Theoretical Computer Science 645 (2016) 112–127

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Search-space size in contraction hierarchies ☆

Reinhard Bauer, Tobias Columbus, Ignaz Rutter *, Dorothea Wagner

Karlsruhe Institute of Technology, Germany

ARTICLE INFO

Article history:
Received 24 November 2015
Received in revised form 12 May 2016
Accepted 4 July 2016
Available online 15 July 2016
Communicated by F.V. Fomin

Keywords:
Shortest paths
Search space size
Contraction hierarchies
Speed-up techniques
Theoretical analysis

ABSTRACT

Contraction hierarchies are a speed-up technique to improve the performance of shortest-path computations, which works very well in practice. Despite convincing practical results, there is still a lack of theoretical explanation for this behavior.

In this paper, we develop a theoretical framework for studying search space sizes in contraction hierarchies. We prove the first bounds on the size of search spaces that depend solely on structural parameters of the input graph, that is, they are independent of the edge lengths. To achieve this, we establish a connection with the well-studied elimination game. Our bounds apply to graphs with treewidth k, and to any minor-closed class of graphs that admits small separators. For trees, we show that the maximum search space size can be minimized efficiently, and the average size can be approximated efficiently within a factor of 2.

We show that, under a worst-case assumption on the edge lengths, our bounds are comparable to those in the recent paper "VC-Dimension and Shortest Path Algorithms" of Abraham et al. [1], whose analysis depends also on the edge lengths. As a side result, we link their notion of highway dimension (a parameter that is conjectured to be small, but is unknown for all practical instances) with the notion of pathwidth. This is the first relation of highway dimension with a well-known graph parameter.

1. Introduction

Computing shortest paths in graphs is a fundamental problem in computer science with many applications, the most popular being the computation of routes in transportation networks. Though Dijkstra's algorithm can be used to efficiently compute shortest paths in $O(n \log n + m)$ time, this is too slow for many applications such as the computation of shortest paths in continental-size road networks. In the last decades substantial progress on improving this query time has been made. So-called speedup-techniques have led to speedups of a factor of one million or more; see [3] for an extensive survey. One particularly successful technique are contraction hierarchies (CH), which are widely used, especially since they have proven to be easily adaptable to various settings, such as time-dependent routes [4], multi-modal transportation networks [11], routing for electrical vehicles [7], shortest paths with multi-criteria objectives [14], and even fast all-pairs-shortest path computations [10].

☆ Partially supported by the DFG under grant WA 654/16-2. A preliminary version of this paper has appeared as R. Bauer, T. Columbus, I. Rutter, D. Wagner, Search Space Size in Contraction Hierarchies, in *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13), pages 93–104, volume 7965 of LNCS, 2013.*

* Corresponding author.
E-mail addresses: reinhard.bauer@kit.edu (R. Bauer), tobias.columbus@kit.edu (T. Columbus), ignaz.rutter@kit.edu (I. Rutter), dorothea.wagner@kit.edu (D. Wagner).

http://dx.doi.org/10.1016/j.tcs.2016.07.003
0304-3975/

Contraction hierarchies were introduced by Geisberger et al. [16], who evaluated their performance experimentally. The main idea of contraction hierarchies is the iterative *contraction* of nodes, where the contraction of a node v removes v from the graph and possibly inserts some shortcut edges between neighbors of v in order to preserve shortest-path distances (we present a brief description of contraction hierarchies in Section 2.2).

A common way to theoretically assess the performance of speed-up techniques is the maximum search space size, i.e., the maximum number of vertices a query has to process [6,5]. It turns out that the contraction hierarchy and also its performance in answering shortest-path queries depend strongly on the contraction order of the nodes. Finding a good node ordering that allows for fast shortest-path computations thus is an important problem. Practical implementations, such as the one by Geisberger et al. [16] employ heuristics for which no provable guarantees are known. Previous theoretical expositions rather focus on minimizing the size of the contraction hierarchy [6,21], i.e., the number of edges that are inserted during the contractions. In particular, it is known that minimizing the size of a contraction hierarchy is NP-complete. The only work providing provable performance guarantees for shortest-path computations in contraction hierarches, we are aware of, is the work of Abraham et al. [1,2]. They introduce the notion of highway dimension, a parameter that is conjectured to be small in real-world road networks, and prove sublinear query times under this assumption. However, the highway dimension of real-world instances is unknown, and may change as the length function changes. By contrast, we focus on providing bounds that rely on purely structural parameters of the graph, such as bounded treewidth or excluding a fixed minor. Our algorithms thus apply to classes of graphs that are defined purely by structural criteria, and our upper bounds are agnostic to the length function.

We note that theoretical results with better query times [28,13] exist, some of them even using similar techniques. They are, however, far from being practical. By contrast, our theoretical bounds apply to a widely used speed-up technique. It is also worth noting that recursive graph separation has been used as a heuristic in practical approaches [27], although, without providing theoretical guarantees. There have also been approaches for exploiting small treewidth for computing shortest paths. Chaudhuri and Zaroliagis [9] describe a data structure for answering shortest-path queries on graphs of small treewidth, which directly works on a tree decomposition. Planken et al. [23] apply a separator-based technique to speed up all-pairs-shortest-path computation in graphs of small treewidth. Recently, Dibbelt et al. [12] have followed our approach and demonstrated experimentally that it can be used for efficient customizable route planning. Very recently Funke and Storandt [15] have claimed provable guarantees on the search space for a randomized preprocessing of contraction hierarchies. However, they impose restrictions on the graph structure and the metric, and their query algorithm resembles a multi-level Dijsktra search rather than the query of a contraction hierarchy.

Contribution and outline We develop a theoretical framework for studying search-space sizes in contraction hierarchies. Due to the iterative definition of an algorithmic contraction hierarchy, it seems quite difficult to prove theoretical bounds, as this appears to inherently require arguments that are based on a local construction only. We overcome this drawback by giving a global description of the contraction hierarchy associated with a node ordering in Section 3.

Afterwards, in Section 4, we establish a connection between contraction hierarchies and two classical problems that have been widely studied. Namely, so-called filled graphs, which were introduced by Parter [22] in his analysis of Gaussian elimination, and elimination trees, which were introduced by Schreiber [26] for Gaussian elimination on sparse matrices. For trees, these connections in particular imply efficient algorithms that minimize the maximum search space size and approximate the average search space size within a factor of 2. This contrasts corresponding hardness results for other speed-up techniques, such as arcflags, where even processing trees optimally is NP-complete [5].

In Section 5, we show that nested dissection, a technique for finding elimination trees of small height, can be applied to construct orders α with provable bounds on the maximum search space size. For graphs of treewidth k and for graphs that admit small separators and exclude a fixed minor, we obtain maximum search space size $O(k \log n)$ and $O(\sqrt{n})$, respectively.

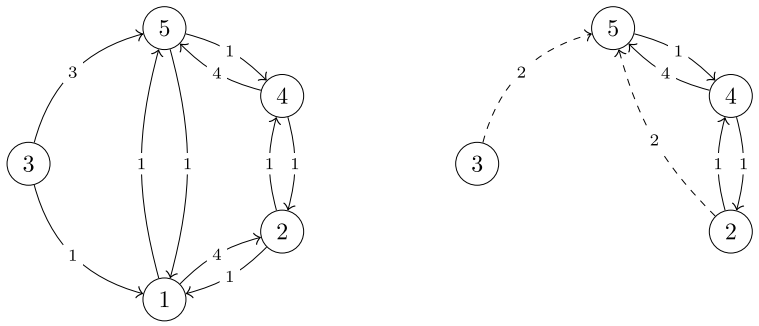
Finally, we compare our results with the results of Abraham et al. [1,2] in Section 7. If the length function is such that the highway dimension is maximal, then our results are comparable to theirs. However, our approach neither requires small maximum degree, nor does it depend on the diameter of the graph, and thus applies to a larger class of graphs. As a side result, we find an unanticipated and novel connection between highway dimension and pathwidth. This is, to our knowledge, the first relation between highway dimension and a more widely known graph parameter.

2. Preliminaries

In this section we collect preliminary notation and terminology that is used throughout the paper.

2.1. Graphs, paths, and lengths

A graph $G = (V, E)$ consists of a set V of vertices and an edge set E, where each edge $e \in E$ connects two vertices in V. In a directed graph, we use uv to denote the edge with source u and target v. In an undirected graph, we do not distinguish the source and target of an edge, and uv and vu denote the same edge. All graphs we consider are simple, i.e., they contain neither parallel edges nor loops. A weighted graph $G = (V, E, \text{len})$ is a graph with a weight function $\text{len}\colon E \to \mathbb{R}_{\geq 0}$. In case of ambiguity, we use a subscript to indicate to which graph a weight function refers, e.g., len_G for the weight function of graph G.



