $\mathcal{O}(n^{1+\varepsilon})$ for any constant $\varepsilon > 0$ [KR10]. Perhaps a little less prominent but certainly worth

mentioning in the context of our work is the model of road networks due to Eppstein and Goodrich [EGo08b; EGo08a] that also features the existence of $\mathcal{O}(\sqrt{n})$-separators and an algorithm to compute these separators in linear time.

We remark that $(b, a\sqrt{n})$-balanced separator decompositions were already studied by Gilbert and Tarjan and we refer the reader to their article [GHT84] for some of the omitted proofs. Nevertheless, we want to give a proof of at least the most crucial result concerning $(b, a\sqrt{n})$-balanced separator decompositions $\mathcal{T}$ of graphs $G \in \mathcal{C}$. More precisely, we want to show that we are able to bound the number of vertices contained in the nodes of a simple path in $\mathcal{T}$ from one of its leaves to the root by $n_0 + \frac{a}{1-\sqrt{b}}\sqrt{n}$, where n_0 denotes the number of vertices contained in a leaf of $\mathcal{T}$. This obviously improves on the bound of $\mathcal{O}(f(n) \cdot \log(n))$ from Chapter 3.2 for the general case. More precisely, we have the following remark.

3.13 Remark: Let $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ be a $(b, a\sqrt{n})$-balanced separator decomposition of a graph $G = (V, E)$. The number of vertices contained in the nodes of a simple path from a leaf of $\mathcal{T}$ to its root, is at most

$$n_0 + \frac{a}{1-\sqrt{b}}\sqrt{n}\,. \qquad \dagger$$

Proof: Recall that a node X of $\mathcal{T}$ is said to be of level i if the simple path from X to the root R of $\mathcal{T}$ is of length i. Observe that a node X of level i contains at most $a\sqrt{b^i n}$ vertices. Therefore, the number of vertices contained in the nodes of a simple path from the root of $\mathcal{T}$ to one of its leaves is certainly less than

$$n_0 + \sum_{i=0}^{\infty} a\sqrt{b^i n}\,.$$

This infinite series converges from below to

$$n_0 + \frac{a}{1-\sqrt{b}}\sqrt{n}\,,$$

which is the claimed bound. □

We thus obtain an upper bound of $\mathcal{O}(\sqrt{n})$ on $\mathcal{S}_{\max}(H_\alpha)$ for a nested dissection order α and all weak contraction hierarchies H_α of a graph G belonging to the class $\mathcal{C}$. According to № 2.17, this also implies an upper bound of $\mathcal{O}(n^{3/2})$ on the number of arcs in G_α. However, for classes $\mathcal{C}$ as studied in this paragraph, a clever analysis due to Gilbert and Tarjan [GT86] shows that there are at most $\mathcal{O}(n \log(n))$ arcs in the filled graph G^α. Hence, the number of arcs in any weak contraction hierarchy H_α is also bounded from above by $\mathcal{O}(n \log(n))$. Altogether, we obtain the following theorem.

Theorem 6 *Contraction hierarchies of graphs with $\mathcal{O}(\sqrt{n})$-separators*
For any minor-closed class $\mathcal{C}$ of graphs, such that any $G \in \mathcal{C}$ with at least n_0 vertices admits a $(b, a\sqrt{n})$-balanced separator, there exists an order α on the vertices of G, such that

$$\mathcal{S}_{\max}(H_\alpha) \leq n_0 + \frac{a}{1-\sqrt{b}}\sqrt{n}$$

for any weak contraction hierarchy H_α of G. The number of arcs in H_α is thereby of the order of $\mathcal{O}(n \log(n))$.

Moreover, given a $(b, \sqrt{n})$-balanced separator decomposition of a graph $G \in \mathcal{C}$, this specific order α, the corresponding maximal weak contraction hierarchy M_α and the contraction hierarchy G_α may be constructed in time $\mathcal{O}(n \log(n))$, $\mathcal{O}(n \log(n))$ and $\mathcal{O}(n \log(n) \cdot \mathsf{T}_{\text{query}})$, respectively. ‡

The above theorem is not as strong as one might wish. Although we have bounded the maximum search space size $\mathcal{S}_{\max}(H_\alpha)$, a distance query in H_α might still require time $\mathsf{T}_{\text{query}} = \Theta(n)$. This stems from the fact, that we are not able to bound the number of arcs occurring in a particular search space $\mathcal{S}(x, H_\alpha)$ and therefore have to act on the assumption that a single search space may contain as many as $\Theta(n)$ arcs. Obtaining sublinear bounds on the number of arcs contained in the search space of a single vertex turns out to be a rather difficult problem. Admittedly, all our attempts failed miserably. Nevertheless, this question is definitely worth considering, because an upper bound on the number of arcs or a slight improvement on the query algorithm would possibly render contraction hierarchies comparable to the data structures by Djidjev [Dji96] or even Sommer [Som12] for answering distance queries in static graphs.

3.4 Nested Dissection and Highway Dimension

In this paragraph we try to compare our results of Chapter 3.1 to the results of Abraham et al. [AFGW10; ADFGW11]. They propose a new graph parameter, called *highway dimension* intended to capture the intuition that all sufficiently long shortest paths in road networks necessarily pass at least one vertex out of a set of very few important access nodes, like e.g. motorway junctions and slip roads. Utilising this new parameter, they deduce upper bounds for the space consumption and query time of several speedup techniques for Dijkstra's algorithm – including contraction hierarchies. Our approach differs from their work in that the edge lengths and shortest paths play a crucial role in both the definition of highway dimension and in the resulting algorithms, while we obtained Theorem 4 by hiding the edge lengths and shortest path structure to arrive at a conceptually simpler problem. As a consequence, Abraham et al. obtain smaller upper bounds on both the space consumption of contraction hierarchies and the query time $\mathsf{T}_{\text{query}}$ if the edge lengths are sufficiently well-behaved. However, our main result in this paragraph states that our approach leads to strikingly similar

upper bounds on both the space consumption of contraction hierarchies and the running time of distance queries if the edge lengths are sufficiently ill-behaved. As highway dimension has been defined only recently, near to nothing is known about its relation to other graph parameters. We contribute to the understanding of highway dimension in the sense that our results imply the existence of edge lengths on any graph G, such that its highway dimension with respect to these lengths is of the order of $\Omega(\mathrm{pw}(G)/\log(n))$, where $\mathrm{pw}(G)$ denotes the pathwidth of G.

Let us briefly recall the definition of highway dimension from [ADFGW11]. For this purpose, let us fix a weighted, undirected graph $G = (V, E)$. Given a vertex $u \in V$, the set of vertices v of distance at most ε from u is called the *ball of radius* ε around u or simply the ε-ball. We denote the ε-ball around u by $B_\varepsilon(u)$. We say that a vertex u *covers* a shortest path p if p contains u. The *highway dimension* $\mathrm{hd}(G)$ of G is the least integer d, such that all the shortest paths p in G of length $\varepsilon < \mathrm{len}(p) \leq 2\varepsilon$ that intersect a given ball of radius 2ε can be covered by at most d vertices. This definition does not generalise to digraphs as it stands and we restrict ourselves to undirected graphs only, as our arguments below rely crucially on this very characterisation of highway dimension. Abraham et al. devise a polynomial-time approximation of $\mathrm{hd}(G)$ with approximation ratio of $\log(\mathrm{hd}(G))$ that is furthermore capable of computing sets of size $\log(\mathrm{hd}(G)) \cdot \mathrm{hd}(G)$ covering all shortest paths of length between ε and 2ε that intersect a given 2ε-ball. Moreover, they describe specific orders α, for which they are able to prove the following theorem.

Theorem (Abraham et al. [ADFGW11]):
Let G be a weighted graph of highway dimension d, diameter D and let Δ denote the maximum degree in G.

(a) There exists an order α on the vertices of G, such that the number of arcs in the contraction hierarchy G_α is of the order of $\mathcal{O}(nd\log(D))$ and such that distance queries in G_α can be answered in time $\mathcal{O}((\Delta + d\log(D)) \cdot d\log(D))$.

(b) There is a polynomial-time algorithm that computes an order α on the vertices of G, such that the number of arcs in G_α is of the order of $\mathcal{O}(nd\log(n)\log(D))$ and such that distance queries in G_α may be answered with running time of the order of $\mathcal{O}((\Delta + d\log(d)\log(D) \cdot d\log(d)\log(D))$. ‡

The reader should be aware that the diameter D referred to in the above theorem is the diameter with respect to integral edge lengths greater or equal to one.

In the remainder of this paragraph, we proceed as follows. We make the most pessimistic assumptions on the edge lengths in G; That is, we consider edge lengths on G, such that $\mathrm{hd}(G)$ with respect to these lengths actually achieves the maximum possible highway dimension k over all edge lengths on G. For this particular choice of lengths, we construct a $(2/3, k)$-balanced separator decomposition of G, which then provides the link to nested dissection orders and Theorem 4.

3.14 Lemma: Let $G = (V, E)$ be a connected undirected graph and denote by k the maximum highway dimension over all possible edge lengths on G. Any connected subgraph H of G containing at least $2k + 2$ vertices may then be separated by the removal of at most k vertices into at least two connected components, each of them containing at most $\lceil n_H/2 \rceil$ vertices. We thereby denote by n_H the number of vertices of H. †

Proof: The idea of this proof is the following. We construct edge lengths on G, such that all paths connecting vertices of an appropriately sized subgraph H_1 of H to vertices in the remainder of G are shortest paths of length equal to one. We then assign "infinitesimal" lengths to all the other edges, so that there is a ball containing the whole of H_1 of radius $3/2$. By our assumptions, we may then choose a set S of at most k vertices, such that each edge having one endpoint in H_1 and the other in the remainder of G is covered by S. It turns out that S is in fact the sought separator of H.

Denote the set of vertices of H by V_H and its cardinality by n_H. Choose a connected subgraph H_1 of H containing $\lfloor n_H/2 \rfloor$ vertices and let V_1 and n_1 denote the set of vertices of H_1 and its cardinality, respectively. We define edge lengths $\mathrm{len}\colon E \to \mathbb{R}^+$ on G as follows. Any edge with one endpoint in V_1 and the other endpoint in $V \setminus V_1$ is assigned a length of 1 and any edge with both endpoints in either V_1 or $V \setminus V_1$ is assigned a length of $\varepsilon = 3/2 \cdot n_H$. Observe that any simple path that lies completely inside H_1 has length at most $n_H \cdot \varepsilon \leq 3/2$. Let us consider a ball B of radius $3/2$ around any vertex u of H_1. Observe that B contains the whole of V_1 by the above computation. According to our choice of k, the highway dimension $\mathrm{hd}(G)$ of G is at most k. We may thus choose a set S of at most k vertices, such that each shortest path in G that intersects H_1 and has length between $3/4$ and $3/2$ contains at least one element of S. We claim that removal of $S \cap V_H$ separates H into at least two connected components with at most $\lceil n_H/2 \rceil$ vertices each. To this end, let us first reassure that removal of S separates H_1 from $H \setminus V_1$. Consider any path with source $s \in V_1$ and target $t \in V \setminus V_1$. This path necessarily contains an edge $\{u, v\}$ with $u \in V_1$ and $v \notin V_1$. However, by the construction of the edge lengths in G, this edge is a shortest path of length 1 and one of its endpoints is therefore contained in S. It remains to verify that the connected components of $H \setminus S$ contain at most $\lceil n_H/2 \rceil$ vertices each and that there are at least two such components. However, the former claim follows immediately from our choice of H_1. In order to settle also the latter claim, note that our assumption $n_H \geq 2k + 2$ implies

$$\begin{aligned} |V_1 \setminus S| &\geq |V_1| - k & \left|V_H \setminus (V_1 \cup S)\right| &\geq |V_H \setminus V_1| - k \\ &\geq k + 1 - k = 1 & &\geq k + 1 - k = 1\,, \end{aligned}$$

which finishes the proof. □

We want to point out that the above argument can easily be adapted to the slightly different definition of highway dimension in [AFGW10]. Although the proof of № 3.14 can hardly be called constructive, we still obtain the following corollary.

3.15 Corollary: Let $G = (V, E)$ be a connected undirected graph and denote by k the maximum highway dimension over all possible edge lengths in G. Then G admits a $(2/3, k)$-balanced separator decomposition $\mathcal{T}$ whose leaves contain at most $2k + 1$ vertices. Moreover, there exists an algorithm with polynomial time complexity that computes a $(2/3, \mathcal{O}(k \log(k)))$-balanced separator decomposition $\mathcal{T}$ of G, such that the leaves of $\mathcal{T}$ contain at most $\mathcal{O}(k \log(k))$ vertices. †

Proof: According to № 3.14, each subgraph consisting of of $n' \geq 2k + 2$ vertices may be separated into at least two connected components with at most $\lceil n'/2 \rceil$ vertices each by the removal of at most k vertices. As $\lceil n'/2 \rceil \leq \frac{2}{3}n'$, we may construct a $(2/3, k)$-balanced separator decomposition as follows: Beginning with G, we choose in each step a separator of size at most k in each remaining connected component of G, make it the root of a $(2/3, k)$-balanced separator decomposition of that specific component and recursively proceed on the connected components after removal of that separator. This obviously gives the desired $(2/3, k)$-balanced separator decomposition $\mathcal{T}$ of G. The leaves of $\mathcal{T}$ thereby have size at most $2k + 1$, for otherwise we could separate the corresponding connected component of G once more.

Employing the earlier-mentioned $\log(k)$-approximation of the minimum set of vertices covering all sufficiently long shortest paths instead of choosing an optimal cover of size k, one obviously obtains the claimed polynomial-time algorithm to compute a $(2/3, \mathcal{O}(k \log(k)))$-balanced separator decomposition of G whose leaves contain at most $\mathcal{O}(k \log(k))$ vertices. ▫

Note that the above corollary together with № 3.10 implies that the minimum elimination tree height $\mathrm{ht}(G)$ of G has an upper bound in terms of the worst-case highway dimension k. More precisely, applying № 3.10 to the $(2/3, k)$-balanced separator decomposition of № 3.15 implies an upper bound of

$$\mathrm{ht}(G) \leq 2k + 1 + k\frac{\log(n)}{\log(3) - 1}$$

on the minimum elimination tree height $\mathrm{ht}(G)$ of G. Furthermore, as it is well known that the pathwidth $\mathrm{pw}(G)$ of a graph G does not exceed its minimum elimination tree height $\mathrm{ht}(G)$, we have just proven the below theorem. We refer the reader once again to the article of Bodlaender et al. [BGKH91] for the relevant definitions and a proof of the inequality $\mathrm{pw}(G) \leq \mathrm{ht}(G)$.