(a) A weighted directed graph G. (b) The graph G after contraction of 1.

Fig. 1. Contraction of a single vertex. Fig. 1a shows a directed graph G on five vertices with arc lengths drawn on the respective arcs. In Fig. 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.

A path p in a (directed) graph $G = (V, E)$ is a sequence $p = (v_1, \dots, v_t)$ of vertices such that $v_i \in V$ and $v_i v_{i+1} \in E$ for $i = 1, \dots, t-1$. We say that p is a path from v_1 to v_t. The length of p is $\text{len}(p) = \sum_{i=1}^{t-1} \text{len}(v_i v_{i+1})$. We use $V(p) = \{v_1, \dots, v_t\}$ and $E(p) = \{v_i v_{i+1} \mid i = 1, \dots, t-1\}$ to denote the vertices and edges of p, respectively. The distance $\text{dist}_G(s, t)$ between two vertices s and t in G is the length of a shortest path from s to t in G, if such a path exists, and ∞ otherwise.

2.2. Contraction hierarchies

Given a weighted directed graph $G = (V, E, \text{len})$ and a vertex $v \in V$, *contraction of* v means

(i) removing v, and
(ii) inserting a shortcut uw of length $\text{dist}_G(u, w)$ for each unique shortest path (u, v, w) in G.

As can also be seen in the example in Fig. 1, if in step (ii) the edge uw already exists, it is replaced by the new edge.

Given an order $\alpha\colon V \to \{1, \dots, n\}$ of the vertices of $G = (V, E)$, a *contraction hierarchy* $C_{\text{algo}}(G, \alpha)$ consisting of two digraphs $C^\wedge$ and $C^\vee$ is constructed as follows: One iteratively contracts the vertices in the order specified by α. Let E' denote the set of shortcuts that is created in this process. Then $C^\wedge = (V, A^\wedge)$ and $C^\vee = (V, A^\vee)$, where $A^\wedge = \{uv \in E \cup E' \mid \alpha(u) < \alpha(v)\}$ and $A^\vee = \{uv \in E \cup E' \mid \alpha(v) < \alpha(u)\}$. In the following, we refer to such a contraction hierarchy as an *algorithmic contraction hierarchy*.

The distance query on an algorithmic contraction hierarchy C works as follows: Given a pair $s, t \in V$ of vertices, one employs Dijkstra's algorithm to compute the distances $\text{dist}_{C^\wedge}(s, v)$ and $\text{dist}_{C^\vee}(v, t)$ for all $v \in V$. The distance between s and t is then given by $\min_{v \in V} \text{dist}_{C^\wedge}(s, v) + \text{dist}_{C^\vee}(v, t)$. The correctness of this query relies on the following proposition, which is due to Geisberger et al. [16].

Proposition 1. $\text{dist}_G(s, t) = \min_{v \in V} \text{dist}_{C^\wedge}(s, v) + \text{dist}_{C^\vee}(v, t)$ *for all* $s, t \in V$.

Recall that Dijkstra's algorithm keeps a priority queue of discovered vertices sorted by their tentative distance from the source vertex s. In each step, the vertex u with least tentative distance is removed from the queue, its actual distance is set to the current tentative distance, and the tentative distance of all its neighbors is updated according to the length of its outgoing edges. We refer to this step as *settling* u.

The performance of the above distance query on contraction hierarchies obviously depends on both the number of vertices and the number of arcs that the two runs of Dijkstra's algorithm have to take into consideration. We focus on the number of vertices and call the set of vertices that are settled during Dijkstra's algorithm starting from s in $C^\wedge$ the *search space* of s. Similarly, we call the set of vertices that are settled during Dijkstra's algorithm starting from t in $C^\vee$ the *reverse search space* of t. Here, we consider Dijkstra's algorithm without halting criterion as we are only interested in upper bounds on the search space size anyway.

Nevertheless, bounds on the search space size can be used to bound the query time. Namely, if the maximum search space size is k, then of course a search space may induce at most k^2 arcs, and therefore the query takes time $O(k^2)$. Hence, as long as $k = o(\sqrt{n \log n})$, this yields a speed-up over Dijkstra's algorithm.

3. A formal model of contraction hierarchies

In this section, we develop a theoretical model of contraction hierarchies that is simpler to work with than algorithmic contraction hierarchies. Fix a directed graph $G = (V, E)$ and an ordering α of the vertices V of G. Let

$$P_\alpha(s,t) = \left\{ v \in V \;\middle|\; \begin{array}{l} \alpha(v) \geq \min\{\alpha(s), \alpha(t)\} \text{ and} \\ \operatorname{dist}_G(s,v) + \operatorname{dist}_G(v,t) = \operatorname{dist}_G(s,t) < \infty \end{array} \right\}.$$

That is, $P_\alpha(s,t)$ contains the vertices that lie on a shortest st-path and above at least one of s and t. The reader should note that $P_\alpha(s,t)$ is empty if and only if there are no st-paths in G. The following theorem provides a global characterization of the algorithmic contraction hierarchy.

Theorem 1. *Let $G = (V, A)$ be a weighted digraph, α an order of its vertices and $C_{\text{algo}}(G,\alpha) = (C^\wedge, C^\vee)$ the associated algorithmic contraction hierarchy. The arcs $A^\wedge$ and $A^\vee$ of $C^\wedge$ and $C^\vee$ are*

$$A^\wedge = \{uv \in A \mid \alpha(u) < \alpha(v)\} \cup \{uv \mid \alpha(u) < \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\}\}$$
$$A^\vee = \{uv \in A \mid \alpha(u) > \alpha(v)\} \cup \{uv \mid \alpha(u) > \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\}\}.$$

The length of a shortcut uv in $C^\wedge$ or $C^\vee$ is $\operatorname{dist}_G(u,v)$.

Before we can prove this theorem, we need a more detailed description of the construction of an algorithmic contraction hierarchy. Let $G = (V, E)$ be a graph and $v \in V$. We denote the graph that is obtained from G by contracting v by $G(v) = (V_v, A_v)$. Given a node order $\alpha\colon V \to \{1,\dots,n\}$, we denote the graph G after the $(i-1)$th contraction by $G(i) = (V_i, A_i)$, i.e., $G(1) = G$, and $G(i) = G(i-1)(v)$, where $v = \alpha^{-1}(i-1)$, for $1 < i \leq n$. We have the following lemma.

Lemma 1. *Let s and t be two vertices and let $k = \min\{\alpha(s), \alpha(t)\}$.*

(a) *For any st-path p in $G(k)$, there exists a sequence of st-paths $p_k, p_{k-1}, \dots, p_1$, such that (i) $p_k = p$, (ii) p_i is a path in $G(i)$, and (iii) the length of each p_i equals the length of p.*
(b) *For any shortest st-path p in G, there exists a sequence of shortest st-paths $p_1, p_2, \dots, p_k$, such that (i) $p_1 = p$, (ii) p_i is a path in $G(i)$ and $V(p) \cap V_i \subseteq V(p_i)$, and (iii) the length of each p_i equals the length of p.*

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

Proof. We start with (a). Beginning with $p_k = p$, we inductively construct the paths p_i as claimed. Suppose that $p_k, \dots, p_i$ 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 cases based on whether p_i contains a shortcut in S or not.

$E(p_i) \cap S = \emptyset$: If the path p_i in $G(i)$ 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 $\operatorname{len}_{G(i-1)}(p_{i-1}) = \operatorname{len}_{G(i)}(p_i)$ since during passage from $G(i-1)$ to $G(i)$ only the arc lengths of shortcuts change.

$E(p_i) \cap S \neq \emptyset$: 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

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

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 hence follows that $\operatorname{len}_{G(i-1)}(p_{i-1}) = \operatorname{len}_{G(i)}(p_i)$.

For statement (b), first observe that according to (a), all st-paths in $G(i)$ induce st-paths 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 $\operatorname{len}_G(p)$. In order to establish that p_i is a shortest path in $G(i)$ of length $\operatorname{len}_G(p)$ it therefore suffices to show that $\operatorname{len}_{G(i)}(p_i) \leq \operatorname{len}_G(p)$. This observation will be used several times in the following.

Starting with $p_1 = p$, we inductively construct the shortest paths p_i in $G(i)$. Suppose that $p_1, \dots, p_i$ 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.

1) *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 $V(p) \cap V_i \subseteq V(p_i)$ immediately yields $V(p) \cap V_{i+1} \subseteq V(p_i)$. Since arc lengths do not grow during contraction, we additionally have

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

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

2) p_i **contains a unique shortest path** (u, v, w) **as subpath:**
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 $V(p_{i+1}) = V(p) \cap V_{i+1}$ follows from $V(p_i) = V(p) \cap V_i$.
3) p_i **contains a non-unique shortest path** (u, v, w) **as subpath:**
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 w of the same length. However, there is no guarantee that this path q does not contain the vertex v. Nevertheless, if one chooses the shortest uw-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 a shortest path from u to w 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 a subpath. If (x, v, y) is not unique, i.e. if there is another shortest xy-path r, then r consists of at least three arcs since $r \neq (x, v, y)$, and since a shortest path (x, z, y) for some $z \neq v$ would yield a shortest path avoiding v. If one replaces the subpath (x, v, y) of q with the path r, one obtains a shortest path from u to w that contains more arcs than q. 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 obtains a path p_i' with $v \notin p_i'$ or a path p_i' containing a unique shortest path (x, v, y) as a subpath. In either case, our previous arguments show that there is a shortest st-path p_{i+1} in $G(i+1)$ 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 $V(p) \cap V_{i+1} \subseteq V(p_{i+1})$ follows from $V(p) \cap V_i \subseteq V(p_i)$. Altogether, we see that p_{i+1} is in fact the desired st-path in $G(i+1)$. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. Recall that $A^\wedge \cup A^\vee$ is a partition of all the arcs $\bigcup_i A_i$ occurring in one of the intermediate contractions $G(i) = (V_i, A_i)$. Since $A = A_1$, it follows from the definition of $A^\wedge$ and $A^\vee$ that

$$\left\{ uv \in A \,\middle|\, \alpha(u) < \alpha(v) \right\} \subseteq A^\wedge \qquad \text{and} \qquad \left\{ uv \in A \,\middle|\, \alpha(u) > \alpha(v) \right\} \subseteq A^\vee .$$

Furthermore, any arc $st \in A^\wedge \setminus A$ or $st \in A^\vee \setminus A$ 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 $P_\alpha(s,t) = \{s,t\}$.

Let s and t be two vertices with $\text{dist}_G(s,t) < \infty$, 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 Lemma 1 that there is for each $i \in \{1, \ldots, k\}$ 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, which hence are 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 $k = \min\{\alpha(s), \alpha(t)\}$. We further choose a shortest st-path p in G. By Lemma 1(b) there exists a shortest st-path p_k in $G(k)$ of length $\text{len}_G(p)$. 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 Lemma 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, Lemma 1 implies that the length of st in G_α is equal to $\text{dist}_G(s,t)$ as $p_k = (s,t)$ has the same length as the shortest path p from s to t in G. □

Not only does this theorem shed some light on the structure of algorithmic contraction hierarchies, but it also suggests an alternative definition. Consider an arc st of G that is no unique shortest path. Then, removing st from G does not change any distances. If st is a unique shortest path, then $P_\alpha(s,t) = \{s,t\}$ anyway. Thus the following definition of a contraction hierarchy works equally well. For a weighted digraph $G = (V, E, \text{len})$ and an order α of its vertices, we define $C(G, \alpha)$ to consist of two digraphs $C^\wedge$ and $C^\vee$, where $C^\wedge = (V, A^\wedge)$ and $C^\vee = (V, A^\vee)$ with

$$A^\wedge = \{uv \mid \alpha(u) < \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\}\}$$
$$A^\vee = \{uv \mid \alpha(u) > \alpha(v) \text{ and } P_\alpha(u,v) = \{u,v\}\}.$$

As in Theorem 1, we set $\text{len}_{C^\wedge}(uv) = \text{dist}_G(u,v)$ and $\text{len}_{C^\vee}(uv) = \text{dist}_G(u,v)$. We call the pair $C = C(G,\alpha) = (C^\wedge, C^\vee)$ a *formal contraction hierarchy*. As the order α will always be clear from the context, we mostly suppress it from our notation

and reserve the letter C for formal contraction hierarchies. Moreover, we always denote the arcs of $C^\wedge$ by $A^\wedge$ and those of $C^\vee$ by $A^\vee$. Finally, we usually write $\text{len}_C(uv)$ to denote either $\text{len}_{C^\vee}(uv)$ or $\text{len}_{C^\wedge}(uv)$.

We remark that if H is the digraph obtained from G by reversing all arcs and if $C = C(G,\alpha)$ and $D = C(H,\alpha)$ are the respective contraction hierarchies of G and H, then $C^\wedge = D^\vee$ and $C^\vee = D^\wedge$. This allows us to prove statements about C by only considering $C^\wedge$ since the analogous statement for $C^\vee$ follows by reversing all arcs.

We now carry over several useful concepts from the algorithmic definition. A *shortcut* is an arc uw of C that is not contained in G, or for which $\text{len}_G(uw) > \text{dist}_G(u,w)$. Note that the latter type of shortcuts are included only to model the possible overwriting of arc lengths in algorithmic contraction hierarchies. Given a shortcut uw in $A^\wedge$ or $A^\vee$, we are able to recover the vertex v that would have caused the insertion of uw in the corresponding algorithmic contraction hierarchy. Namely, let $S = \{v \in V \setminus \{u,w\} \mid \text{dist}_G(u,v) + \text{dist}_G(v,w) = \text{dist}_G(u,w)\}$. Note that $S \neq \emptyset$, for otherwise uw would be no shortcut. Note further that $\alpha(v) < \alpha(u)$ and $\alpha(v) < \alpha(w)$ for all $v \in S$ since otherwise we would have $v \in P_\alpha(u,w) = \{u,w\}$. We call the vertex $v \in S$ with $\alpha(v)$ maximal the *supporting vertex of* uw. It is easy to show that exactly the contraction of v causes the insertion of uw in the algorithmic contraction hierarchy $C_{\text{algo}}(G,\alpha)$.

Lemma 2. *Let uw be a shortcut of $C = C(G,\alpha)$ and let v be its supporting vertex. Then $C^\vee$ contains the arc uv and $C^\wedge$ contains the arc vw.*

Proof. It suffices to prove the existence of $uv \in A^\vee$, for the existence of $vw \in A^\wedge$ then follows by symmetry. Let us assume for the sake of contradiction that $uv \notin A^\vee$. By construction of the supporting vertex v, it is $\text{dist}_G(u,v) < \infty$ and $P_\alpha(u,v)$ is hence non-empty and contains $\{u,v\}$ as a subset. Moreover, $\alpha(u) > \alpha(v)$ since v is the supporting vertex of uw. As $uv \notin A^\vee$, 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

$$\text{dist}_G(u,v) = \text{dist}_G(u,x) + \text{dist}_G(x,v).$$

Moreover, it follows from $\alpha(x) > \alpha(v)$ and the fact that the supporting vertex v maximizes α among those vertices lying on a shortest uw-path that no such shortest path contains the vertex x, i.e., that

$$\text{dist}_G(u,w) = \text{dist}_G(u,v) + \text{dist}_G(v,w)$$
$$\text{and } \text{dist}_G(u,w) < \text{dist}_G(u,x) + \text{dist}_G(x,w).$$

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

$$\begin{aligned}\text{dist}_G(u,w) &= \text{dist}_G(u,v) + \text{dist}_G(v,w) = \text{dist}_G(u,x) + \text{dist}_G(x,v) + \text{dist}_G(v,w)\\ &\geq \text{dist}_G(u,x) + \text{dist}_G(x,w) > \text{dist}_G(u,w),\end{aligned}$$

which is obviously contradictory. □

We call the arcs, whose existence is guaranteed by Lemma 2, the *supporting arcs* of uw, and write $\sup(uw) = (uv, vw)$. Observe that if uv is a supporting arc of uw, then $\alpha(v) < \alpha(w)$, and thus chains of supporting arcs in $C^\wedge$ and $C^\vee$ are acyclic and do not descend indefinitely. This allows us to perform induction on the depth of the nested shortcuts supporting a given arc uw. We define the *shortcut depth* $\text{scd}(uw)$ of an arc of a formal contraction hierarchy C by $\text{scd}(uw) = 1$ if uw is no shortcut, and by $\text{scd}(uw) = \text{scd}(uv) + \text{scd}(vw)$ if uw is a shortcut with $\sup(uw) = (uv, vw)$. It is readily seen that, for a shortcut uw with $\sup(uw) = (uv, vw)$, we have $\text{len}_C(uw) = \text{len}_C(uv) + \text{len}_C(vw)$. Hence $C(G,\alpha)$ still possesses the most essential properties of the algorithmic contraction hierarchy $C_{\text{algo}}(G,\alpha)$.

A close look at the proof of Proposition 1 reveals that it merely depends on the fact that each arc uw with $P_\alpha(u,w) = \{u,w\}$ is contained in $C_{\text{algo}}(G,\alpha)$ and has length $\text{len}_C(uw) = \text{dist}_G(u,w)$. Thus Proposition 1 also holds for formal contraction hierarchies, implying that a bidirectional variant of Dijkstra's algorithm on the contraction hierarchy $C(G,\alpha)$ can be used to compute shortest paths in G.