Theorem 7 *Highway dimension and pathwidth*
Let G be a weighted undirected graph. There exist edge lengths on G, such that

$$\mathrm{hd}(G) \geq \frac{(\log(3) - 1)(\mathrm{pw}(G) - 1)}{\log(n) + 2\log(3) - 2} > \frac{1}{2} \cdot \frac{\mathrm{pw}(G) - 1}{\log(n) + 1}\,. \qquad ‡$$

The above theorem is to our knowledge a novel and unanticipated relation between highway dimension and more commonly used width parameters.

Moreover, № 3.15 may also be utilised to compare the performance of contraction hierarchies computed with nested dissection orders to those computed on the basis of highway dimension. More precisely, employing the estimates of № 3.15 in Theorem 4 and № 3.12, we obtain the following theorem.

Theorem 8 *Nested dissection and highway-dimension based preprocessing*
Let $G = (V, E)$ be an undirected graph of diameter D and with maximum degree Δ. Further denote by β the order whose existence is guaranteed by the theorem of Abraham et al. and by δ the order as computed by the polynomial-time preprocessing algorithm of Abraham et al. There exist edge lengths on G and a nested dissection order α, such that

(a) the worst-case space consumption of G_α is at most a factor of $\mathcal{O}(\log(n)/\log(D))$ greater than that of G_β.

(b) the worst-case running time of distance queries in G_α is at most a factor of $\mathcal{O}(\log^2(n)/\log^2(D))$ greater than that in G_β.

Moreover, there is a polynomial-time algorithm that computes a nested dissection order γ, such that

(c) the worst-case space consumption of G_γ is at most a factor of $\mathcal{O}(\log(n)/\log(D))$ greater than that of G_δ.

(d) the worst-case running time of distance queries in G_γ is at most a factor of $\mathcal{O}(\log^2(n)/\log^2(D))$ greater than that in G_δ. ‡

Proof: We choose edge lengths on G such that the maximum highway dimension k of G is actually assumed. Recall from above that the optimal order β of the theorem of Abraham et al. [ADFGW11] results in a contraction hierarchy G_β that has

$$m_\beta = \mathcal{O}\Big(nk \log(D) \Big)$$

arcs and on which a distance query has worst-case running time

$$\mathsf{T}^\beta_{\text{query}} = \mathcal{O}\Big((\Delta + k\log(D)) \cdot k\log(D) \Big) .$$

By virtue of Theorem 4 and № 3.12, the nested dissection order α associated with the $(2/3, k)$-balanced separator decomposition from № 3.15 results in a contraction hierarchy G_α, such that

$$\mathcal{S}_{\max}(G_\alpha) = \mathcal{O}(2k + 1 + k\log(n)) .$$

By № 2.17, this implies that G_α has at most

$$m_\alpha = \mathcal{O}\Big(n \cdot (2k + 1 + k\log(n)) \Big) = \mathcal{O}\left(kn\log(n) \right)$$

arcs, which is a factor of $\mathcal{O}(\log(n)/\log(D))$ worse than the upper bound on m_β. Moreover, we may estimate the worst-case query time in G_α by simply assuming that Dijsktra's algorithm relaxes $\mathcal{O}(\mathcal{S}_{\max}(G_\alpha)^2)$ arcs. This rather pessimistic assumption then results in an upper bound of

$$\begin{aligned}\mathsf{T}^\alpha_{\text{query}} &= \mathcal{O}\left(\mathcal{S}_{\max}(G_\alpha) \cdot \log\left(\mathcal{S}_{\max}(G_\alpha)\right) + \mathcal{S}_{\max}(G_\alpha)^2\right) \\ &= \mathcal{O}\left(k\log(n) \cdot \log\left(k\log(n)\right) + k^2\log^2(n)\right)\end{aligned}$$

on the running time of a distance query. This is a factor of at most

$$\mathcal{O}\left(\frac{k\log(n) \cdot \log\left(k\log(n)\right) + k^2\log^2(n)}{(\Delta + k\log(D)) \cdot k\log(D)}\right) = \mathcal{O}\left(\frac{k^2\log^2(n)}{k^2\log^2(D)}\right) = \mathcal{O}\left(\frac{\log^2(n)}{\log^2(D)}\right)$$

worse than the upper bound on $\mathsf{T}^\beta_{\text{query}}$. The proof of the second part of the statement is completely analogous. The only difference is that we employ the polynomial-time computable $(2/3, \mathcal{O}(k\log(k)))$-balanced separator decomposition instead of the optimal one we used above. □

4 Local Modification of Contraction Hierarchies

The contraction hierarchy G_α of some fixed weighted digraph G depends crucially on the order α on V. In this chapter we are concerned with the question of how modifications to α influence the structure of a possibly weak contraction hierarchy G_α. However, our theory does not allow for arbitrary rearrangements of the order. Instead, we have to restrict ourselves to the case where the relative order of the endpoints s and t of certain well-behaved arcs st is interchanged. These limitations are mostly due to technical reasons and a prospective improvement on our formalism might provide deeper insight into the impact of modifications of α on G_α.

The chapter is divided into three parts, of which the first introduces *constitutive pairs*, which are used to classify all orders β for which G_α and G_β are equal, that is, to give a complete description of the arguably most simple modifications of α. The second part of this chapter is concerned with *tame arcs* – our notion of well-behaved arcs – and the last part is concerned with the question of how the rearrangement of the relative order of the endpoints of such a tame arc affects the contraction hierarchy G_α in general. These investigations may serve as a solid foundation for postprocessing algorithms for contraction hierarchies.

4.1 Constitutive Pairs

The aim of this paragraph is to gain a basic understanding of modifications to a given order α. The conceivably simplest modification to α is one that does not change G_α at all, and we initially focus on precisely these alterations of α. Let β denote the outcome of such a modification to α, i.e. let β be an order on V, such that $G_\alpha = G_\beta$. It is clear that if uv is an arc of $G_\alpha^\wedge$, then uv is also an arc of $G_\beta^\wedge$. As $\beta(u) < \beta(v)$ for any arc uv of $G_\beta^\wedge$, we see that $\beta(u) < \beta(v)$ for all arcs uv of $G_\alpha^\wedge$. More generally, the same argument applies to any pair of vertices u and v, where one is contained in the search space of the other. If for example $v \in \mathcal{S}(u, G_\alpha^\wedge)$, then $v \in \mathcal{S}(u, G_\beta^\wedge)$ and hence $\beta(u) \leq \beta(v)$. Similar observations apply of course also to $G_\alpha^\vee$ and $G_\beta^\vee$. After introducing some convenient terminology and establishing some technical prerequisites, we will actually show that these conditions are not only necessary but already sufficient for G_α and G_β to be equal.

Inspired by our above discussion, let us call a pair u and v of vertices *constitutive of* G_α if one vertex is contained in the search space of the other. More precisely, we say that u and v are constitutive of G_α, if at least one of the four conditions

$$u \in \mathcal{S}(v, G_\alpha^\wedge) \qquad u \in \mathcal{S}(v, G_\alpha^\vee)$$
$$v \in \mathcal{S}(u, G_\alpha^\wedge) \qquad v \in \mathcal{S}(u, G_\alpha^\vee)$$

is satisfied. Let us further say that two orders α and β *induce the same relative order* on two vertices u and v, if either both $\alpha(u) < \alpha(v)$ and $\beta(u) < \beta(v)$ or if both $\alpha(u) > \alpha(v)$ and $\beta(u) > \beta(v)$. With this terminology at hand, our discussion at the beginning of this paragraph may be summarised by saying that if $G_\alpha = G_\beta$, then α and β necessarily induce the same relative order on any pair of vertices that is constitutive of G_α. We also indicated that these conditions are already sufficient for G_α and G_β to coincide. Prior to proving this claim, we need to establish some more or less technical results concerning the relative order of pairs constitutive of G_α.

There is an easy criterion to decide whether two orders α and β induce the same relative order on all pairs of vertices that are constitutive of G_α in terms of the following remark.

4.1 Remark: Two orders α and β induce the same relative order on any pair u and v of vertices that is constitutive of G_α if and only if they induce the same relative order on the endpoints of any arc uv of G_α. †

Proof: As the endpoints of any arc of G_α are constitutive of G_α, it is obvious that α and β induce the same relative order on the endpoints of all arcs of G_α if they induce the same relative order on any pair of vertices that is constitutive of G_α.

Suppose on the other hand that α and β induce the same relative order on the endpoints of any arc of G_α and consider a pair u and v of vertices that is constitutive of G_α. By the symmetry of $G_\alpha^\wedge$ and $G_\alpha^\vee$, we may assume without loss of generality that $v \in \mathcal{S}(u, G_\alpha^\wedge)$. There is then a directed path $p = (u = x_1, \ldots, x_k = v)$ in $G_\alpha^\wedge$. Since α and β induce the same relative order on any arc of G_α, it follows that $\beta(x_i) < \beta(x_{i+1})$ for all i and hence $\beta(u) \leq \beta(v)$ by the transitivity of "$\leq$". □

If the equality of G_α and G_β is really equivalent to the fact that α and β induce the same relative order on all pairs of vertices that are constitutive of G_α, then this latter condition ought to be symmetric. Apart from being useful in our subsequent proofs, the following lemma may therefore be regarded as indirect evidence for the accuracy of our claim.

4.2 Lemma: Let $G = (V, A)$ be a weighted digraph and let α and β be two orders on its vertices V. If α and β induce the same relative order on each pair of vertices that is constitutive of G_α, then they also induce the same relative order on each pair of vertices that is constitutive of G_β. †

Proof: Let α and β induce the same relative order on each pair of vertices that is constitutive of G_α. In view of № 4.1, it suffices to show that α and β induce the same relative order on the endpoints of any arc of G_β. By the symmetry of $G_\beta^\wedge$ and $G_\beta^\vee$, we may furthermore restrict our attention to arcs $uv \in A_\beta^\wedge$. Consider any arc $uv \in A_\beta^\wedge$. In order to prove that α and β induce the same relative order on u and v, it suffices to show that $\alpha(u) < \alpha(v)$. Let us assume for the sake of contradiction that $\alpha(u) \geq \alpha(v)$. By Theorem 2, there is a vertex $w \in P_\alpha(u,v)$, such that $\operatorname{dist}_G^\wedge(u,w) = \operatorname{dist}_G(u,w)$ and $\operatorname{dist}_G^\vee(w,v) = \operatorname{dist}_G(w,v)$ and such that $\alpha(w) > \min\{\alpha(u), \alpha(v)\}$. Our assumption $\alpha(v) \leq \alpha(u)$ then implies $\alpha(v) < \alpha(w)$. Observe that $\operatorname{dist}_G^\wedge(u,w) < \infty$ implies $w \in \mathcal{S}(u, G_\alpha^\wedge)$ and that u and w are therefore constitutive of G_α. The vertices w and v are constitutive of G_α by an analogous argument involving $\operatorname{dist}_G^\vee(w,v)$. This implies in particular that α and β induce the same relative order on the pairs u, w and w, v and we therefore obtain the inequalities

$$\beta(u) \leq \beta(w) \qquad \text{and} \qquad \beta(v) < \beta(w) \tag{4.1}$$

from $\alpha(u) \leq \alpha(w)$ and our earlier established $\alpha(v) < \alpha(w)$. Recall that $uv \in A_\beta^\wedge$ and hence $\beta(u) < \beta(v)$. In combination with (4.1) we thus obtain $\beta(u) < \beta(w)$ and hence $w \notin \{u,v\}$. Moreover, our choice of $w \in P_\alpha(u,v)$ implies

$$\operatorname{dist}_G(u,v) = \operatorname{dist}_G(u,w) + \operatorname{dist}_G(w,v)\,.$$

Altogether we have thus found a vertex $w \in P_\beta(u,v)$ distinct from both u and v. This contradicts $uv \in A_\beta^\wedge$ and therefore finishes the proof. □

In view of the above lemma, it makes perfect sense to say that α and β induce the same relative order on all constitutive pairs of vertices – without actually referring to G_α or G_β. There is just one minor technicality left before we can finally approach our yet unproven claim.

4.3 Remark: Let $u, v \in V$ with $\operatorname{dist}_G(u,v) < \infty$ and assume $P_\alpha(u,v) \neq \{u,v\}$. Then there exist $w_1, w_2 \in P_\alpha(u,v) \setminus \{u,v\}$ such that both u, w_1 and v, w_2 are constitutive of G_α. †

Proof: Choose $x \in P_\alpha(u,v) \setminus \{u,v\}$. By Theorem 2, there is then some $w \in P_\alpha(u,x)$ such that

$$\begin{aligned} \operatorname{dist}_G^\wedge(u,w) &= \operatorname{dist}_G(u,w) \\ \text{and } \operatorname{dist}_G^\vee(w,x) &= \operatorname{dist}_G(w,x) \end{aligned}$$

and such that $\alpha(w) > \min\{\alpha(u), \alpha(x)\}$. The latter inequality implies $\alpha(w) > \alpha(u)$ since $\alpha(x) \geq \alpha(u)$ by the choice of x. Moreover, $w \in P_\alpha(u,x)$ implies $w \in P_\alpha(u,v)$ and $w \neq v$ by № 2.5 and our choice of x. We therefore find $w \in P_\alpha(u,v) \setminus \{u,v\}$

as claimed. Now, it will suffice to show that $w \in \mathcal{S}(u, G_\alpha^\wedge)$ which follows readily from $\operatorname{dist}_G^\wedge(u, w) = \operatorname{dist}_G(u, w) < \infty$. This proves the existence of w_1 and a completely analogous argument proves the existence of a vertex w_2 possessing the claimed properties. □

Finally, we have mastered all technicalities and are ready to prove the main result of this paragraph.

4.4 Proposition: Let $G = (V, A)$ be a weighted digraph and let α and β be two orders on V. The following statements are equivalent:

(i) $G_\alpha = G_\beta$.

(ii) α and β induce the same relative order on the endpoints of all arcs of G_α.

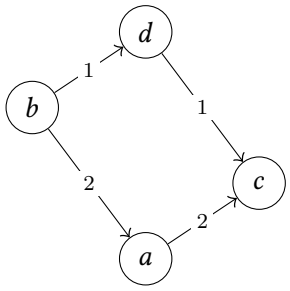
(iii) α and β induce the same relative order on all constitutive pairs of vertices. †

Proof: It is clear that (i) implies (ii). Furthermore, (ii) and (iii) are equivalent by № 4.1. Hence, it remains to show that (iii) implies (i). To this end, in light of № 2.2, we need to consider only $G_\alpha^\wedge$ and $G_\beta^\wedge$. Moreover, the equality of $G_\alpha^\wedge$ and $G_\beta^\wedge$ is already implied by $A_\alpha^\wedge \subseteq A_\beta^\wedge$ since by № 4.2 our claim is symmetric in the sense that the roles of α and β may be interchanged.

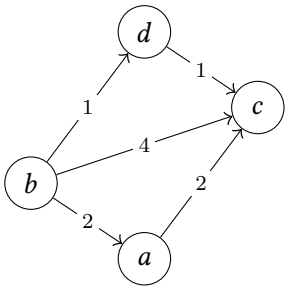
Let us therefore assume that α and β induce the same relative order on all constitutive pairs of vertices and consider some arc $uv \in A_\alpha^\wedge$. Note that $\alpha(u) < \alpha(v)$ by our choice of $uv \in A_\alpha^\wedge$. Since u and v are constitutive of G_α, it follows that $\beta(u) < \beta(v)$, too. In order to show $uv \in A_\beta^\wedge$ it therefore suffices to show $P_\beta(u, v) = \{u, v\}$. To this end, assume $P_\beta(u, v) \neq \{u, v\}$. By № 4.3, we then find some $w \in P_\beta(u, v) \setminus \{u, v\}$, such that u and w are constitutive of G_β. Note that $\operatorname{dist}_G(u, v) = \operatorname{dist}_G(u, w) + \operatorname{dist}_G(w, v)$ and $\min\{\beta(u), \beta(v)\} < \beta(w)$. This latter inequality and the earlier established inequality $\beta(u) < \beta(v)$ imply $\beta(u) < \beta(w)$. Furthermore, u and w are constitutive, so that we obtain $\alpha(u) < \alpha(w)$. Altogether, we have just found a vertex $w \in P_\alpha(u, v)$ that is distinct from both u and v. This contradicts $uv \in A_\alpha^\wedge$ and therefore finishes the proof. □

The above proposition completely classifies the modifications to α that leave G_α unchanged. By negating the conditions (ii) or (iii) of № 4.4, we thus obtain a description of the modifications to α that have an actual effect on G_α. In order to effectively alter G_α, one therefore has to rearrange the relative order of at least one pair of constitutive vertices s and t. We study those modifications to α that leave the relative order of all but one constitutive pair of vertices fixed in the following two paragraphs.

Let us close this paragraph with some remarks about the validity of our findings for weak contraction hierarchies. Observe that speaking of constitutive pairs in weak contraction hierarchies makes perfect sense, for their definition utilised only search spaces, which do also exist in weak contraction hierarchies. Moreover, the



(a) The weak contraction hierarchy H_α depicted as $H_\alpha^\wedge \cup H_\alpha^\vee$.



(b) The weak contraction hierarchy H_β depicted as $H_\beta^\wedge \cup H_\beta^\vee$.

Figure 4.1: Two weak contraction hierarchies H_α and H_β, where α and β induce the same relative order on each pair of vertices that is constitutive of H_α but induce different relative orders on the vertices b and c that are constitutive of H_β. Arc lengths are drawn on the respective arcs and the orders α and β are assumed to increase with the vertical position of the vertices in both Figure 4.1a and Figure 4.1b.

proofs of № 4.1 and № 4.3 merely involved search spaces and the existence of certain paths in G_α, which are of course present in any weak contraction hierarchy, too. Therefore, these two statements also apply to weak contraction hierarchies. However, the proof of № 4.2 depended crucially on the fact that each arc uv of G_α satisfies $P_\alpha(u, v) = \{u, v\}$ – a defining property of contraction hierarchies that is generally not shared by their weak counterparts. It is not only our proof that fails but the statement of № 4.2 is simply not true for weak contraction hierarchies. This may also be seen in Figure 4.1, where there are two contraction hierarchies G_α and G_β such that α and β induce the same relative order on any pair of vertices that is constitutive of G_α but not on any pair of vertices that is constitutive of G_β. We already argued above that № 4.2 should be understood as an indicator of the truth of № 4.4. Moreover, our above example shows that there cannot be a well-defined notion of a constitutive pair of weak contraction hierarchies independent of the order α. Condition (iii) of № 4.4 hence makes no sense for weak contraction hierarchies and it therefore shouldn't come as a surprise that № 4.4, too, does not hold for arbitrary weak contraction hierarchies. This, too, may be seen in Figure 4.1 but there are even simpler counterexamples, which may be constructed by simply inserting some appropriate arc into a given weak contraction hierarchy H_α while keeping the order α fixed. The resulting weak contraction hierarchy and H_α obviously satisfy condition (ii) of № 4.4 but condition (i) fails for them. Furthermore, any possible adaption of condition (iii) to weak contraction hierarchies also holds true in this example, because α and β are exactly the same order.

Despite the above problems concerning an immediate translation of № 4.4 to weak contraction hierarchies, we are still able to draw some important conclusions. To this

end, note that both our counterexamples exploited the fact that weak contraction hierarchies H_α are not completely determined by the order α but may include an almost arbitrary set of additional shortcuts. However, if two weak contraction hierarchies are equal, they contain the same set of additional shortcuts and their orders most certainly induce the same relative order on each pair of vertices that is constitutive of one of those hierarchies. Moreover, if one forbids arbitrary shortcut insertions and deletions and instead concentrates on modifications of the order α only, one indeed obtains an analogon of № 4.4 for weak contraction hierarchies. This is precisely the subject of the following proposition.

4.5 Proposition: Let $G = (V, A)$ be a weighted digraph and let α and β be two orders on V. Further consider two contraction hierarchy H_α and H_β. Then

(a) If $H_\alpha = H_\beta$, then α and β induce the same relative order on all pairs of vertices that are constitutive of H_α or H_β.

(b) If α and β induce the same relative order on each pair of vertices that is constitutive of H_α, then H_α and H_β differ at most by weak shortcuts. †

Proof:

(a) If u and v are constitutive of H_α or H_β, there is a path from u to v in one and hence in both of the weak contraction hierarchies. We may assume without loss of generality that $\alpha(x) < \alpha(y)$ and $\beta(x) < \beta(y)$ for each arc xy of this path. Hence, we see that $\alpha(u) \leq \alpha(v)$ if and only if $\beta(u) \leq \beta(v)$.

(b) If α and β induce the same relative order on each pair of vertices that is constitutive of H_α and H_β, they induce in particular the same relative order on each pair of vertices that is constitutive of G_α and G_β. Now, № 4.4 implies $G_\alpha = G_\beta$. Any arc contained in H_α and not in H_β or vice versa is thus a weak shortcut as claimed. □

Note in particular that the above proposition has the same consequences as the non-weak analogon № 4.4 when it comes to the classification of modifications to α that have an actual effect on H_α. More precisely, we have just shown that in order to achieve an actual effect on H_α with modifications to α only, we have to change the relative order of at least one pair of vertices that is constitutive of H_α.

4.2 Tame Pairs

We saw in the preceding paragraph that a modification of α has to change the relative order of at least one pair of constitutive vertices of a contraction hierarchy G_α or a weak contraction hierarchy H_α in order to have any effect at all. For the purpose of clear analysis, we focus on rearranging the relative order of exactly one pair of

constitutive vertices s and t while keeping the relative order of all other pairs of constitutive vertices fixed. We call such a pair s and t of vertices a *tame pair* and say that any order β which changes the relative order of s and t but retains the relative order of all other pairs of constitutive vertices is obtained from α by *swapping s and t*. In view of № 4.4 it is even plausible to speak of *the* order α obtained from swapping s and t since any two orders β_1 and β_2 obtained from α by swapping s and t necessarily give rise to equal contraction hierarchies G_{β_1} and G_{β_2} or weak contraction hierarchies H_{β_1} and H_{β_2} that only differ by a set of weak shortcuts. It is a priori not clear, which vertex pairs s and t are tame or if there are any tame pairs at all. We will immediately address the former problem and develop a more manageable description of tame pairs but delay a proof that they actually do exist until later. For this purpose, we focus on actual contraction hierarchies G_α but point out to the reader that all our results are equally well applicable to weak contraction hierarchies.

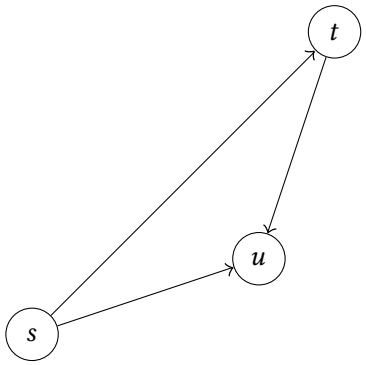
As a step in understanding tame pairs, let us consider the directed path that exists between any two vertices s and t that are constitutive of G_α. It is rather easy to see that this path has to consist of a single arc only if s and t ought to be tame.

4.6 Remark: Let s and t be a tame pair of vertices. Then there is neither in $G_\alpha^\wedge$ nor in $G_\alpha^\vee$ any path with endpoints s and t that contains more than a single arc. Since s and t are constitutive of G_α this means in particular that G_α contains at least one of the arcs st and ts. †

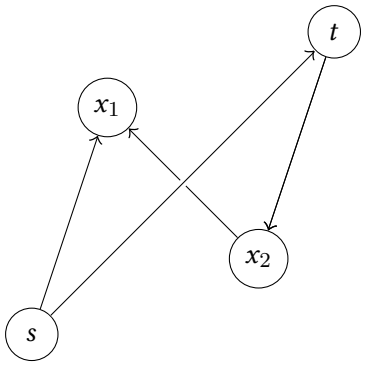
Proof: Let s and t be tame and let β be an order obtained from α by swapping s and t. We may suppose $t \in \mathcal{S}(s, G_\alpha)$ by the symmetry of $G_\alpha^\wedge$ and $G_\alpha^\vee$. Note that $\beta(s) > \beta(t)$ by our choice of β. Now consider any path p in $G_\alpha^\wedge$ or $G_\alpha^\vee$ with endpoints s and t. Let us assume for the sake of contradiction that there is some vertex $u \in p$ distinct from s and t. Then $\alpha(s) < \alpha(u) < \alpha(t)$ and both s, u and u, t are constitutive of G_α. Our choice of β implies in particular that $\beta(s) < \beta(u) < \beta(t)$, which contradicts our earlier established inequality $\beta(s) > \beta(t)$. □

Given a tame pair s and t of vertices, we call the arcs st and ts – of which at least one is guaranteed to exist by the above remark – tame, too.

In view of № 4.6, one might be tempted to conjecture that an arc st is tame if there is no path with endpoints s and t in G_α other than (s, t). However, this turns out to be false in general as may also be seen in Figure 4.2a. Therein, s, u and u, t are pairs of vertices that are constitutive of the depicted contraction hierarchy G_α and any order β that induces the same relative order on each constitutive pair distinct from s and t consequently satisfies $\beta(s) < \beta(u)$, $\beta(u) < \beta(t)$ and hence also $\beta(s) < \beta(t)$. Therefore, st cannot be tame even though there is no path between s and t distinct from (s, t) in neither $G_\alpha^\wedge$ nor $G_\alpha^\vee$. Phrased a bit more generally, it was not a path in $G_\alpha^\wedge$ or $G_\alpha^\vee$ that hindered st from being tame in this example, but a sequence $a_1, \ldots, a_k$ of arcs of G_α, where one endpoint of a_i coincided with one endpoint of a_{i+1}, that is, a path in the undirected graph associated with $G_\alpha^\wedge \cup G_\alpha^\vee$. Indeed, it is easy to see that if there



(a) An arc st that is not tame even though there is no path in $G_\alpha^\wedge \cup G_\alpha^\vee$ between s and t apart from (s,t).



(b) Example of an arc st that is tame despite the fact that there is a path $(s, x_1, x_2, t) \neq (s,t)$ in the undirected graph associated with $G_\alpha^\wedge \cup G_\alpha^\vee$.

Figure 4.2: Example of a non-tame and a tame arc st. Both Figure 4.2a and Figure 4.2b depict the graph $G_\alpha^\wedge \cup G_\alpha^\vee$ for a specific contraction hierarchy G_α. The order on the vertices is assumed to be ascending with the vertical position of the nodes.

is no such path in the undirected graph associated with $G_\alpha^\wedge \cup G_\alpha^\vee$, then st is a tame arc. However, the non-existence of an undirected path from s to t other than (s,t) turns out to be a criterion too strong. This may also be seen in Figure 4.2b, where one can easily convince oneself that st is a tame arc even though (s, x_1, x_2, t) is a path in the undirected graph associated with the shown contraction hierarchy G_α. These two examples strongly suggest that tame arcs may possibly be characterised by requiring that certain paths between s and t in some graph intermediate between G_α and the undirected graph associated with G_α do not exist. Actually, this is precisely the strategy we embark on.

Fix a pair s and t of vertices such that $\alpha(s) < \alpha(t)$ and such that s and t are constitutive of G_α. Let $D_\alpha(s,t)$ denote the set of vertices that are above s and below t, i.e. let

$$D_\alpha(s,t) = \left\{ u \in V \;\middle|\; \alpha(s) \leq \alpha(u) \leq \alpha(t) \right\}.$$

Further let $K_\alpha(s,t)$ denote the union of $G_\alpha^\wedge[D_\alpha(s,t)]$ and $G_\alpha^\vee[D_\alpha(s,t)]$ with all arcs reversed, that is, let $K_\alpha(s,t) = (D_\alpha(s,t), B)$ be given by

$$B = \left\{ uv \mid u, v \in D_\alpha(s,t) \text{ and } uv \in A_\alpha^\wedge \text{ or } vu \in A_\alpha^\vee \right\}.$$

It is precisely this graph $K_\alpha(s,t)$, where we locate the obstruction that possibly hinders a pair s, t from being tame. More precisely, we show that s and t are tame if and only if there is no path from s to t in $K_\alpha(s,t)$ other than (s,t).

4.7 Proposition: Let $s, t \in V$ be a pair of vertices that is constitutive of G_α and suppose $\alpha(s) < \alpha(t)$. The following statements are then equivalent.

(i) The pair s and t is tame.

(ii) There is no path from s to t in $K_\alpha(s,t)$ other than (s,t). †

Proof: During the course of this proof we will simply write K instead of $K_\alpha(s,t)$ and D instead of $D_\alpha(s,t)$.

"(i) ⇒ (ii)": If s and t are tame, then there is an order β on V, such that $\beta(s) > \beta(t)$ and such that α and β induce the same relative order on all constitutive pairs of G_α except on s and t. Assume for the sake of contradiction that there is a path $p = (s = x_1, \dots, x_k = t)$ other than (s,t) in K. Note that since p is distinct from (s,t), we have $\{x_i, x_{i+1}\} \neq \{s,t\}$ for each i. Each arc $x_i x_{i+1}$ of p corresponds to an arc $x_i x_{i+1} \in A_\alpha^\wedge$ or an arc $x_{i+1}x_i \in A_\alpha^\vee$ by construction of K. In either case it is $\alpha(x_i) < \alpha(x_{i+1})$ and thus $\beta(x_i) < \beta(x_{i+1})$ by our choice of β. Altogether, we see that $\beta(s) < \beta(t)$, which contradicts our choice of β.