To measure the performance of such computations on $C = C(G,\alpha)$, we define the search space of a query from s to t in a directed graph $H = (V,A)$ as $S(s,H) = \{u \in V \mid \text{dist}_H(s,u) < \infty\}$ and $R(t,H) = \{u \in V \mid \text{dist}_H(u,t) < \infty\}$. We use $S_{\max}(H) = \max_{s \in V} |S(s,H)|$ to denote the maximum search space size in H. Clearly, the shortest-path query from s to t in $C(G,\alpha)$ settles at most the vertices in $S(s,C^\wedge) \cup R(t,C^\vee)$. To maximize the performance of query algorithms, one is interested in an ordering α that minimizes $\max_{s,t \in V} |S(s,C^\wedge)| + |R(t,C^\vee)|$. To simplify the analysis, we rather concentrate on minimizing the *maximum search space size* $S_{\max}(C) = \max\{|S(v,C^\wedge)|, |R(v,C^\vee)|\}$. Note that

$$S_{\max}(C) \leq \max_{s,t \in V} |S(s,C^\wedge)| + |R(t,C^\vee)| \leq 2 \cdot S_{\max}(C),$$

and therefore optimizing $S_{\max}(C)$ gives asymptotically tight bounds for the number of settled vertices. In particular, bounding $S_{\max}(C)$ gives a guarantee on the query performance in terms of the number of settled nodes. Note that $S_{\max}$ is defined both for digraphs and contraction hierarchies. Similarly, we define the *average search space size* $S_{\text{avg}}(C) = 1/n \cdot \sum_{v \in V} |S(v,C^\wedge)| + |R(v,C^\vee)|$.

There is still one downside of formal contraction hierarchies: Practical implementations of contraction hierarchies do not compute an actual formal (or even algorithmic) contraction hierarchy. Instead of inserting a shortcut only when it is strictly necessary, fast heuristics are used to quickly exclude the necessity of a shortcut in many cases. In some cases, this results in the addition of shortcuts that are not necessary. Thus bounding $S_{\max}(C)$ may not have any practical implications since the additional shortcuts might increase the search space arbitrarily. To overcome this downside, we introduce one final type of contraction hierarchies, which also allow for additional shortcuts, yet preserve the properties that have turned out to be key to contraction hierarchies.

A *weak contraction hierarchy* W of a weighted digraph $G = (V, A, \text{len})$ with respect to some order α on the vertices V of G is a pair $(W^\wedge, W^\vee)$ of digraphs $W^\wedge = (V, B^\wedge)$ and $W^\vee = (V, B^\vee)$, such that the following conditions are satisfied.

(w1) $C(G, \alpha) \subseteq W$
(w2) $\alpha(u) < \alpha(v)$ for each $uv \in B^\wedge$ and each $vu \in B^\vee$
(w3) If uw is an arc of W that is not contained in G, then there is at least one pair of arcs $uv \in B^\vee$ and $vw \in B^\wedge$.

In the remainder of this section, we indicate how to extend our previous findings for contraction hierarchies to weak contraction hierarchies and investigate the relationship between different weak contraction hierarchies for the same ordering α. For this purpose, we fix a weighted digraph G and an ordering α of its vertices. As usual, we denote its formal contraction hierarchy by $C = C(G, \alpha) = (C^\wedge, C^\vee)$. Additionally, we fix a weak contraction hierarchy $W = (W^\wedge, W^\vee)$, whose arcs we denote by $B^\wedge$ and $B^\vee$, as above.

The notions of shortcuts and shortcut depth carry over literally to W. It follows immediately from (w1) and (w3) that for each shortcut uw in W, there is a pair of supporting arcs $uv \in B^\vee$ and $vw \in B^\wedge$. Although distances and arc lengths are only of secondary importance in the remaining sections, we still want to point out that it is not hard to give a "correct" definition of arc lengths on W, such that the following lemma holds true.

Lemma 3. *Let W be a weak contraction hierarchy.*

(a) $\text{len}_W(uw) \geq \text{dist}_G(u, w)$ *for all arcs uw of W.*
(b) $\text{len}_W(uw) = \text{dist}_G(u, w)$ *for all arcs uw of W with $P_\alpha(u, w) = \{u, w\}$.*

The "correct" definition of the arc lengths is given by

$$\text{len}_W(uw) = \min_{uv \in B^\vee, vw \in B^\wedge} \text{len}_W(uv) + \text{len}_W(vw) \quad \text{if } uw \text{ is a shortcut,}$$

and by

$$\text{len}_W(uw) = \text{len}_G(uw), \text{otherwise.}$$

Proof of Lemma 3. If $\text{scd}(uw) = 1$, then uw is an arc of G and $\text{len}_G(uw) = \text{dist}_G(u, w)$. Given our definition of $\text{len}_W(-)$, it is obvious that $\text{len}_W(uw)$ possesses both claimed properties.

If $\text{scd}(uw) > 1$, then uw is a shortcut of W and its length equals $\text{len}_W(uv) + \text{len}_W(vw)$, where $(uv, vw) = \sup(uw)$. By the triangle inequality and the induction hypothesis, we thus find $\text{len}_W(uw) \geq \text{dist}_G(u, w)$.

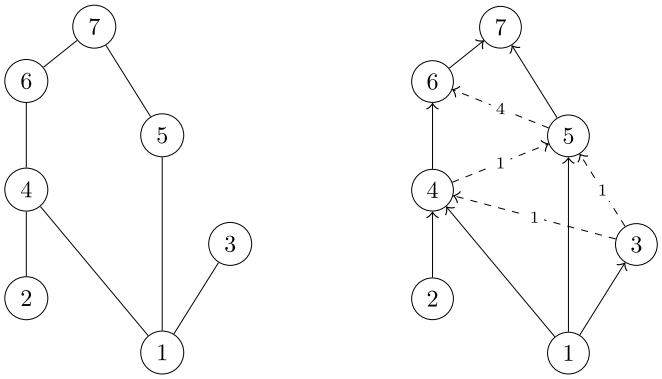
Let us now suppose $P_\alpha(u, w) = \{u, w\}$. In this case uw is an arc of $C = C(G, \alpha)$ and uw has supporting arcs $\sup(uw) = (uv, vw)$ in $C \subseteq W$. It follows from the induction hypothesis that $\text{len}_W(uv) = \text{dist}_G(u, v)$ and $\text{len}_W(vw) = \text{dist}_G(v, w)$. Employing these equalities in the definition of $\text{len}_W(-)$ finally gives

$$\text{len}_W(uw) \leq \text{dist}_G(u, v) + \text{dist}_G(v, w) = \text{dist}_G(u, w),$$

where the right-hand equality follows from the definition of the supporting vertex v in C. □

As indicated above, the proof of Proposition 1 for the algorithmic contraction hierarchy $C_{\text{algo}}(G, \alpha)$ relies on exactly the containment of the formal contraction hierarchy $C(G, \alpha)$ in $C_{\text{algo}}(G, \alpha)$ and the properties guaranteed by Lemma 3. Proposition 1 thus holds also for any weak contraction hierarchy. In particular, the same shortest-path algorithm works for weak contraction hierarchies.

In view of property (w1) it is clear that $C = C(G, \alpha)$ is the smallest weak contraction hierarchy. Moreover, if W_0 and W_1 are weak contraction hierarchies with respect to the same order α, then $W_0 \cup W_1$ is a weak contraction hierarchy. Thus, there exists a unique maximal weak contraction hierarchy, which we denote by $M = M(G, \alpha)$. Again, we reserve the letter M for the maximal weak contraction hierarchy and suppress the order α from our notation as there is little risk of confusion. It is not difficult to see that $S(u, C^\wedge) \subseteq S(u, W^\wedge) \subseteq S(u, M^\wedge)$, and symmetrically $R(u, C^\vee) \subseteq R(u, W^\vee) \subseteq R(u, M^\vee)$ for all weak contraction hierarchies W. In particular, this shows that $S_{\max}(C) \leq S_{\max}(W) \leq S_{\max}(M)$ for all weak contraction hierarchies W. Thus, we will concentrate on bounding $S_{\max}(M)$ in the following sections. Before we do so, we give a more explicit description of M.



(a) An undirected graph $G = (V, E)$ with an order α on its vertices. The values $\alpha(v)$ are drawn inside the vertices of G.

(b) The filled graph $F = F(G, \alpha)$ associated with the graph G and the order α. Fill edges are drawn dashed and labelled with $\alpha(v)$, where v is the vertex whose elimination led to their insertion.

Fig. 2. A graph G and an associated filled graph $F = F(G, \alpha)$.

Lemma 4. *A weak contraction hierarchy W is maximal if and only if W satisfies the following properties.*

(i) *Each arc of G is contained in W.*
(ii) *For any two arcs uv in $W^\vee$ and vw in $W^\wedge$, W also contains uw.*

Proof. It is clear that a maximal contraction hierarchy M satisfies both these conditions. On the other hand, let us consider any weak contraction hierarchy W satisfying (i) and (ii). We want to show that W is in fact nothing but M, i.e., that W contains any other weak contraction hierarchy W'. To this end, we do induction on the shortcut depth $\mathrm{scd}(uw)$ of the arcs of W'. If $\mathrm{scd}(uw) = 1$, then uw is an arc of G, which is contained in W by condition (i).

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

Note that this immediately implies an efficient way to construct the arcs of M by inserting shortcuts between each pair of neighbors during the contraction. In particular, the structure of M is independent of the weights on G.

Finally, we note that the query performance does not solely depend on the number of vertices in the search space, but also on the number of arcs. For a search space $S(u, M^\wedge)$, this number is certainly bounded by $|S(u, M^\wedge)|^2$. Moreover, the size $|M|$ has a crude upper bound in terms of $S_{\max}(M)$.

Lemma 5. *For any n-vertex directed graph with an ordering α of its vertices, we have $|M| \leq 2n \cdot S_{\max}(M)$.*

Proof. The target v of any arc uv of $M^\wedge$ is certainly contained in the search space $S(u, M^\wedge)$. Likewise, u is contained in $R(v, M^\vee)$ for any arc uv of $M^\vee$. The number of arcs in M is hence bounded from above by

$$\sum_{u \in V} |S(u, M^\wedge)| + |R(u, M^\vee)| \leq 2n \cdot S_{\max}(M). \qquad \square$$

4. Contraction hierarchies and filled graphs

In this section, we establish a link between contraction hierarchies and the more well-studied graph elimination game, which was introduced by Parter [22]. Let $G = (V, E)$ be an undirected graph and let α be an ordering of its vertices. We consider the so-called *elimination game* played on G. Beginning at $G^1 = G$, one removes in each step $i = 1, \ldots, n$ the vertex $v_i = \alpha^{-1}(i)$ and its incident edges from G^i. Afterwards, the graph G^{i+1} is obtained from G^i by inserting *fill edges*, such that the neighbors of v_i in G_i form a clique in G^{i+1}. Denote by E^i the set of edges inserted in step i, and let $E_f = \bigcup_{i=1}^n E^i$. The filled graph $F = F(G, \alpha)$ is now commonly defined to be the undirected graph with edge set $E \cup E_f$. For our purposes it is more convenient to define the filled graph as the according directed graph with all arcs pointing upwards with respect to α. That is, the *filled graph* $F = F(G, \alpha) = (V, A^f)$ is defined by $A^f = \{uv \mid \{u, v\} \in E \cup E_f \text{ and } \alpha(u) < \alpha(v)\}$. An example of an undirected graph G and its associated filled graph $F(G, \alpha)$ can be seen in Fig. 2. Note that the only difference from

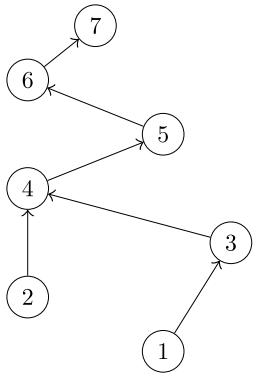


Fig. 3. The elimination tree $T = T(G, \alpha)$ associated with the filled graph $F(G, \alpha)$ shown in Fig. 2. It is rooted in the vertex 7 and has height $\mathrm{ht}(T) = 5$.

the construction of $M = M(G, \alpha)$ is that M is constructed from a digraph, whereas the elimination game is played on an undirected graph. We define the filled graph $F = F(G, \alpha)$ of a directed graph simply to be the filled graph of the undirected graph underlying G. The following theorem immediately follows from the construction of M and F.

Theorem 2. *Let G be a directed graph and α an ordering of its vertices. Further let $M = M(G, \alpha)$ be the corresponding maximal contraction hierarchy and let $\overleftarrow{M}^{\vee}$ denote $M^{\vee}$ with reversed arcs. Then both $M^{\wedge}$ and $\overleftarrow{M}^{\vee}$ are subgraphs of the filled graph $F = F(G, \alpha)$. Moreover, if G contains for each arc uv also the opposite arc vu, then $M^{\wedge} = \overleftarrow{M}^{\vee} = F$.*

Theorem 2 has many far-reaching consequences, and we will only explore a few of them in this paper. It turns out that the definition of M is nothing essentially new, and has indeed already been defined and studied by Rose and Tarjan [24]. However, much of the work on filled graphs is primarily concerned with the problem of minimizing the number of arcs in $F(G, \alpha)$; see the survey by Heggernes [19]. Minimizing the fill-in corresponds to minimizing the number of shortcuts in a contraction hierarchy, and hence its space requirements. We rather focus on the implications of Theorem 2 regarding search spaces and their size.

Corollary 1. *Let $G = (V, A)$ be a weighted digraph with vertex ordering α. Let $M = M(G, \alpha)$ be the associated maximal contraction hierarchy and $F = F(G, \alpha)$ the associated filled graph. Then $S(u, M^{\wedge}), R(u, M^{\vee}) \subseteq S(u, F)$ for all $u \in V$. In particular $S_{\max}(M) \leq S_{\max}(F)$.*

An analogous statement holds for the average search space size S_{avg}. Our next goal is an alternative description of $S_{\max}(F)$ known as the height of the elimination tree of F. For this purpose consider again an undirected graph G with filled graph F. Associated with F is the so-called *elimination tree* $T = T(G, \alpha)$ of G. The elimination tree T has vertex set V but contains for each $u \in V$ only the arc uv of F with minimal $\alpha(v)$. An example of an elimination tree can be seen in Fig. 3. Again the usual definition of T is undirected, but it is natural to choose $\alpha^{-1}(n)$ as the root. With this choice the usual definition coincides with our definition. The height of T with respect to this root is the *elimination tree height*, denoted by $\mathrm{ht}(T)$ or $\mathrm{ht}(G, \alpha)$. The following lemmas relate the search space size in F with $\mathrm{ht}(T)$.

Lemma 6. *Let G be a connected graph, α an order of V and $T = T(G, \alpha)$ the associated elimination tree. Then $S_{\max}(T) = 1 + \mathrm{ht}(T)$.*

Proof. Let r denote the root of T, and denote by $p(u)$ the vertices lying on the path from u to r. Then $\mathrm{ht}(T) = \max_{u \in V} |p(u)| - 1$ and it therefore suffices to show $S(u, T) = p(u)$ for all $u \in V$. This is trivially satisfied, since due to the connectivity of G, each vertex $u \in V$ distinct from r is the source of precisely one arc uv. □

The crucial property of the elimination tree T is that if u and v are two vertices connected by a path p in the corresponding filled graph, then there is a path p' from u to v in T. It obviously suffices to prove this statement when p is an arc, as the general case then follows by induction.

Lemma 7. *If uv is an arc of the filled graph $F = F(G, \alpha)$, then there exists a path with source u and target v in the elimination tree $T = T(G, \alpha)$.*

Proof. Denote by $p(u)$ the unique path in T from u to the root r. We show by descending induction on $\alpha(u)$ that we have $p(v) \subseteq p(u)$ for all arcs uv of the filled graph F. 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.

If $\alpha(u) = n-1$, then $\alpha(v) = n$, and hence $v = r$. The claim holds trivially.

If $\alpha(u) < n-1$, let uw be the unique arc of T with source u. By the definition of T, we have $\alpha(w) \leq \alpha(v)$, and $p(u) = uw \cdot p(w)$. If $v = w$, we are done. Otherwise, F contains the arc vw as both v and w are neighbors of u at the time of its removal during the elimination game. The induction hypothesis therefore implies $p(v) \subseteq p(w)$, and hence $p(v) \subseteq p(u)$. □

This allows us to finally relate search spaces in the filled graph F and in the elimination tree T.

Corollary 2. *Let $G = (V, E)$ be a graph, α an order on V, $F = F(G, \alpha)$ the corresponding filled graph and $T = T(G, \alpha)$ the corresponding elimination tree. Then $S(u, T) = S(u, F)$ for all $u \in V$.*

Proof. The fact that $T \subseteq F$ immediately implies $S(u, T) \subseteq S(u, F)$. Conversely, if $v \in S(u, F)$, applying Lemma 7 to all arcs of a path from u to v in F, yields a path from u to v in T, implying $v \in S(u, T)$. □

Corollaries 1 and 2 immediately imply the following.