"(ii) ⇒ (i)": Note that K is an acyclic digraph by construction and that one may therefore speak of the search space $\mathcal{S}(s,K)$ of s in K. We abbreviate $\mathcal{S}(s,K) \setminus \{t\}$ by S. The basic idea of this part of the proof is as follows. We construct an explicit order β by "pushing" S above t and thereby retaining the relative order among all the vertices in S and all the vertices in $D \setminus S$. The orders α and β then induce the same relative order on any two vertices that are both contained in S or both contained in $D \setminus S$. Moreover, if $u \in S$ and $v \in D \setminus S$ are constitutive of G_α, then $\alpha(v) < \alpha(u)$, for there would otherwise be a path from s via u to v in K witnessing $v \in S$. This is also depicted in Figure 4.3, where a sketch of both the original order α and the order β may be seen.

Let us make this argument precise. We write $s = x_0, \dots, x_k$ to denote the vertices contained in S and $y_0, \dots, y_{l-1} = t$ to denote those in $D \setminus S$. We further assume the vertices x_i and y_i to be sorted by ascending values of α, that is

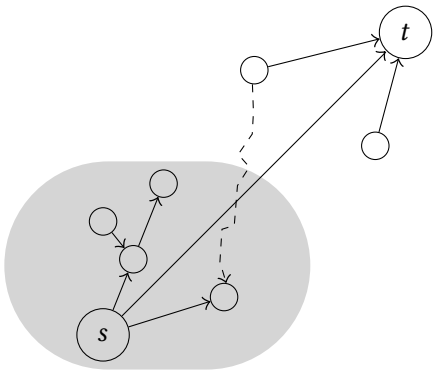
$$\alpha(s) = \alpha(x_0) < \cdots < \alpha(x_k)$$

and

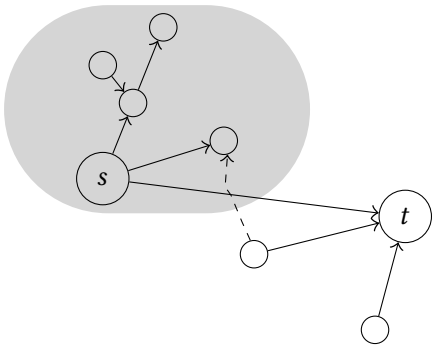
$$\alpha(y_0) < \cdots < \alpha(y_{l-1}) = \alpha(t)\,.$$

Let us define an order β on V by

$$\beta(u) = \begin{cases} \alpha(t) - k + i & \text{if } u = x_i \\ \alpha(t) - k - l + i & \text{if } u = y_i \\ \alpha(u) & \text{otherwise.} \end{cases} \tag{4.2}$$



(a) The graph $G_\alpha^\wedge \cup G_\alpha^\vee$. The vertices of $S = \mathcal{S}(s, K) \setminus \{t\}$ are fit in a grey shape. The dashed path corresponds to the one in Figure 4.3b.



(b) The order of the vertices after "moving" S above t while retaining the relative order among the vertices in S and among the vertices in $D \setminus S$.

Figure 4.3: Sketch of the construction of an order β that changes the relative order of the endpoints of a tame arc st and fixes the relative order between any other pair of constitutive vertices. The orders α and β on the vertices are indicated by their vertical position. Any constitutive pair of vertices other than $\{s, t\}$ on which α and β do not induce the same relative order, gives rise to a path like the dashed one in Figure 4.3b. Any such path would violate the assumptions of № 4.7, for it is part of a path from s to t in K other than (s, t).

Note that β is well-defined. Further observe that $\alpha(s) = \alpha(t) - k - l$ and that

$$\beta\Big(\{x_0, \dots, x_k\}\Big) = \Big\{\alpha(t) - k, \dots, \alpha(t)\Big\}$$

and

$$\beta\Big(\{y_0, \dots, y_{l-1}\}\Big) = \Big\{\alpha(t) - k - l, \dots, \alpha(t) - k - 1\Big\}.$$

The map $\beta\colon V \to \{1, \dots, n\}$ is therefore in fact bijective. Moreover, $\beta(s) > \beta(t)$ by construction. It only remains to verify that α and β induce the same relative order on any pair of vertices that is constitutive of G_α but distinct from $\{s, t\}$.

To this end, note that α and β induce the same relative order on any two vertices u and v, where $\{u, v\} \not\subseteq D$. Let us now consider some pair u and v of vertices constitutive of G_α such that $\{u, v\} \subseteq D$ and such that $\{u, v\} \neq \{s, t\}$. Note that if $\{u, v\} \subseteq S$ or if $\{u, v\} \subseteq D \setminus S$, then we are finished with the proof, for α and β induce the same relative order on any such pair by the construction of β. Let us consider the case $u \in S$ and $v \in D \setminus S$ instead. Note that $\beta(v) < \beta(u)$ for any such pair by the construction of β. If $u \in S$, there is a path p from s to u in K. Furthermore, u and v are constitutive of G_α, so that there is a directed path q between u and v in $G_\alpha^\wedge$ or $G_\alpha^\vee$. Observe that q induces a path q_K in K with source u and target v if it is a path with source u and target v in $G_\alpha^\wedge$ or a path with source v and target u in $G_\alpha^\vee$. The concatenation $p \cdot q_K$ would then

be a path from s to v in K and would therefore witness $v \in \mathcal{S}(s,K) \setminus S$. This implies $v = t$ and hence $p \cdot q_K = (s,t)$ by assumption. Since $u \in p \cdot q_K$, it follows that $\{u,v\} = \{s,t\}$, which contradicts our initial choice of u and v. We may therefore conclude that q is a path with source v and target u in $G_\alpha^\wedge$ or a path with source u and target v in $G_\alpha^\vee$. In either case it is $\alpha(v) < \alpha(u)$, which finishes the proof since we already saw $\beta(v) < \beta(u)$ earlier. □

The above proposition finally enables us to formally confirm the existence of tame pairs. It even turns out that each vertex is incident to at least one tame arc.

4.8 Corollary: Let K_α denote the union of $G_\alpha^\wedge$ and $G_\alpha^\vee$ with all arcs reversed and let s be a vertex. Furthermore, let us be the arc of K_α with maximal value of $\alpha(u)$ and let sv be the arc of K_α with minimal value of $\alpha(v)$. Then both u, s and s, v are tame.†

Proof: We restrict our attention to the arc us only, for the proof of the statement concerning sv is completely analogous.

Note that $K_\alpha(u,s) \subseteq K_\alpha$. Suppose that u and s are not tame, i.e. there is a path p from u to s in $K(u,s)$ other than (u,s). Decompose p into $p = (u,w) \cdot q$ for some vertex $w \notin \{u,s\}$. Since all arcs in K_α are pointing upwards, it now follows that $\alpha(u) < \alpha(w) < \alpha(s)$, which contradicts our choice of u. □

Note that № 4.7 does not only classify the set of tame arcs in a convenient way but has some rather practical consequences, too. The characterisation of tame arcs st by the existence of a path in $K_\alpha(s,t)$ is perfectly suited for the employment in an algorithm. Moreover, for such an algorithm, it is neither necessary to explicitly compute $K_\alpha(s,t)$ nor to examine the whole of $K_\alpha(s,t)$. Consider Algorithm 4.1. Given an arc $st \in A_\alpha^\wedge$ or $ts \in A_\alpha^\vee$, it performs an implicit depth-first traversal with source vertex s in $K_\alpha(s,t)$ by taking into consideration only the arcs $uv \in A_\alpha^\wedge$ and $vu \in A_\alpha^\vee$ with $\alpha(v) < \alpha(t)$. As soon as the depth-first traversal discovers an arc $ut \in A_\alpha^\wedge$ or $tu \in A_\alpha^\vee$, where $u \neq s$, it returns **False**, for it has found a path from s to t in $K_\alpha(s,t)$ different from (s,t). If the algorithm terminates without ever finding such an arc $ut \in A_\alpha^\wedge$ or $tu \in A_\alpha^\vee$, it returns **True**, for there cannot be a path from s to t other than (s,t). This algorithm is clearly correct and may be implemented with running time $\mathcal{O}(\#\mathcal{S}(s,K_\alpha(s,t)) + m_K)$, where m_K denotes the number of arcs in $\mathcal{S}(s,K_\alpha(s,t))$. We have just proven the following corollary.

4.9 Corollary: Given an arc $st \in A_\alpha^\wedge$ or $ts \in A_\alpha^\vee$, Algorithm 4.1 decides whether the given arc is tame in time $\mathcal{O}(\#\mathcal{S}(s,K_\alpha(s,t)) + m_K)$, where m_K denotes the number of arcs in $\mathcal{S}(s,K_\alpha(s,t))$. †

Algorithm 4.1 may be adapted to not only decide whether one specific arc st is tame but to rather find all tame arcs $st \in A_\alpha^\wedge$ and $ts \in A_\alpha^\vee$ for a prescribed vertex s. For this purpose, the algorithm traverses $K_\alpha(s,t)$, where t is the neighbour of s with

Algorithm 4.1: Deciding whether a given arc st is tame

Data: (Weak) Contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$
Input: Arc st of G_α
Output: **True** if st is tame and **False** otherwise

```
/* Initialisation                                                    */
(s, t) ← (s, t) if α(s) < α(t) else (t, s)
Q ← ⟨s⟩
D ← ∅
/* Depth-first traversal of S(s, K_α(s, t))                          */
while Q ≠ ⟨⟩ do
    u ← Q.pop()
    D ← D ∪ {u}
    foreach uv ∈ A_α^∧, vu ∈ A_α^∨ do
        if u ≠ s and v = t then return False      // A path ≠ (s, t) from s to t
        if α(v) < α(t) and v ∉ D then Q.push(v)
return True
```

maximal $\alpha(t)$ and does not terminate when it discovers a path other than (s, v) but simply marks the arcs sv and vs as not tame and continues until all of $\mathcal{S}(s, K_\alpha(s, t))$ is eventually taken care of. An implementation of this approach in pseudo code may be seen in Algorithm 4.2. As marking arcs is clearly possible in constant time, the time complexity of this algorithm is again $\mathcal{O}(\#\mathcal{S}(s, K_\alpha(s, t)) + m_K)$. Altogether, we have the following corollary.

4.10 Corollary: Given a vertex s, Algorithm 4.2 computes the set

$$\left\{ st \in A_\alpha^\wedge \;\middle|\; st \text{ is tame} \right\} \cup \left\{ ts \in A_\alpha^\vee \;\middle|\; ts \text{ is tame} \right\}$$

in time $\mathcal{O}\left(\#\mathcal{S}(s, K_\alpha(s, t)) + m_K\right)$, where t is the neighbour of s in G_α with maximal $\alpha(t)$ and where m_K denotes the number of arcs in $\mathcal{S}(s, K_\alpha(s, t))$. †

Note that the subgraph of G consisting only of the tame arcs is also known as the *transitive reduction* of G. The problem of determining the transitive reduction of a given directed graph is an interesting problem on its own and was first studied by Aho, Garey and Ullman [AGU72]. An algorithm solving the problem on directed acyclic graphs with running time $\mathcal{O}(n + m_r \cdot k)$, where m_r denotes the number of tame arcs and where k denotes the width of a so-called chain-decomposition of G, was given by Simon [Sim88].

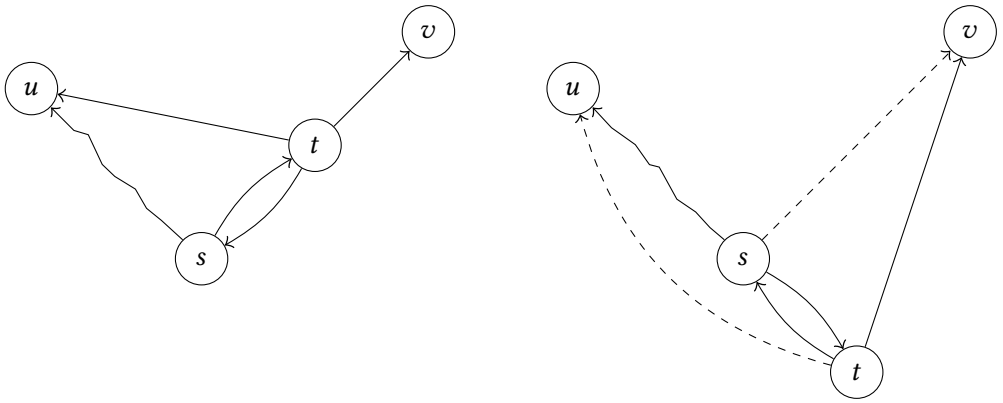
Algorithm 4.2: Computation of all tame arcs $st \in A_\alpha^\wedge$ and $ts \in A_\alpha^\vee$ for a given vertex s

Data: (Weak) Contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$
Input: Vertex s of G_α
Output: Set of all tame arcs $st \in A_\alpha^\wedge$ or $ts \in A_\alpha^\vee$.

```
/* Initialisation                                                    */
T ← { su | su ∈ A_α^∧ } ∪ { us | us ∈ A_α^∨ }
Q ← ⟨s⟩
D ← ∅
/* Depth-first traversal of S(s, K_α(s, t))                          */
while Q ≠ ⟨⟩ do
    u ← Q.pop()
    D ← D ∪ {u}
    foreach uv ∈ A_α^∧, vu ∈ A_α^∨ do
        if u ≠ s then T ← T ∖ {uv, vu}        // A path ≠ (s, v) from s to v
        if α(v) < α(t) and v ∉ D then Q.push(v)
return T
```

4.3 Swapping of Tame Pairs

After being able to determine the set of tame pairs in a given possibly weak contraction hierarchy G_α and having seen an explicit construction of the order β obtained from α by swapping the endpoints of a tame arc, there still remains one major question to be answered: How exactly does G_α differ from G_β? Prior to delving into the technical details, let us try to develop some intuition for what should happen upon swapping the relative order of a tame pair s and t. The structure of G_α and G_β, respectively, essentially depends on the sets $P_\alpha(u, v)$ and $P_\beta(u, v)$ only. Both $P_\alpha(u, v)$ and $P_\beta(u, v)$ depend on the respective order α or β and the distances and shortest paths in G. However, changing the order has no impact whatsoever on the distances in G and any discrepancy between G_α and G_β therefore has to originate from a discrepancy between α and β. More precisely, any vertex $x \in P_\alpha(u, v)$ with $x \notin P_\beta(u, v)$ necessarily satisfies $\alpha(x) > \min\{\alpha(u), \alpha(v)\}$ but also $\beta(x) < \min\{\beta(u), \beta(v)\}$. That is, x was "moved downwards" relative to u and v when we swapped s and t. If we consider the explicit construction (4.2) of the order β and borrow related notation like S and D from the proof of № 4.7, it is clear that the only vertices that are moved downwards in this sense are those of $D \setminus S$ and the only vertices that are moved upwards are those of S. That said, one might get an idea of what G_β should look like by closely inspecting Figure 4.4. Assuming that s lies on a shortest path from t to u, one realises that the arc tu in Figure 4.4a becomes obsolete after swapping s and t in Figure 4.4b, for there



(a) The digraph $G_\alpha^\wedge \cup G_\alpha^\vee$ before swapping the relative order of a tame pair s and t of vertices.

(b) The digraph $G_\beta^\wedge \cup G_\beta^\vee$ obtained from swapping the vertices s and t in the hierarchy $G_\alpha^\wedge \cup G_\alpha^\vee$ shown to the left.

Figure 4.4: Sketch of the neighbourhood of a tame pair s and t before and after swapping their relative order. The vertex s is assumed to lie on a shortest path from t to u and t is assumed to lie on a shortest path from s to v. Observe that the arc tu in Figure 4.4a is obsolete in Figure 4.4b, for there is now a path $(t, s, \dots, u)$ of equal length. Observe further that the dashed shortcut sv in Figure 4.4b becomes necessary since the former path (s, t, v) of $G_\alpha^\wedge$ is not contained in $G_\beta^\wedge$ anymore.

is then the path $(t, s, \dots, u)$ of equal length. Moreover, the potential shortcut sv is unnecessary in Figure 4.4a, for there is still the path (s, t, v). This path is missing in Figure 4.4b after swapping s and t, which is why a new shortcut sv becomes necessary.

In what follows below, we make the above discussion precise and are eventually able to deploy a proof of Theorem 9, which completely determines the structure of G_β in terms of G_α only. We additionally aim at a characterisation of G_β that is applicable in actual algorithms to compute G_β without the need to construct a whole contraction hierarchy from scratch. Albeit rather technical, this material is certainly the technical centrepiece of the whole current chapter. Let us fix some notation for this undertaking. We consider a contraction hierarchy G_α and a tame pair s and t with $\alpha(s) < \alpha(t)$. We denote by β the order obtained from α by swapping s and t. We further assume without loss of generality that β is precisely the order β constructed in (4.2) during the proof of № 4.7. In that instance, we obtained β by "pushing the vertices $x \in S$ above t", where S denoted the search space $\mathcal{S}(s, K) \setminus \{t\}$ of s in the graph $K = K_\alpha(s, t)$ with vertices $D = D_\alpha(s, t)$. Additionally, we do not only borrow the definition of β but also this related notation from the proof of № 4.7.

Let us start off by reassuring ourselves that swapping the endpoints of a tame arc st preserves this arc in an appropriate manner, that is, moves st from $G_\alpha^\wedge$ to $G_\alpha^\vee$.

Note that this is precisely what is depicted in Figure 4.4 and was also implicit but unsubstantiated in our informal discussion above.

4.11 Remark:

(a) If $st \in A^{\wedge}_{\alpha}$, then $st \in A^{\vee}_{\beta}$.

(b) If $ts \in A^{\vee}_{\alpha}$, then $ts \in A^{\wedge}_{\beta}$. †

Proof: Note that (a) and (b) are symmetric, so that it suffices to prove (a) only. In order to show $st \in A^{\vee}_{\beta}$ it further suffices to prove $P_\beta(s,t) = \{s,t\}$ since $\beta(t) < \beta(s)$ by assumption. Let us assume the contrary, i.e. $P_\beta(s,t) \neq \{s,t\}$. There is then some $x \in P_\beta(s,t) \setminus \{s,t\}$. By the construction (4.2) of β, we see that x is either an element of S or is not contained in D at all. Note that α and β then induce the same relative order on x and s and we have thus found a vertex $x \in P_\alpha(s,t) \setminus \{s,t\}$. This contradicts $st \in A^{\wedge}_{\alpha}$ and therefore finishes the proof. □

The argument used to prove the above remark is exemplary for the more involved proofs below. We mostly exploit our explicit construction (4.2) of β to show that certain elements of $P_\beta(u,v)$ are also contained in $P_\alpha(u,v)$ or vice versa. Note that these kinds of arguments would be far more involved with an arbitrary order obtained from α by swapping s and t, for we would have to find ad-hoc justifications for each argument involving the relative order induced by α and β on some non-constitutive pair of vertices.

The following lemma provides a first characterisation of the the arcs of G_α that become superfluous after swapping s and t.

4.12 Lemma: Let u and v be vertices with $\alpha(u) < \alpha(v)$. The statements

(i) $uv \in A^{\wedge}_{\alpha}$ and $P_\beta(u,v) \neq \{u,v\}$

(ii) $uv \in A^{\wedge}_{\alpha}$ and $u = t$ and $\operatorname{dist}_G(t,s) + \operatorname{dist}_G(s,v) = \operatorname{dist}_G(t,v) < \infty$

as well as

(i′) $vu \in A^{\vee}_{\alpha}$ and $P_\beta(v,u) \neq \{v,u\}$

(ii′) $vu \in A^{\vee}_{\alpha}$ and $u = t$ and $\operatorname{dist}_G(v,s) + \operatorname{dist}_G(s,t) = \operatorname{dist}_G(v,t) < \infty$

are equivalent. Moreover, $ts \in A^{\vee}_{\alpha}$ or $st \in A^{\wedge}_{\alpha}$ whenever there exists an arc $tv \in A^{\wedge}_{\alpha}$ satisfying (i) and (ii) or an arc $vt \in A^{\vee}_{\alpha}$ satisfying (i′) and (ii′), respectively. †

Proof: Note that it suffices to prove the equivalence of (i) and (ii), for the equivalence of (i′) and (ii′) then follows by reversing all the arcs. More precisely, we show that conditions (i) and (ii) are equivalent and that the existence of an arc $uv \in A^{\wedge}_{\alpha}$ satisfying condition (i) implies $ts \in A^{\vee}_{\alpha}$.

"(ii) ⇒ (i)": Note that $uv \in A^{\wedge}_{\alpha}$ and $u = t$ imply $\alpha(v) > \alpha(t)$ and hence $v \neq s$. Furthermore, our assumption $\mathrm{dist}_G(t,s) + \mathrm{dist}_G(s,v) = \mathrm{dist}_G(t,v) < \infty$ immediately gives $s \in P_\beta(u,v) \setminus \{u,v\}$ since $u = t$, $\beta(s) > \beta(t)$ and $s \neq v$.

"(i) ⇒ (ii)": Note that $P_\alpha(u,v) = \{u,v\}$ by our assumption $uv \in A^{\wedge}_{\alpha}$. We choose a vertex $x \in P_\beta(u,v) \setminus \{u,v\}$ such that $\alpha(x)$ is maximal. By this choice of x, we then have $\beta(x) > \min\{\beta(u), \beta(v)\}$. Moreover, $\alpha(x) < \min\{\alpha(u), \alpha(v)\}$, for otherwise x would be an element of $P_\alpha(u,v)$ distinct from both u and v. Note that $\alpha(u) = \min\{\alpha(u), \alpha(v)\}$ since $uv \in A^{\wedge}_{\alpha}$ by assumption. A close look at the definition (4.2) of β now reveals that the inequalities $\alpha(x) < \alpha(u) < \alpha(v)$ and $\beta(x) > \min\{\beta(u), \beta(v)\}$ can be satisfied simultaneously only if $x \in S$ and at least one of the vertices u and v is contained in $D \setminus S$. For the moment, let us assume $u \notin D \setminus S$. Then u is either an element of S or not contained in D at all. It turns out that α and β induce the same relative order on x and u in both cases. Since $\alpha(u) > \alpha(x)$ and $\beta(x) > \min\{\beta(u), \beta(v)\}$ are satisfied simultaneously, it follows that $\beta(u) > \beta(x) > \beta(v)$. Observe that α and β induce different relative orders on u and v; Since s and t are tame by assumption, we find $u = s$ and $v = t$, which contradicts № 4.11. Altogether, we may thus draw the conclusion that $x \in S$ and $u \in D \setminus S$.

We claim that $ux \in A^{\vee}_{\alpha}$. Let us assume the contrary, i.e. $P_\alpha(u,x) \neq \{u,x\}$. According to № 4.3, we may then choose a vertex $y \in P_\alpha(u,x) \setminus \{u,x\}$ such that $y \in \mathcal{R}(x, G^{\vee}_{\alpha})$. Observe that $\alpha(y) < \alpha(u)$ because otherwise

$$\begin{aligned} &\mathrm{dist}_G(u,v) = \mathrm{dist}_G(u,x) + \mathrm{dist}_G(x,v) = \\ &\quad \mathrm{dist}_G(u,y) + \mathrm{dist}_G(y,x) + \mathrm{dist}_G(x,v) \geq \mathrm{dist}_G(u,y) + \mathrm{dist}_G(y,v) \end{aligned} \tag{4.3}$$

would prove $y \in P_\alpha(u,v) \setminus \{u,v\}$. Since s and t are tame and since $x \in S$ and $y \in \mathcal{R}(x, G^{\vee}_{\alpha})$, either $y \in S$, too, or $x = s$ and $y = t$. The latter case is contradictory as it would imply $\alpha(t) = \alpha(y) < \alpha(u)$ and hence $u \notin D$. We therefore conclude $y \in S$. This implies $\beta(y) > \beta(x)$ since both x and y are elements of S and α and β induce the same relative order on any two vertices contained in S by the construction of β. By (4.3), we may now conclude that $y \in P_\beta(u,v)$. Note that $\alpha(y) > \alpha(x)$ since $\alpha(y) > \min\{\alpha(u), \alpha(x)\}$ by the choice of y and $\alpha(y) < \alpha(u)$. This contradicts our choice of x and we thus find $P_\alpha(u,x) = \{u,x\}$, hence $ux \in A^{\vee}_{\alpha}$.

Altogether we therefore have $ux \in A^{\vee}_{\alpha}$ and $x \in S$ and $u \in D \setminus S$. Since s and t are tame by assumption, it follows that $u = t$ and $x = s$. This finishes this proof, for $ux = ts \in A^{\vee}_{\alpha}$ and $x = s$ was chosen from $P_\beta(u,v)$, hence satisfies $\mathrm{dist}_G(u,v) = \mathrm{dist}_G(u,s) + \mathrm{dist}_G(s,v)$. ▫

Observe that the statement of the above lemma coincides with what we arrived at by our preceding discussion. The only arcs $uv \in A^{\wedge}_{\alpha}$ that become superfluous upon

swapping s and t have source $u = t$ and there is a condition for the disappearance of such an arc tv in terms of the existence of a shortest path from t to v containing s. However, the condition we conjectured differs from the one we have proven in that we initially used arguments in G_α but № 4.12 employs distances in G. The following remark bridges this gap and fully justifies our informal discussion regarding arcs tv of $G_\alpha^\wedge$ that do not appear in G_β.

4.13 Remark: Let u and v be vertices with $\alpha(u) < \alpha(v)$.

(a) There is a path $p = (s, \dots, v)$ in $G_\alpha^\wedge$ of length $\operatorname{dist}_G(s, v)$ for each arc $tv \in A_\alpha^\wedge$ satisfying the equivalent conditions (i) and (ii) of № 4.12.

(b) There is a path $p = (v, \dots, s)$ in $G_\alpha^\vee$ of length $\operatorname{dist}_G(v, s)$ for each arc $vt \in A_\alpha^\vee$ satisfying the equivalent conditions (i′) and (ii′) of № 4.12. †

Proof: It suffices to prove (a) since (b) then follows by symmetry. Assume that there is no path p satisfying our claim. According to Theorem 2 there exists then some vertex $x \in P_\alpha(s, v)$, such that $\operatorname{dist}_G^\wedge(s, x) = \operatorname{dist}_G(s, x)$ and $\operatorname{dist}_G^\vee(x, v) = \operatorname{dist}_G(x, v)$. Note that $\alpha(x) > \alpha(v)$ since otherwise $x = v$ and we would have found a path p as claimed. Furthermore, our choice of $x \in P_\alpha(s, v)$ and condition (ii) of № 4.12 imply

$$\begin{aligned}\operatorname{dist}_G(t, v) &= \operatorname{dist}_G(t, s) + \operatorname{dist}_G(s, v) \\ &= \operatorname{dist}_G(t, s) + \operatorname{dist}_G(s, x) + \operatorname{dist}_G(x, v) \geq \operatorname{dist}_G(t, x) + \operatorname{dist}_G(x, v)\,,\end{aligned}$$

which proves $x \in P_\alpha(t, v)$. This contradicts $tv \in A_\alpha^\wedge$ as $\alpha(x) > \alpha(v)$ implies that x is distinct from both t and v. □

Combining the above remark with № 4.12, we already obtain a rather convenient description of the arcs of G_α that are missing in G_β. Due to the restrictions imposed by № 4.13, we are even able to compute the sets $A_{\alpha\smallsetminus\beta}^\wedge = A_\alpha^\wedge \smallsetminus A_\beta^\wedge$ and $A_{\alpha\smallsetminus\beta}^\vee = A_\alpha^\vee \smallsetminus A_\beta^\vee$ quite efficiently. To this end, consider Algorithm 4.3. Provided that $ts \in A_\alpha^\vee$ this algorithm collects all the arcs $tv \in A_\alpha^\wedge$ whose target v is reachable from s and that further satisfy

$$\operatorname{len}_G^\alpha(tv) = \operatorname{len}_G^\alpha(ts) + \operatorname{dist}_G^\wedge(s, v)\,.$$

Note that this latter equation is in fact equivalent to condition (ii) of № 4.12 because

$$\operatorname{len}_G^\alpha(tv) = \operatorname{dist}_G(t, v)\,, \quad \operatorname{len}_G^\alpha(ts) = \operatorname{dist}_G(t, s)$$

and

$$\operatorname{dist}_G^\wedge(s, v) \geq \operatorname{dist}_G(s, v)\,.$$

It thus follows from № 4.12 and № 4.13 that Algorithm 4.3 correctly determines the set of arcs $uv \in A_\alpha^\wedge$ such that $P_\beta(u, v) \neq \{u, v\}$. Furthermore, the only arc of $G_\alpha^\wedge$ that is possibly missing from $G_\beta^\wedge$ but not contained in this set, is the arc st. However,

Algorithm 4.3: Computation of the arcs of $G_\alpha^\wedge$ that become superfluous after swapping a tame pair s and t in a contraction hierarchy G_α

Data: Contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$
Input: Tame pair s and t with $\alpha(s) < \alpha(t)$.
Output: Arcs $A^\wedge_{\alpha\setminus\beta} = A^\wedge_\alpha \setminus A^\wedge_\beta$, where β is the order obtained from α by swapping s and t.