Corollary 3. *For any connected weighted digraph G with vertex ordering α, corresponding maximal contraction hierarchy $M = M(G, \alpha)$ and corresponding elimination tree $T = T(G, \alpha)$ we have*

$$S_{\max}(M) \leq \mathrm{ht}(T) + 1\,.$$

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 $\mathrm{ht}(T)$ into upper bounds on $S_{\max}(M)$. Upper bounds from the literature are not seldomly accompanied by algorithms to determine orders α so that $\mathrm{ht}(T)$ is at most the upper bound at hand. Without any further modifications these algorithms may be used to compute contraction orders α with good upper bounds on $S_{\max}(M)$.

As a first application of the above result let us consider contraction hierarchies of undirected trees T. For this purpose we consider the undirected tree T as a directed graph that contains each edge in both directions. In this case, each simple path is the unique shortest path between its endpoints. Hence $C(T, \alpha)$ contains each possible shortcut in the sense that whenever it contains uv and vw with $\alpha(v) < \min\{\alpha(u), \alpha(w)\}$, then it contains the shortcut uw. Hence, $C(T, \alpha)$, the maximal weak contraction hierarchy $M(T, \alpha)$ and the filled graph $F(T, \alpha)$ coincide. Moreover, Schäffer [25] has given a linear-time algorithm to compute optimal elimination orders for trees, and we may thus conclude that the problem of minimizing $S_{\max}(C(T, \alpha))$ is solvable in linear time. The techniques of the next section additionally give a 2-approximation for $S_{\mathrm{avg}}(C(T, \alpha))$; see Section 6.

5. Contraction orders from nested dissection

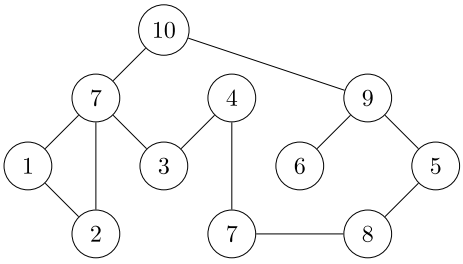
Next we show how to compute orders that yield small search spaces for several classes of graphs. The main idea is to exploit Corollary 3, which relates search space sizes and elimination tree heights. One way to construct orderings that give guarantees on the elimination tree height is the use of *nested dissection*, which goes back to George [17].

Let $0 < b < 1$ and let $f(n)$ be a monotonically increasing function. A (b, f)-*balanced separator decomposition* of an undirected n-vertex graph $G = (V, E)$ is a rooted tree $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ whose nodes $X \in \mathcal{X}$ are disjoint subsets of V and that is recursively defined as follows. If $n \leq n_0$ for some fixed constant n_0, then $\mathcal{T}$ 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 the roots of (b, f)-balanced separator decompositions of $G_1, \ldots, G_d$. For clarity, we will always refer to the vertices of $\mathcal{T}$ as *nodes*. We use $\mathcal{T}_X$ to denote the subtree of $\mathcal{T}$ rooted at a node X, and by G_X the connected subgraph of G induced by the vertices contained in $\mathcal{T}_X$. For a vertex $u \in V$, we denote the unique node X of $\mathcal{T}$ with $u \in X$ by X_u. A node X of $\mathcal{T}$ has *level* $\mathrm{level}(X) = i$ if the unique simple path from X to the root of $\mathcal{T}$ has length i. Fig. 4 contains an example of such a (b, f)-balanced separator decomposition that may also serve as an illustration of the following remark.

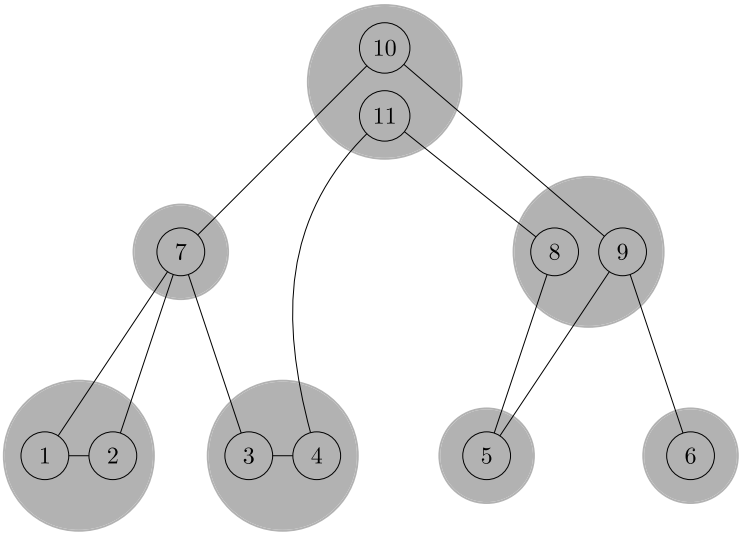
Remark 1. Let G be an undirected graph and let $\mathcal{T}$ be a (b, f)-balanced separator decomposition of G. If $\{u, v\}$ is an edge of G with $\mathrm{level}(X_u) \geq \mathrm{level}(X_v)$, then X_v is an ancestor of X_u.

Proof. Let R denote the root of $\mathcal{T}$. Consider the lowest common ancestor X of X_u and X_v in $\mathcal{T}$. If $X \neq X_v$, then X_u and X_v lie in distinct subtrees of $\mathcal{T}_X$. However, by construction of $\mathcal{T}$, this means that X separates X_u from X_v, contradicting the existence of the edge $\{u, v\}$. □

Given a (b, f)-balanced separator decomposition of G, we determine an associated (b, f)-*balanced nested dissection order* α on the vertices of G by performing a post-order traversal of $\mathcal{T}$, where the vertices of each node are visited in an arbitrary order. We denote such an order by $\alpha(\mathcal{T})$. It follows immediately from Remark 1 and the construction of $\alpha = \alpha(\mathcal{T})$



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



(b) A $(1/2, \sqrt{n})$-balanced separator decomposition $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ of the graph shown in Figure 4a. The nodes $X \in \mathcal{X}$ are drawn as grey disks containing precisely those $u \in V$ with $u \in X$. The root of $\mathcal{T}$ is drawn at the top.

Fig. 4. A graph $G = (V, E)$ and a $(1/2, \sqrt{n})$-balanced separator decomposition of G.

that for any edge $\{u, v\}$ of G with $\alpha(u) < \alpha(v)$ the node X_v is an ancestor of X_u. This property remains valid also for the corresponding filled graph $F(G, \alpha)$ as can also be seen in Fig. 5.

Lemma 8. *Let $G = (V, E)$ be an undirected graph and $\alpha = \alpha(\mathcal{T})$ a 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 $F(G, \alpha)$ with $\alpha(u) < \alpha(v)$.*

Proof. It suffices to show that X_v is an ancestor of X_u for each edge $\{u, v\}$ with $\alpha(u) < \alpha(v)$ of G^i, where G^i is the graph before the ith step in the elimination game. Observe that the above discussion establishes exactly this property for $G = G^1$. We show that G^{i+1} satisfies the property if G^i does.

Consider the vertex $v_i = \alpha^{-1}(i)$ that is removed in the ith step of the elimination game. Let $\{u, v\}$ be a fill-edge with $\alpha(u) < \alpha(v)$ that is inserted in this step. Then G^i contains edges $\{v_i, u\}$ and $\{v_i, v\}$. By the induction hypothesis, we have that both X_u and X_v are ancestors of X_{v_i} in $\mathcal{T}$ since $\alpha(v_i)$ is minimal among the vertices in G^i. Note that this implies that either X_u is an ancestor of X_v or vice versa. Our assumption $\alpha(u) < \alpha(v)$ and the construction of α imply that X_u is an ancestor of X_v. This finishes the proof. □

To simplify notation, we denote by $\mathcal{T}(u)$ the union of all nodes that lie on the unique simple path from X_u to the root of $\mathcal{T}$. The above lemma has immediate implications in terms of search spaces and elimination tree height. An easy induction over the length of a path in $F(G, \alpha)$ yields the following.

Corollary 4. *Let $G = (V, E)$ be an undirected graph, $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ an (b, f)-balanced separator decomposition of G, $\alpha = \alpha(\mathcal{T})$ an associated nested dissection order and $F = F(G, \alpha)$ the corresponding filled graph with elimination tree $T = T(G, \alpha)$. Then (i) $S(u, F) \subseteq \mathcal{T}(u)$, and (ii) $\mathrm{ht}(T) \leq |\mathcal{T}(u)|$.*

In particular, Corollaries 1 and 4 together imply the following theorem.