1 $A^\wedge_{\alpha\setminus\beta} \leftarrow \{st\}$ **if** $st \in A^\wedge_\alpha$ **else** $\varnothing$
2 **if** $ts \in A^\vee_\alpha$ **then**
3 Compute $\operatorname{dist}^\wedge_G(s, v)$ for all $v \in \mathcal{S}(s, G^\wedge_\alpha)$
4 **foreach** $tv \in A^\wedge_\alpha$ **do**
5 **if** $\operatorname{len}^\alpha_G(tv) = \operatorname{len}^\alpha_G(ts) + \operatorname{dist}^\wedge_G(s, v)$ **then** $A^\wedge_{\alpha\setminus\beta} \leftarrow A^\wedge_{\alpha\setminus\beta} \cup \{tv\}$
6 **return** $A^\wedge_{\alpha\setminus\beta}$

Algorithm 4.3 takes care of st in line 1, so that it is correct indeed. Further note that Algorithm 4.3 has running time $\mathcal{O}(\mathsf{T}_{\text{query}} + \deg(t))$, as it computes $\operatorname{dist}^\wedge_G(s, v)$ for all $v \in \mathcal{S}(s, G^\wedge_\alpha)$ and subsequently examines each arc $tv \in A^\wedge_\alpha$ once. Moreover, there is an analogous algorithm that computes $A^\vee_{\alpha\setminus\beta}$ with running time $\mathcal{O}(\mathsf{T}_{\text{query}} + \deg(t))$. Altogether, we have just proven the following corollary.

4.14 Corollary: For a given contraction hierarchy G_α there is an algorithm that computes the sets $A^\wedge_\alpha \setminus A^\wedge_\beta$ and $A^\vee_\alpha \setminus A^\vee_\beta$ in time $\mathcal{O}\big(\mathsf{T}_{\text{query}} + \deg(t)\big)$. †

Algorithm 4.3 is a quite satisfactory solution to the problem of determining the arcs of G_α that become superfluous upon swapping s and t. Therefore, all that remains is to get hold of the arcs that are contained in G_β but not in G_α. To a certain extent № 4.12 already allows us to argue about those arcs. Swapping s and t twice yields G_α again and the arcs of G_β that are not contained in G_α are therefore precisely those arcs that become superfluous upon swapping s and t in G_β, hence the following corollary.

4.15 Corollary: Let u and v be vertices with $\alpha(u) < \alpha(v)$. The statements

(i) $uv \in A^\wedge_\beta$ and $P_\alpha(u, v) \neq \{u, v\}$
(ii) $uv \in A^\wedge_\beta$ and $u = s$ and $\operatorname{dist}_G(s, t) + \operatorname{dist}_G(t, v) = \operatorname{dist}_G(s, v) < \infty$

as well as

(i′) $vu \in A^\vee_\beta$ and $P_\alpha(v, u) \neq \{v, u\}$
(ii′) $vu \in A^\vee_\beta$ and $u = s$ and $\operatorname{dist}_G(v, t) + \operatorname{dist}_G(t, s) = \operatorname{dist}_G(v, s) < \infty$

are equivalent. †

The above corollary is hardly suited for a computation of G_β from G_α, for it still involves G_β. However, reasoning quite similar to the proof of № 4.12 leads to a characterisation of the additional arcs in G_β using conditions in G_α only.

4.16 Lemma: The statements

(i) $sv \in A_\beta^\wedge$ and $\operatorname{dist}_G(s,t) + \operatorname{dist}_G(t,v) = \operatorname{dist}_G(s,v) < \infty$.

(ii) (s,t,v) is the only path from s to v of length $\operatorname{dist}_G(s,v)$ in $G_\alpha^\wedge$ and there is furthermore no vertex $x \in P_\alpha(s,v) \setminus \{v\}$, such that $\operatorname{dist}_G^\wedge(s,x) = \operatorname{dist}_G(s,x)$ and $\operatorname{dist}_G^\vee(x,v) = \operatorname{dist}_G(x,v)$.

as well as

(i′) $vs \in A_\beta^\vee$ and $\operatorname{dist}_G(v,t) + \operatorname{dist}_G(t,s) = \operatorname{dist}_G(v,s)$.

(ii′) (v,t,s) is the only path from v to s of length $\operatorname{dist}_G(v,s)$ in $G_\alpha^\vee$ and there is furthermore no vertex $x \in P_\alpha(v,s) \setminus \{v\}$, such that $\operatorname{dist}_G^\wedge(v,x) = \operatorname{dist}_G(v,x)$ and $\operatorname{dist}_G^\vee(x,s) = \operatorname{dist}_G(x,s)$.

are equivalent. †

Proof: Like before, it is sufficient to prove the equivalence of (i) and (ii) since the equivalence of (i′) and (ii′) then follows by symmetry.

"(i) ⇒ (ii)": Let us first prove $st \in G_\alpha^\wedge$. Assume the contrary, i.e. that there exists some vertex $x \in P_\alpha(s,t) \setminus \{s,t\}$. Observe that $P_\beta(s,t) \subseteq P_\beta(s,v)$ and $v \notin P_\beta(s,t)$, for t lies on a shortest path from s to v by assumption. According to № 4.3, we may assume without loss of generality that $x \in \mathcal{S}(s, G_\alpha^\wedge)$. Since $x \neq t$, it then follows that either $x \in S$ or $x \notin D$. In both cases, α and β induce the same relative order on s and x. Note that this implies $x \in P_\beta(s,t)$, hence $x \in P_\beta(s,v)$. This contradicts our assumption $sv \in A_\beta^\wedge$ and thus proves $st \in G_\alpha^\wedge$.

Let us now consider t and v. Note that $sv \in A_\beta^\wedge$ implies $\beta(v) > \beta(s)$ and thus either $v \in S$ or $v \notin D$. We claim that only the latter case occurs. Let us assume the contrary, i.e. $v \in S$. Choose $x \in P_\alpha(t,v)$ such that $\operatorname{dist}_G^\wedge(t,x) = \operatorname{dist}_G(t,x)$ and $\operatorname{dist}_G^\vee(x,v) = \operatorname{dist}_G(x,v)$. Such a vertex does always exist by Theorem 2. Observe that $x \neq t$, for otherwise $\operatorname{dist}_G^\vee(x,v) < \infty$ would imply the existence of a path from s via $v \in S$ to t in K_α that would be distinct from (s,t) since $v \notin \{s,t\}$. This means in particular that $\alpha(x) > \alpha(t)$ and thus $x \notin D$. Note that this also implies $\alpha(x) > \alpha(v)$ since $v \in S$. Moreover, α and β induce the same relative order on $x \notin D$ and $v \in S$ and we thus find $\beta(x) > \beta(v)$. Utilising our assumptions and the choice of $x \in P_\alpha(t,v)$ one further computes

$$\begin{aligned}\operatorname{dist}_G(s,v) &= \operatorname{dist}_G(s,t) + \operatorname{dist}_G(t,v)\\ &= \operatorname{dist}_G(s,t) + \operatorname{dist}_G(t,x) + \operatorname{dist}_G(x,v) \geq \operatorname{dist}_G(s,x) + \operatorname{dist}_G(x,v)\end{aligned}$$

and hence finds $x \in P_\beta(s,v)$. This contradicts our assumption $sv \in A^\wedge_\beta$ since $\alpha(x) > \alpha(t)$ implies both $x \neq s$ and $x \neq v$ as seen above.

Altogether, we may draw the conclusion that $v \notin D$ and thus $\alpha(v) > \alpha(t)$. In order to show $tv \in A^\wedge_\alpha$, it now suffices to prove $P_\alpha(t,v) = \{t,v\}$. However, any vertex $x \in P_\alpha(t,v) \setminus \{t,v\}$ satisfies $\alpha(x) > \alpha(t)$ and is thus not contained in D. This implies in particular that α and β induce the same relative order on t and any such $x \in P_\alpha(t,v)$. It follows that $P_\alpha(t,v) \subseteq P_\beta(t,v) = \{t,v\}$, which finishes the proof of $tv \in A^\wedge_\alpha$.

Up to this point, we have proven the existence of a path (s,t,v) in G_α. Note that this path has length $\operatorname{dist}_G(s,v)$ by the definition of the arc lengths in G_α and our assumptions. Let us now show that there is no other path from s to v of length $\operatorname{dist}_G(s,v)$ in $G^\wedge_\alpha$. For this purpose consider an arbitrary such path p. Note that $p \neq (s,v)$ since $sv \notin A^\wedge_\alpha$. We may thus choose some vertex $x \in p$ distinct from both s and v. Since there is a path from s to x and since s and t are tame by assumption, it follows that either $x = t$, $x \in S$ or $x \notin D$. The latter two cases are obviously contradictory since α and β would induce the same relative order on s and x and we would therefore find $x \in P_\beta(s,v)$. Let us suppose $x = t$ instead. In that case, it follows that $p = (s,t) \cdot q$ since s and t are tame and there thus cannot be any other path from s to t other than (s,t). Moreover, the path q has length $\operatorname{dist}_G(t,v)$ and any $q \neq (t,v)$ would therefore contain some vertex of $P_\alpha(t,v) \setminus \{t,v\}$. This would obviously contradict $tv \in A^\wedge_\alpha$, which we have just shown above. We may therefore conclude that $p = (s,t,v)$, which proves the path (s,t,v) to be unique.

It only remains to verify that v is the only vertex $x \in P_\alpha(s,v)$ that satisfies both $\operatorname{dist}^\wedge_G(s,x) = \operatorname{dist}_G(s,x)$ and $\operatorname{dist}^\vee_G(x,v) = \operatorname{dist}_G(x,v)$. Note that any such x necessarily satisfies $\alpha(x) \geq \alpha(v)$ and hence $x \notin D$. This means in particular, that v is the only such vertex, for they all belong to $P_\beta(s,v) = \{s,v\}$ by arguments completely analogous to the ones we employed above.

"(ii) ⇒ (i)": We choose to prove $sv \in A^\wedge_\beta$ and $P_\alpha(s,v) \neq \{s,v\}$, which is equivalent to condition (i) by № 4.15. Note that $t \in P_\alpha(s,v)$ and hence also $P_\alpha(s,v) \neq \{s,v\}$ follows immediately from our assumptions. It only remains to show $sv \in A^\wedge_\beta$. To this end, let us assume $P_\beta(s,v) \neq \{s,v\}$. We choose $x \in P_\beta(s,v) \setminus \{s,v\}$ such that $\alpha(x)$ is maximal. Note that any $x \in P_\beta(s,v)$ satisfies $\beta(x) \geq \beta(s)$ and is therefore an element of S or not contained in D at all. As above, it follows that α and β induce the same relative order on x and s, that is $\alpha(x) \geq \alpha(s)$, and hence $x \in P_\alpha(s,v)$.

By Theorem 2, there is some vertex $y \in P_\alpha(s,x)$, such that

$$\operatorname{dist}^\wedge_G(s,y) = \operatorname{dist}_G(s,y)\,, \qquad \operatorname{dist}^\vee_G(y,x) = \operatorname{dist}_G(y,x)\,.$$

Recall from above that either $x \neq s$ and $x \in S$ or $x \notin D$. Since there is a path from y to x in $G_\alpha^\vee$, it follows in both cases that $y \neq t$. As there is furthermore a path from s to y in $G_\alpha^\wedge$, it follows that $y \in S$ or $y \notin D$. Observe that α and β induce the same relative order on s and y in both cases. We thus find $\beta(y) > \beta(s)$ and hence $y \in P_\beta(s,x) \subseteq P_\beta(s,v)$. Our initial choice of x now implies $x = y$ since $\alpha(y) \geq \alpha(x)$ follows from the existence of a path from y to x in $G_\alpha^\vee$. This implies in particular that $\mathrm{dist}_G^\wedge(s,x) = \mathrm{dist}_G(s,x)$.

Let us now consider the vertices x and v. By Theorem 2 there exists some vertex $z \in P_\alpha(x,v)$ such that

$$\mathrm{dist}_G^\wedge(x,z) = \mathrm{dist}_G(x,z)\,, \qquad \mathrm{dist}_G^\vee(z,v) = \mathrm{dist}_G(z,v)\,,$$

which implies

$$\begin{aligned}\mathrm{dist}_G^\wedge(s,z) &\leq \mathrm{dist}_G^\wedge(s,x) + \mathrm{dist}_G^\wedge(x,z)\\ &= \mathrm{dist}_G(s,x) + \mathrm{dist}_G(x,z)\\ &= \mathrm{dist}_G(s,z)\,.\end{aligned}$$

Observe that any $z \in P_\alpha(x,v)$ is also contained in $P_\alpha(s,v)$ since $x \in P_\beta(s,v)$ lies on a shortest path from s to v and since we have already seen above that $x = y$ and $\alpha(y) \geq \alpha(s)$. Altogether, we have thus found a vertex $z \in P_\alpha(s,v)$ such that $\mathrm{dist}_G^\wedge(s,z) = \mathrm{dist}_G(s,z)$ and $\mathrm{dist}_G^\vee(z,v) = \mathrm{dist}_G(z,v)$. Our assumptions thus imply $z = v$. Note that x then lies on a path from s to v of length $\mathrm{dist}_G(s,v)$, which gives $x \in \{s,t,v\}$ by our assumptions. As $\beta(x) \geq \beta(s)$, we finally obtain $x \in \{s,v\}$, which contradicts our initial choice of x and therefore finishes this proof. □

Note that the statement of the above lemma is in fact suited for an actual computation of the arcs $A_{\beta\smallsetminus\alpha}^\wedge = A_\beta^\wedge \smallsetminus A_\alpha^\wedge$ that have to be inserted upon swapping s and t. Dijkstra's algorithm in contraction hierarchies may easily be modified, so that it does not only compute the distance of two vertices s and t but also the number of vertices x that satisfy $\mathrm{dist}_G^\wedge(s,x) + \mathrm{dist}_G^\vee(x,t) = \mathrm{dist}_G(s,t)$. Moreover, it is also possible to count for each such x the number of predecessors on shortest paths from s to x in $G_\alpha^\wedge$. Both these modifications can be implemented with constant overhead, so that condition (ii) of № 4.16 can actually be checked in time $\mathcal{O}(\mathsf{T}_{\text{query}})$. As one has to take each arc tv into consideration, the resulting Algorithm 4.4 has running time $\mathcal{O}(\deg(t) \cdot \mathsf{T}_{\text{query}})$. Likewise, there is an algorithm computing the arcs $A_{\beta\smallsetminus\alpha}^\vee = A_\beta^\vee \smallsetminus A_\alpha^\vee$ with running time $\mathcal{O}(\deg(t) \cdot \mathsf{T}_{\text{query}})$. Altogether, we have the following corollary.

4.17 Corollary: There is an algorithm that computes for a given contraction hierarchy G_α the sets $A_\beta^\wedge \smallsetminus A_\alpha^\wedge$ and $A_\beta^\vee \smallsetminus A_\alpha^\vee$ in time $\mathcal{O}\left(\deg(t) \cdot \mathsf{T}_{\text{query}}\right)$. †

Having finally mastered all the above technicalities, we merely have to assemble these results to obtain a complete description of G_β in terms of G_α.