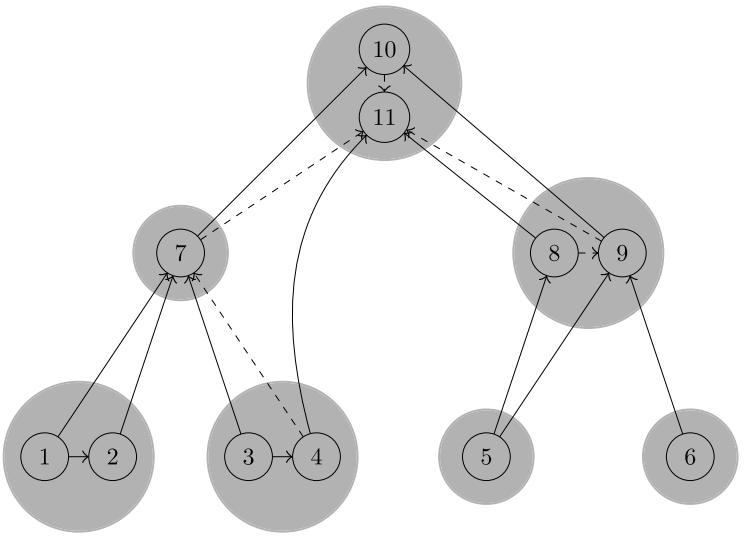


Fig. 5. The filled graph $F(G,\alpha)$ of the graph G from Fig. 4a with dashed fill arcs for a $(1/2,\sqrt{n})$-balanced nested dissection order $\alpha=\alpha(\mathcal{T})$ determined by the decomposition shown in Fig. 4b. Vertices u of T are labeled with $\alpha(u)$.

Theorem 3. *Let $G=(V,E)$ be a weighted digraph and $\alpha=\alpha(\mathcal{T})$ a nested dissection order of a (b,f)-balanced separator decomposition of the undirected graph underlying G and $M=M(G,\alpha)$ the maximal contraction hierarchy w.r.t. α. Then we have $|S(u,M^{\wedge})|, |R(u,M^{\vee})| \le |\mathcal{T}(u)|$.*

In order to find upper bounds on $S_{\max}(M)$, it remains to bound $|\mathcal{T}(u)|$. This issue cannot be handled simultaneously for all families of graphs, but needs special treatment depending on the properties of G. We will first study a rather general setting, and afterwards specialize to graphs that exclude a fixed minor.

It is not hard to see that $|\mathcal{T}(u)| \le n_0 + h\cdot f(n)$, where h is the height of $\mathcal{T}$. Further, it follows from the balance of the decomposition that $h \le \log_{1/b} n$. In particular, for graphs with treewidth at most k, the $(1/2,1)$-balanced separator decomposition for trees facilitates a $(1/2,k+1)$-balanced separator decomposition with $n_0=k+1$. We have the following theorem.

Theorem 4. *Let G be a weighted digraph of treewidth at most k. There exists an order α with corresponding maximal contraction hierarchy $M=M(G,\alpha)$, such that $S_{\max}(M) \le (k+1)(1+\log n)$ and $|M| \le 2n(k+1)(1+\log n)$.*

If the separator size is not fixed, but depends on the graph size, better bounds can be achieved. For minor-closed graph classes that admit $(b,a\sqrt{n})$-balanced separators, it is known that the elimination tree height $\mathrm{ht}(G,\alpha)$ for a nested dissection order α is at most $n_0+\sum_{i=0}^{\infty} a\sqrt{b^i n} = n_0 + a/(1-\sqrt{b})\sqrt{n} = O(\sqrt{n})$. According to Lemma 5, this yields a contraction hierarchy of size $O(n^{3/2})$. However, by results of [20], any minor-closed class of graphs that admit $(b,O(\sqrt{n}))$-balanced separators is a family of sparse graphs, and a more sophisticated analysis due to Gilbert and Tarjan [18] then proves that the number of fill arcs is $O(n\log n)$. The next theorem summarizes this discussion.

Theorem 5. *Let $\mathcal{C}$ be a minor-closed graph class with $(b,O(\sqrt{n}))$-balanced separators for some fixed $0<b<1$. Any $G\in\mathcal{C}$ admits an order α with corresponding maximal contraction hierarchy $M=M(G,\alpha)$ such that $S_{\max}(M)=O(\sqrt{n})$ and $|M|=O(n\log n)$.*

6. 2-Approximation of S_{avg} for trees

In this section we consider contraction hierarchies of undirected trees. It turns out that the particularly simple nested dissection orders of undirected trees yield contraction hierarchies C whose average search space size $S_{\mathrm{avg}}(C) = 1/n\cdot\sum_{v\in V}|S(v,C^{\wedge})| + |R(v,C^{\vee})|$ is approximately optimal. Apart from the fact that undirected trees can always be separated by just a single vertex, the key property of contraction hierarchies of trees that enters the proof is the following lemma.

Lemma 9. *Let $T=(V,E)$ be an undirected tree and let $T'\subseteq T$ be a subtree with vertices U. Further let α be an order on V and let α' be its restriction to U. The contraction hierarchy $C(T',\alpha')$ of T' is the subgraph of $C(T,\alpha)$ induced by U.*

Proof. Let $u, v \in U$. Suppose that uv is an arc of $C = C(T, \alpha)$, i.e., that all inner vertices w on the unique path between u and v in T satisfy $\alpha(w) < \min\{\alpha(u), \alpha(v)\}$. This unique path between u and v in T is also the only path between u and v in T' and hence uv is an arc of $C' = C(T', \alpha')$, too.

Conversely, if uv is an arc of C' but not of C, there is (a) a path between u and v in $T' \subseteq T$ all whose inner vertices w satisfy $\alpha(w) < \min\{\alpha(u), \alpha(v)\}$ and (b) a path between u and v in T with at least one inner vertex w, such that $\alpha(w) > \min\{\alpha(u), \alpha(v)\}$. Thus, there is a cycle in T—a contradiction. □

Given this lemma, it is now straightforward to relate the average search space size in a contraction hierarchy $C = C(T, \alpha)$ of a tree T to the average search space sizes in the contraction hierarchies of the connected components of $T - v$ for (a) the α-maximal vertex v, and (b) a 1/2-separator v of T. These relations are the content of the following two technical lemmas.

Lemma 10. *Let $T = (V, E)$ be an undirected tree, α an order on V with associated contraction hierarchy $C = C(T, \alpha)$ and $v \in V$ the α-maximal vertex. Let further d be the number of connected components T_i of $T - v$ and let $C_i = C(T_i, \alpha_i)$ be the contraction hierarchy of $T_i = (V_i, E_i)$ with respect to the restriction α_i of α to V_i. Then*

$$S_{\text{avg}}(C) = 2 + \frac{1}{n} \cdot \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(C_i)\,,$$

where $n_i = |V_i|$.

Proof. By Lemma 9, the search space $S(s, C^\wedge)$ of any vertex $s \in V_i$ decomposes as the disjoint union of $\{v\}$ and $S(s, C_i^\wedge)$. Moreover, T is assumed to be undirected and hence $S(s, C^\wedge) = R(s, C^\vee)$. Therefore,

$$\begin{aligned} n \cdot S_{\text{avg}}(C) &= \sum_{s \in V} |S(s, C^\wedge)| + |R(s, C^\vee)| = \sum_{s \in V} 2 \cdot |S(s, C^\wedge)| \\ &= 2 + \sum_{i=1}^{d} \sum_{s \in V_i} 2 \cdot |S(s, C^\wedge)| = 2 + \sum_{i=1}^{d} \sum_{s \in V_i} (2 \cdot |S(s, C_i^\wedge)| + 2) \\ &= 2 + 2(n-1) + \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(C_i) = 2n + \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(C_i), \end{aligned}$$

which gives the desired equality after division by n. □

Lemma 11. *Let $T = (V, E)$ be an undirected tree, β an arbitrary order on V and $v \in V$ a 1/2-separator, such that v is not β-maximal. Denote the components of $T - v$ by $T_i = (V_i, E_i)$ and their number of vertices by n_i. Further let $C = C(T, \beta)$ and $C_i = C(T_i, \beta_i)$, where β_i denotes the restriction of β to V_i. Then*

$$S_{\text{avg}}(C) \geq 1 + \frac{1}{n} \cdot \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(C_i).$$

Proof. Assume without loss of generality that the β-maximal vertex u is contained in V_1. Then

$$\begin{aligned} n \cdot S_{\text{avg}}(C) &= \sum_{s \in V} |S(s, C^\wedge)| + |R(s, C^\vee)| = \sum_{s \in V} 2 \cdot |S(s, C^\wedge)| \\ &= 2 \cdot |S(v, C^\wedge)| + \sum_{s \in V_1} 2 \cdot |S(s, C^\wedge)| + \sum_{i=2}^{d} \sum_{s \in V_i} 2 \cdot |S(s, C^\wedge)|. \end{aligned}$$