Algorithm 4.4: Computation of the arcs of $G_\beta^\wedge$ that have to be inserted upon swapping a tame pair s and t in a contraction hierarchy G_α

Data: Contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$
Input: Tame pair s and t with $\alpha(s) < \alpha(t)$.
Output: Arcs $A^\wedge_{\beta\setminus\alpha} = A^\wedge_\beta \setminus A^\wedge_\alpha$, where β is the order obtained from α by swapping s and t.

1 $A^\wedge_{\beta\setminus\alpha} \leftarrow \{ts\}$ **if** $ts \in A^\vee_\alpha$ **else** $\varnothing$
2 **if** $st \in A^\wedge_\alpha$ **then**
3 **foreach** $tv \in A^\wedge_\alpha$ **do**
4 **if** *condition (ii) of № 4.16 is satisfied* **then** $A^\wedge_{\beta\setminus\alpha} \leftarrow A^\wedge_{\beta\setminus\alpha} \cup \{tv\}$
5 **return** $A^\wedge_{\beta\setminus\alpha}$

Theorem 9 *Swapping the relative order of a tame pair*
Let s and t be a tame pair with $\alpha(s) < \alpha(t)$ and let β be the order obtained from α by swapping s and t. Further let

$$\Delta_{st} = \{st\} \cap A^\wedge_\alpha \qquad\qquad \Delta_{ts} = \{ts\} \cap A^\vee_\alpha .$$

Then

$$A^\wedge_\beta = A^\wedge_\alpha \cup \left(\Delta_{ts} \cup \left\{ sv \;\middle|\; \begin{array}{l} (s,t,v) \text{ is a unique shortest } s\text{-}v\text{-path of} \\ \text{length } \mathrm{dist}_G(s,v) \text{ in } G^\wedge_\alpha \text{ and there is no} \\ x \in P_\alpha(s,v) \setminus \{v\} \text{ with } \mathrm{dist}^\wedge_G(s,x) = \mathrm{dist}_G(s,x) \\ \text{and } \mathrm{dist}^\vee_G(x,v) = \mathrm{dist}_G(x,v). \end{array} \right\} \right)$$
$$\setminus \left(\Delta_{st} \cup \left\{ tv \in A^\wedge_\alpha \;\middle|\; \begin{array}{l} \text{There is a path } (s,\dots,v) \text{ of length } \mathrm{dist}_G(s,v) \\ \text{in } G^\wedge_\alpha \text{ and } \mathrm{dist}_G(t,v) = \mathrm{dist}_G(t,s) + \mathrm{dist}_G(s,v). \end{array} \right\} \right)$$

and

$$A^\vee_\beta = A^\vee_\alpha \cup \left(\Delta_{st} \cup \left\{ vs \;\middle|\; \begin{array}{l} (v,t,s) \text{ is a unique shortest } v\text{-}s\text{-path of} \\ \text{length } \mathrm{dist}_G(v,s) \text{ in } G^\vee_\alpha \text{ and there is no} \\ x \in P_\alpha(v,s) \setminus \{v\} \text{ with } \mathrm{dist}^\wedge_G(v,x) = \mathrm{dist}_G(v,x) \\ \text{and } \mathrm{dist}^\vee_G(x,s) = \mathrm{dist}_G(x,s). \end{array} \right\} \right)$$
$$\setminus \left(\Delta_{ts} \cup \left\{ vt \in A^\vee_\alpha \;\middle|\; \begin{array}{l} \text{There is a path } (v,\dots,s) \text{ of length } \mathrm{dist}_G(v,s) \\ \text{in } G^\vee_\alpha \text{ and } \mathrm{dist}_G(v,t) = \mathrm{dist}_G(v,s) + \mathrm{dist}_G(s,t). \end{array} \right\} \right).$$

There is furthermore an algorithm computing this contraction hierarchy G_β from G_α in time $\mathcal{O}\left(\deg(t) \cdot \mathsf{T}_{\text{query}}\right)$. ‡

Proof: The above description of $A^\wedge_\beta \setminus A^\wedge_\alpha$ and $A^\vee_\beta \setminus A^\vee_\alpha$ follows from № 4.16. The characterisation of $A^\wedge_\alpha \setminus A^\wedge_\beta$ and $A^\vee_\alpha \setminus A^\vee_\beta$ follows from № 4.12 and № 4.13. The appended remark concerning an algorithm that computes G_β from G_α is just a summary of our results № 4.14 and № 4.17 concerning Algorithm 4.3 and Algorithm 4.4. ▫

The reader should note that the above theorem is nothing but a precise paraphrasing of our informal discussion at the very beginning of this paragraph. In the end, our intuition turned out to be true.

Given the above description of G_β in terms of G_α, we may now reason about G_β using variables in G_α only. It is in particular possible to determine for any given tame pair s and t of vertices, whether swapping the relative order of s and t results in a contraction hierarchy with smaller search space sizes. It remains to be determined experimentally which criteria, like maximum search space size or average search space size of vertices in $\mathcal{R}(s, G_\alpha^\wedge)$ and $\mathcal{R}(t, G_\alpha^\wedge) \setminus \mathcal{R}(s, G_\alpha^\wedge)$ are actually suited to be employed in such a postprocessing of contraction hierarchies. The above theorem may serve as a solid foundation for such practical algorithms.

Conclusion

In this thesis, we developed and justified a model of contraction hierarchies sufficiently sophisticated to capture actual applications like the graphs computed by the original algorithms of Geisberger et al. [GSSD08] but yet concise enough to lend itself to theoretical investigations as carried out in Chapter 3 and Chapter 4. In Chapter 3.1, we uncovered an unanticipated interrelation between contraction hierarchies and filled graphs that directly relates the search space size and space consumption of contraction hierarchies to the height of elimination trees and the number of fill-edges in filled graphs, respectively. These relations allow the transfer of a huge amount of both theoretical and practical research concerning filled graphs to the younger problem of constructing efficient contraction hierarchies. Employing previous results on a heuristic to compute elimination trees of low height known as nested dissection, we deduced in Chapter 3.2 and Chapter 3.3 upper bounds of $\mathcal{O}(\log(n) \cdot \mathrm{tw}(G))$ and $\mathcal{O}(\sqrt{n})$ on the maximum search space size of contraction hierarchies of graphs of bounded treewidth $\mathrm{tw}(G)$ and graphs belonging to minor-closed families of graphs with separators of size $\mathcal{O}(\sqrt{n})$, respectively. Moreover these bounds are accompanied by reasonable upper bounds on the space consumption of the corresponding contraction hierarchies and practical algorithms to construct these contraction hierarchies. These findings seem to be the first results on the performance of contraction hierarchies that do not depend on the edge lengths of the underlying graph. In Chapter 3.4, we subsequently compared our findings to previous work of Abraham et al. [AFGW10; ADFGW11] that incorporates the edge lengths of the underlying graph and argued that our approach does not perform worse if the edge lengths are only sufficiently ill-behaved. Moreover, these considerations also led to a a lower bound on a parameter called highway dimension in terms of pathwidth. Finally, in Chapter 4 we considered the problem of locally modifying contraction hierarchies. We obtained a complete description of the effect of swapping the relative order of the endpoints of specific arcs that may serve as a solid foundation for postprocessing contraction hierarchies.

Open Questions

The interrelation between filled graphs, elimination trees and contraction hierarchies we described in Chapter 3 raises more questions than can be answered in a single thesis. We investigated the implications only for a single heuristic for finding elimination trees of low height. However, there are other algorithms with other characteristics that might perform better in practise or theory. Moreover, there also remains one conceptual problem with all of our results concerned with the performance of distance queries. Although we deduced upper bounds on the search space size in contraction hierarchies, we were not able to find a satisfactory answer to the question of how search space size relates to the running time of distance queries. This is certainly a difficult question and an answer may involve modifications to both our model of contraction hierarchies or the actual query.

Furthermore, we were not concerned at all with the problem of minimising the space consumption of contraction hierarchies – an important issue when deploying mobile applications. As already indicated in Chapter 3.1, there is a vast amount of research and hands-on experience concerned with the equivalent problem for filled graphs. Our model of contraction hierarchies allows the transfer of this knowledge to contraction hierarchies – an endeavour we have just begun.

The upper bound on pathwidth in terms of highway dimension that we have proven in Chapter 3.4 also raises some questions. On the one hand, further investigation of the relation between pathwidth and highway dimension seems to be a worthwhile project. On the other hand, highway dimension is not only related to contraction hierarchies but to several other speedup techniques for Dijkstra's algorithm. It is not known whether any other speedup technique relates to the pathwidth or treewidth of the underlying graph. Any answer to these questions would greatly add to our understanding of road networks, for both highway dimension and speedup techniques are specifically targeted at road networks while pathwidth and treewidth are studied extensively and quite well understood.

However, there are also more specific questions to be answered. As already indicated at the end of Chapter 3.1, the investigation of the gap between maximum search space size and elimination tree height is an interesting and open problem. Bounding this gap would imply the existence of approximation algorithms for contraction hierarchies with optimal maximum search space size on a variety of graphs. Related to this question is the problem of finding lower bounds on the maximum search space size in contraction hierarchies. Last but not least, an actual implementation and experimental evaluation of our algorithms would also add to our understanding of the existing heuristics currently in use. To this end, we point out that the recent practical algorithm to compute balanced separators in road networks due to Delling et al. [DGRW11] facilitates the practical computation of nested dissection orders. Furthermore, postprocessing existing contraction hierarchies based on our investigations of tame pairs and their swapping presented in Chapter 4 and experimentally developing strong local optimality criteria also seems to be a project that might pay off in the

end.

Index of Notation

The following is an alphabetically sorted list of the most frequently used symbols including short explanations and pointers to their definition. If there is more than one definition, all of them are listed.

A

α	A bijective map $\alpha\colon V \to [n]$ assigning number to the vertices V of some graph G.	p. 15
A_v	The arcs of the graph $G = (V, A)$ after contraction of a single vertex v.	p. 13
A_i	The arcs of the graph $G = (V, A)$ after contraction of the first $(i-1)$ vertices as specified by an order α on its vertices.	p. 15
$A_\alpha^\wedge$	The arcs of a contraction hierarchy $\bar{G}_\alpha$ or G_α whose target vertex lies above its source vertex.	p. 19, 23
$A_\alpha^\vee$	The arcs of a contraction hierarchy $\bar{G}_\alpha$ or G_α whose target vertex lies below its source vertex.	p. 19, 23

B

$B_\alpha^\wedge$	The arcs of a weak contraction hierarchy H_α whose target vertex lies above its source vertex.	p. 38
$B_\alpha^\vee$	The arcs of a weak contraction hierarchy H_α or whose target vertex lies below its source vertex.	p. 38

D

$\mathrm{dist}_G(s,t)$	Distance between vertices u and v in a weighted graph G.	p. 5
$\mathrm{dist}_G^\wedge$(u,v)	The length of a shortest path from u to v in $\bar{G}_\alpha^\wedge$ or $G_\alpha^\wedge$.	p. 19, 23
$\mathrm{dist}_G^\vee$(u,v)	The length of a shortest path from u to v in $\bar{G}_\alpha^\vee$ or $G_\alpha^\vee$.	p. 19, 23
$\mathrm{dist}_H^\wedge(u,v)$	Distance between vertices u and v in the weak contraction hierarchy $H_\alpha^\wedge$.	p. 39
$\mathrm{dist}_H^\vee(u,v)$	Distance between vertices u and v in the weak contraction hierarchy $H_\alpha^\vee$.	p. 39

G

${}^*G$	The undirected graph obtained from a directed graph G by forgetting the direction of all the arcs of G.	p. 52
$G(v)$	The graph G after contraction of a single vertex v.	p. 13
$G(i)$	The graph G after contraction of the first $(i-1)$ vertices as specified by an order α on its vertices.	p. 15
$\bar{G}_\alpha = (\bar{G}^\wedge_\alpha, \bar{G}^\vee_\alpha)$	The contraction hierarchy of G with respect to the order α as defined by Geisberger. The graph $\bar{G}^\wedge_\alpha$ contains those arcs of $\bar{G}_\alpha$ that point upwards, while $\bar{G}^\vee_\alpha$ consists of the arcs pointing downwards.	p. 19
$G_\alpha = (G^\wedge_\alpha, G^\vee_\alpha)$	The contraction hierarchy of G with respect to the order α. The graph $G^\wedge_\alpha$ contains those arcs that point upwards, while $G^\vee_\alpha$ consists of the arcs pointing downwards.	p. 23
G^α	Chordal graph associated with an undirected graph G and an order α on its vertices by means of the elimination game.	p. 52

H

$H_\alpha = (H^\wedge_\alpha, H^\vee_\alpha)$	A pair of digraphs $H^\wedge_\alpha$ and $H^\vee_\alpha$ containing $A^\wedge_\alpha$ and $A^\vee_\alpha$ and satisfying some compatibility conditions.	p. 38
$\mathrm{ht}(G^\alpha)$	Height of the elimination tree associated with an undirected graph G and an order α on its vertices.	p. 55
$\mathrm{ht}(G)$	Minimum height among all elimination trees of a fixed, undirected graph $G = (V, E)$.	p. 55

L

$\mathrm{len}_G(uv)$	Length of the arc uv in a weighted digraph G.	p. 5
$\mathrm{len}^\alpha_G(uv)$	Length of the arc uv in the contraction hierarchy G_α of G.	p. 19, 23
$\mathrm{len}^\alpha_H(uv)$	Length of the arc uv in the weak contraction hierarchy H_α of G.	p. 39

M

M_α	The unique maximal weak contraction hierarchy of a digraph G with respect to a fixed order α.	p. 45

N

$[n]$	The set of integers $\{1, 2, \ldots, n\}$ linearly ordered by the usual less than or equal relation "$\leq$".	p. 15

P

$P_\alpha(s,t)$	Set of vertices that are above s or above t and that additionally lie on a shortest path from s to t.	p. 21

R

$\mathcal{R}(v,G)$	Set of vertices from which a vertex v is reachable in a directed acyclic graph G.	p. 34
$\#\mathcal{R}(v,G)$	Number of vertices from which a vertex v is reachable in a directed acyclic graph G.	p. 35

S

$\mathcal{S}(v,G)$	Set of vertices that are reachable from a vertex v in a directed acyclic graph G.	p. 34
$\#\mathcal{S}(v,G)$	Number of vertices that are reachable from a vertex v in a directed acyclic graph G.	p. 35
$\#\mathcal{S}(G)$	Arithmetic mean of the search space sizes of all vertices in a directed acyclic graph G.	p. 35
$\mathcal{S}_{\max}(G_\alpha)$	Maximum search space size among all vertices in a fixed contraction hierarchy G_α.	p. 35
$\mathcal{S}_{\text{avg}}(G_\alpha)$	Sum of the unidirectional average search space sizes $\#\mathcal{S}(G_\alpha^\wedge)$ and $\#\mathcal{S}(G_\alpha^\vee)$ of the contraction hierarchy $G_\alpha = (G_\alpha^\wedge, G_\alpha^\vee)$.	p. 37
$\mathcal{S}_{\max}(G)$	Minimum of $\mathcal{S}_{\max}(G,\alpha)$ among all contraction hierarchies G_α of a given graph G.	p. 35
$\text{scd}(uw)$	Measure of how many nested shortcuts are below a given arc uw of a contraction hierarchy G_α.	p. 25
$\sup(uw)$	Pair uv and vw of arcs that exist for any shortcut uw of a contraction hierarchy G_α.	p. 25

T

$T(G^\alpha)$	Rooted subtree of the filled graph G^α associated with an undirected graph G and an order α on its vertices.	p. 55
$\mathcal{T} = (\mathcal{X}, \mathcal{E})$	Decomposition of a graph G into a tree with nodes $\mathcal{X}$ and edges $\mathcal{E}$, such that each node $X \in \mathcal{X}$ is a subset of the vertices of G whose removal separates some specific subgraph of G.	p. 62
$\mathsf{T}_{\text{query}}$	Running time of the distance query Algorithm 2.3 in a given contraction hierarchy G_α.	p. 35

V

V_v	The vertices of the graph $G = (V, A)$ after contraction of a single vertex v.	p. 13

V_i	The vertices of the graph $G = (V, A)$ after contraction of the first $(i-1)$ vertices as specified by an order α on its vertices.	p. 15

Bibliography

[AD96] Lyudmil Aleksandrov and Hristo Djidjev. *Linear Algorithms for Partitioning Embedded Graphs of Bounded Genus*. In: *SIAM Journal on Discrete Mathematics* 9.1 (1996), pp. 129–150.

[ADFGW11] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg and Renato F. Werneck. *VC-Dimension and Shortest Path Algorithms*. In: *ICALP (1)*. Ed. by Luca Aceto, Monika Henzinger and Jiri Sgall. Vol. 6755. Lecture Notes in Computer Science. Springer, 2011, pp. 690–699.

[ADGW11] Ittai Abraham, Daniel Delling, Andrew V. Goldberg and Renato F. Werneck. *A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks*. In: *SEA*. Ed. by Panos M. Pardalos and Steffen Rebennack. Vol. 6630. Lecture Notes in Computer Science. Springer, 2011, pp. 230–241.

[ADGW12] Ittai Abraham, Daniel Delling, Andrew V. Goldberg and Renato F. Werneck. *Hierarchical Hub Labelings for Shortest Paths*. In: *ESA*. Ed. by Leah Epstein and Paolo Ferragina. Vol. 7501. Lecture Notes in Computer Science. Springer, 2012, pp. 24–35.

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg and Renato F. Werneck. *Highway Dimension, Shortest Paths, and Provably Efficient Algorithms*. In: *SODA*. Ed. by Moses Charikar. SIAM, 2010, pp. 782–793.

[AGU72] Alfred V. Aho, Michael R. Garey and Jeffrey D. Ullman. *The Transitive Reduction of a Directed Graph*. In: *SIAM Journal on Computing* 1.2 (1972), pp. 131–137.

[AH94] Bengt Aspvall and Pinar Heggernes. *Finding minimum height elimination trees for interval graphs in polynomial time*. In: *BIT Numerical Mathematics* 34.4 (1994), pp. 484–509.

[AMOT90] Ravindra K. Ahuja, Kurt Mehlhorn, James B. Orlin and Robert E. Tarjan. *Faster Algorithms for the Shortest Path Problem*. In: *Journal of the ACM* 37.2 (1990), pp. 213–223.

[BBRW12] Reinhard Bauer, Moritz Baum, Ignaz Rutter and Dorothea Wagner. *On the Complexity of Partitioning Graphs for Arc-Flags*. In: *ATMOS*. Ed. by Daniel Delling and Leo Liberti. Vol. 25. OASICS. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012, pp. 71–82.

[BCKKW10] Reinhard Bauer, Tobias Columbus, Bastian Katz, Marcus Krug and Dorothea Wagner. *Preprocessing Speed-Up Techniques Is Hard*. In: *CIAC*. Ed. by Tiziana Calamoneri and Josep Díaz. Vol. 6078. Lecture Notes in Computer Science. Springer, 2010, pp. 359–370.

[BDDW09] Reinhard Bauer, Gianlorenzo D'Angelo, Daniel Delling and Dorothea Wagner. *The Shortcut Problem - Complexity and Approximation*. In: *SOFSEM*. Ed. by Mogens Nielsen, Antonín Kucera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tuma and Frank D. Valencia. Vol. 5404. Lecture Notes in Computer Science. Springer, 2009, pp. 105–116.

[BDJKKMT98] Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller and Zsolt Tuza. *Rankings of Graphs*. In: *SIAM Journal on Discrete Mathematics* 11.1 (1998), pp. 168–181.

[BDSSSW10] Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdekker, Dominik Schultes and Dorothea Wagner. *Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm*. In: *ACM Journal of Experimental Algorithmics* 15 (2010).

[BGKH91] Hans L. Bodlaender, John R. Gilbert, Ton Kloks and Hjálmtyr Hafsteinsson. *Approximating Treewidth, Pathwidth, and Minimum Elimination Tree Height*. In: *WG*. Ed. by Gunther Schmidt and Rudolf Berghammer. Vol. 570. Lecture Notes in Computer Science. Springer, 1991, pp. 1–12.

[Col09] Tobias Columbus. *On the Complexity Of Contraction Hierarchies*. Student Thesis. Karlsruhe Institute of Technology, Institute for Theoretical Informatics, 2009.

[Der07] Dariusz Dereniowski. *Easy and hard instances of arc ranking in directed graphs*. In: *Discrete Applied Mathematics* 155.18 (2007), pp. 2601–2611.

[DGJ09] Camil Demetrescu, Andrew V. Goldberg and David S. Johnson. *The Shortest Path Problem: Ninth Dimacs Implementation Challenge*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 2009.

[DGRW11] Daniel Delling, Andrew V. Goldberg, Ilya Razenshteyn and Renato F. Werneck. *Graph Partitioning with Natural Cuts*. In: *IPDPS*. IEEE, 2011, pp. 1135–1146.

[Die06] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2006.

[Dij59] Edsger W. Dijkstra. *A note on two problems in connexion with graphs*. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[Dji96] Hristo Djidjev. *On-Line Algorithms for Shortest Path Problems on Planar Digraphs*. In: *WG*. Ed. by Fabrizio d'Amore, Paolo Giulio Franciosa and Alberto Marchetti-Spaccamela. Vol. 1197. Lecture Notes in Computer Science. Springer, 1996, pp. 151–165.

[DKKM94] Jitender S. Deogun, Ton Kloks, Dieter Kratsch and Haiko Müller. *On Vertex Ranking for Permutations and Other Graphs*. In: *STACS*. Ed. by Patrice Enjalbert, Ernst W. Mayr and Klaus W. Wagner. Vol. 775. Lecture Notes in Computer Science. Springer, 1994, pp. 747–758.

[DKKM99] Jitender S. Deogun, Ton Kloks, Dieter Kratsch and Haiko Müller. *On the Vertex Ranking Problem for Trapezoid, Circular-arc and Other Graphs*. In: *Discrete Applied Mathematics* 98.1–2 (1999), pp. 39–63. Revised version of *On Vertex Ranking for Permutations and Other Graphs*. In: *STACS*. Ed. by Patrice Enjalbert, Ernst W. Mayr and Klaus W. Wagner. Springer, 1994, pp. 747–758.

[DN06] Dariusz Dereniowski and Adam Nadolski. *Vertex rankings of chordal graphs and weighted trees*. In: *Information Processing Letters* 98.3 (2006), pp. 96–100.

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes and Dorothea Wagner. *Engineering Route Planning Algorithms*. In: *Algorithmics of Large and Complex Networks*. Ed. by Jürgen Lerner, Dorothea Wagner and Katharina Anna Zweig. Vol. 5515. Lecture Notes in Computer Science. Springer, 2009, pp. 117–139.

[EG08a] David Eppstein and Michael T. Goodrich. *Studying (Non-Planar) Road Networks Through an Algorithmic Lens*. In: *CoRR* abs/0808.3694 (2008). Expanded version of *Studying (non-planar) road networks through an algorithmic lens*. In: *GIS*. Ed. by Walid G. Aref, Mohamed F. Mokbel and Markus Schneider. ACM, 2008, p. 16.

[EG08b] David Eppstein and Michael T. Goodrich. *Studying (non-planar) road networks through an algorithmic lens*. In: *GIS*. Ed. by Walid G. Aref, Mohamed F. Mokbel and Markus Schneider. ACM, 2008, p. 16.

[FG65] Delbert R. Fulkerson and Oliver A. Gross. *Incidence matrices and interval graphs*. In: *Pacific Journal of Mathematics* 15.3 (1965), pp. 835–855.

[FHL05] Uriel Feige, Mohammad T. Hajiaghayi and James R. Lee. *Improved approximation algorithms for minimum-weight vertex separators*. In: *STOC*. Ed. by Harold N. Gabow and Ronald Fagin. ACM, 2005, pp. 563–572.

[FT87] Michael L. Fredman and Robert E. Tarjan. *Fibonacci heaps and their uses in improved network optimization algorithms*. In: *Journal of the ACM* 34.3 (1987), pp. 596–615.

[Gei07] Robert Geisberger. *Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks*. Diploma Thesis. Universität Karlsruhe, Institute for Theoretical Informatics, 2007.

[Geo73] Alan George. *Nested Dissection of a Regular Finite Element Mesh*. In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 345–363.

[GHT84] John R. Gilbert, Joan P. Hutchinson and Robert E. Tarjan. *A Separator Theorem for Graphs of Bounded Genus*. In: *Journal of Algorithms* 5.3 (1984), pp. 391–407.

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes and Daniel Delling. *Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks*. In: *WEA*. Ed. by Catherine C. McGeoch. Vol. 5038. Lecture Notes in Computer Science. Springer, 2008, pp. 319–333.

[GSSV12] Robert Geisberger, Peter Sanders, Dominik Schultes and Christian Vetter. *Exact Routing in Large Road Networks Using Contraction Hierarchies*. In: *Transportation Science* 46.3 (2012), pp. 388–404.

[GT86] John R. Gilbert and Robert E. Tarjan. *The analysis of a nested dissection algorithm*. In: *Numerische Mathematik* 50.4 (1986), pp. 377–404.

[Heg06] Pinar Heggernes. *Minimal triangulations of graphs: A survey*. In: *Discrete Mathematics* 306.3 (2006), pp. 297–317.

[Heg92] Pinar Heggernes. *Minimizing Fill-in Size and Elimination Tree Height in Parallel Cholesky Factorization*. Master's Thesis. University of Bergen, Department of Informatics, 1992.

[HP08] Pinar Heggernes and Barry W. Peyton. *Fast computation of minimal fill inside a given elimination ordering*. In: *SIAM Journal on Matrix Analysis and Applications* 30.4 (2008), pp. 1424–1444.

[HPS07] Chung-Hsien Hsu, Sheng-Lung Peng and Chong-Hui Shi. *Constructing a minimum height elimination tree of a tree in linear time*. In: *Information Sciences* 177.12 (2007), pp. 2473–2479.

[IRV88] Ananth V. Iyer, H. Donald Ratliff and Gopalakrishnan Vijayan. *Optimal node ranking of trees*. In: *Information Processing Letters* 28.5 (1988), pp. 225–229.

[KR10] Ken-ichi Kawarabayashi and Bruce A. Reed. *A Separator Theorem in Minor-Closed Classes*. In: *FOCS*. IEEE Computer Society, 2010, pp. 153–162.

[KT99] Jan Kratochvíl and Zsolt Tuza. *Rankings of Directed Graphs*. In: *SIAM Journal on Discrete Mathematics* 12.3 (1999), pp. 374–384.

[Liu90] Joseph W. H. Liu. *The role of elimination trees in sparse factorization*. In: *SIAM Journal on Matrix Analysis and Applications* 11.1 (1990), pp. 134–172.

[LT79] Richard J. Lipton and Robert E. Tarjan. *A Separator Theorem for Planar Graphs*. In: *SIAM Journal on Applied Mathematics* 36.2 (1979), pp. 177–189.

[Mad67] Wolfgang Mader. *Homomorphieeigenschaften und mittlere Kantendichte von Graphen*. In: *Mathematische Annalen* 174 (4 1967), pp. 265–268.

[MS12] Shay Mozes and Christian Sommer. *Exact distance oracles for planar graphs*. In: *SODA*. Ed. by Yuval Rabani. SIAM, 2012, pp. 209–222.

[Par61] Seymour V. Parter. *The Use of Linear Graphs in Gauss Elimination*. In: *SIAM Reviews* 3.2 (1961), pp. 119–130.

[Pey01] Barry W. Peyton. *Minimal Orderings Revisited*. In: *SIAM Journal on Matrix Analysis and Applications* 23.1 (2001), pp. 271–294.

[Pot88] Alex Pothen. *The Complexity of Optimal Elimination Trees*. Technical Report CS-88-13. Pennsylvania State University, 1988.

[RT78] Donald J. Rose and Robert E. Tarjan. *Algorithmic Aspects of Vertex Elimination on Directed Graphs*. In: *SIAM Journal on Applied Mathematics* 34.1 (1978), pp. 176–197.

[RW09] Bruce A. Reed and David R. Wood. *A linear-time algorithm to find a separator in a graph excluding a minor*. In: *ACM Transactions on Algorithms* 5.4 (2009).

[Sch82] Robert Schreiber. *A New Implementation of Sparse Gaussian Elimination*. In: *ACM Transactions on Mathematical Software* 8.3 (1982), pp. 256–276.

[Sch89] Alejandro A. Schäffer. *Optimal Node Ranking of Trees in Linear Time*. In: *Information Processing Letters* 33.2 (1989), pp. 91–96.

[Sim88] Klaus Simon. *An Improved Algorithm for Transitive Closure on Acyclic Digraphs*. In: *Theoretical Computer Science* 58 (1988), pp. 325–346.

[Som12] Christian Sommer. *Shortest-Path Queries in Static Networks*. Submitted to ACM Computing Surveys. 2012.

[Tho03] Mikkel Thorup. *Integer priority queues with decrease key in constant time and the single source shortest paths problem.* In: *STOC*. Ed. by Lawrence L. Larmore and Michel X. Goemans. ACM, 2003, pp. 149–158.

[Tho99] Mikkel Thorup. *Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time.* In: *Journal of the ACM* 46.3 (1999), pp. 362–394.

[Yan81] Mihalis Yannakakis. *Computing the Minimum Fill-In is NP-Complete.* In: *SIAM Journal on Algebraic and Discrete Methods* 2.1 (1981), pp. 77–79.