We may estimate (i) $|S(v, C^\wedge)| \geq 1$, (ii) $|S(s, C^\wedge)| \geq |S(s, C_1^\wedge)|$ for all $s \in V_1$ and (iii) $|S(s, C^\wedge)| \geq 1 + |S(s, C_i^\wedge)|$ for all $s \in V_i$ with $i \neq 1$. Hence, we get

$$\begin{aligned} n \cdot S_{\text{avg}}(C) &\geq 2 + \sum_{s \in V_1} 2 \cdot |S(s, C_1^\wedge)| + \sum_{i=2}^{d} \sum_{s \in V_i} 2 \cdot |S(s, C_i^\wedge)| + 2 \\ &= 2 + 2(n - n_1 - 1) + \sum_{i=1}^{d} n_i \cdot |S_{\text{avg}}(C_i)| = 2(n - n_1) + \sum_{i=1}^{d} n_i \cdot |S_{\text{avg}}(C_i)|. \end{aligned}$$

Now $n - n_1 \geq n/2$ since v is a 1/2-separator and the claim follows by division through n. □

Theorem 6. *Let $T = (V, E)$ be an undirected tree, let α be a nested dissection order associated with a $(\frac{1}{2}, 1)$-balanced separator decomposition, and let $C = C(T, \alpha)$ be the corresponding contraction hierarchy. Then $S_{\text{avg}}(C) \leq 2 \cdot S_{\text{avg}}(D)$ for all contraction hierarchies $D = C(T, \beta)$ of T.*

Proof. The proof is by induction on $n = |V|$. If $n = 1$, there is nothing to show. If $n > 1$, Lemma 10, the induction hypothesis and Lemma 11 imply

$$\begin{aligned} S_{\text{avg}}(C) &= 2 + \frac{1}{n} \cdot \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(C_i) \leq 2 + \frac{1}{n} \cdot \sum_{i=1}^{d} n_i \cdot 2 \cdot S_{\text{avg}}(D_i) \\ &= 2 \cdot \left(1 + \frac{1}{n} \cdot \sum_{i=1}^{d} n_i \cdot S_{\text{avg}}(D_i)\right) \leq 2 \cdot S_{\text{avg}}(D), \end{aligned}$$

where $C_i = C(T_i, \alpha_i)$ and $D_i = C(T_i, \beta_i)$ are the contraction hierarchies of the connected components of $T - \alpha^{-1}(n)$ with respect to the restrictions α_i and β_i of α and β to V_i. □

7. Comparison with highway dimension

In this section, we compare our bounds with the ones obtained by Abraham et al. [1,2]. Their results employ the highway dimension, a notion that, unlike the graph parameters we use, also depends on the edge lengths. We show that their bounds are comparable to ours if the edge lengths are sufficiently ill-behaved.

First, we briefly recall the definition of highway dimension. Let $G = (V, E)$ be a weighted undirected graph. 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, and is denoted by $B_\varepsilon(u)$. We say that a vertex u *covers* a shortest path p if p contains u, and that p *intersects* a ball $B_\varepsilon(u)$ if p contains a vertex of $B_\varepsilon(u)$.

The *highway dimension* $\text{hd}(G)$ of G is the smallest integer d, such that for all $\varepsilon > 0$ and for each $u \in V$, all the shortest paths p in G of length $\varepsilon < \text{len}(p) \leq 2\varepsilon$ that intersect $B_{2\varepsilon}(u)$ can be covered by at most d vertices. Abraham et al. [1] prove that for a n-vertex weighted graph G with highway dimension d, diameter D and maximum degree Δ, there exists an ordering α, such that the size of $C(G, \alpha)$ is $O(nd \log D)$, and such that distance queries can be answered in time $O((\Delta + d \log D) \cdot d \log D)$.

In the remainder of this section, we proceed as follows. We consider the maximum k of $\text{hd}(G)$ over all possible choice of length functions on G and construct a $(2/3, k)$-balanced separator decomposition of G, which then provides the link to nested dissection orders. We note that the same proof can be adapted to the slightly different definition of highway dimension in [2].

Lemma 12. *Let $G = (V, E)$ be a connected graph, let k be the maximum highway dimension over all possible edge lengths on G, and let $H \subseteq G$ be a connected subgraph with $|V(H)| \geq 2k + 2$. Then H can be separated into at least two connected components of size at most $\lceil |V(H)|/2 \rceil$ by removing at most k vertices.*

Proof. Let H be a connected subgraph of G with h vertices and let $H_1 \subseteq H$ be a connected subgraph with $\lfloor h/2 \rfloor$ vertices. Denote the vertex sets of H and H_1 by V_H and V_1, respectively. Define lengths $\text{len}\colon E \to \mathbb{R}^+$ by setting the length of an edge to 1 if it has exactly one endpoint in V_1, and to $\varepsilon = 3/h$, otherwise. Observe that the length of a simple path in H_1 is at most $h\varepsilon/2 \leq 3/2$.

Consider a ball B with radius $3/2$ around any vertex u of H_1. Then $V_1 \subseteq B$. By our choice of k, and the definition of highway dimension, there exists 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 the removal of $S \cap V_H$ separates H into at least two connected components with at most $\lceil h/2 \rceil$ vertices, each.

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 $v \in V \setminus V_1$. By the choice of edge lengths, this edge is a shortest path of length 1, and thus one of its endpoints is in S. Hence S separates H_1 from $H \setminus V_1$. It remains to verify that the connected components of $H \setminus S$ contain at most $\lceil h/2 \rceil$ vertices, each, and that there are at least two such components. The former claim follows immediately from our choice of H_1. For the latter claim, note that $|V_1| \geq k + 1$ and $|V_H \setminus V_1| \geq k + 1$ imply $|V_1 \setminus S| \geq |V_1| - k \geq 1$ and $|V_H \setminus (V_1 \cup S)| \geq |V_H \setminus V_1| - k \geq 1$. □

This lemma allows us to separate each subgraph of $n' \geq 2k + 2$ vertices into at least two connected components with at most $\lceil n'/2 \rceil \leq \frac{2}{3}n'$ vertices by removing at most k vertices. We have the following corollary.

Corollary 5. *Let $G = (V, E)$ be a connected undirected graph with maximum highway dimension k. Then G admits a $(2/3, k)$-balanced separator decomposition, whose leaves have size at most $2k + 1$.*

A simple calculation shows that the nested dissection order resulting from Corollary 5 yields an elimination tree T of height $\mathrm{ht}(T) \leq 2k + 1 + k \cdot \frac{\log n}{\log(3)-1}$. It is known for the pathwidth $\mathrm{pw}(G)$ that $\mathrm{pw}(G) \leq \mathrm{ht}(T)$ [8].

Theorem 7. *Let G be an undirected graph. There exist edge lengths on G, such that $\mathrm{hd}(G) > \frac{\mathrm{pw}(G)-1}{2\log n+1}$.*

To our knowledge, this is a novel and unanticipated relation between highway dimension and more commonly used graph parameters. Moreover, Corollary 5 allows a comparison of our results with those of Abraham et al. [1].

Theorem 8. *Let G be an undirected graph with diameter D and maximum degree Δ. Let β denote the order constructed by Abraham et al. [1] and let $C = C(G, \beta)$ be the associated contraction hierarchy. There exist edge lengths on G and a nested dissection order α with associated maximal contraction hierarchy $M = M(G, \alpha)$, such that*

(a) $|M| \leq O(\log n / \log D)|C|$, *and*
(b) *the worst-case running time of distance queries in M is at most a factor of $O(\log^2(n)/\log^2(D))$ greater than that in C.*

Proof. Choose the edge lengths such that G attains its maximum highway dimension k. Recall from [1], that their optimal order β results in a contraction hierarchy $C(G, \beta)$ that has $m_\beta = O(nk\log(D))$ arcs and on which a distance query has worst-case running time $T^{\beta}_{\text{query}} = O((\Delta + k\log D) \cdot k \log D)$.

By virtue of Theorem 4 and Corollary 5, there exists a nested-dissection ordering α, such that the maximum search space size $S_{\max}(M)$ in $M = M(G, \alpha)$ is of the order of $O(2k + 1 + k\log n) = O(k \log n)$. Using Lemma 5, we have $|M| = O(nk \log n)$, which immediately implies (a). For (b), we use that Dijkstra's algorithm relaxes at most $S_{\max}(M)^2$ edges, and we thus have $T^{\alpha}_{\text{query}} = O(k^2 \log^2 n)$. □

We note that, for graphs that bear some resemblance to road networks, it seems quite likely that $\log n / \log D = \Theta(1)$, e.g., if the shortest-path diameter is roughly n^k for some $0 < k < 1$. It is remarkable that the results are so close, given that Abraham et al. bound both the vertices and arcs in the search space, while our crude bound on the number of arcs is simply the square of the number of vertices in the search space. Any improvement on this bound would immediately imply faster query times. It is moreover worth noting that our machinery neither requires small maximum degree nor small diameter.

8. Conclusion

We have developed a theoretical framework for studying search-space sizes in contraction hierarchies. Our main contributions are a global description of contraction hierarchies and the connection to elimination games. Using nested dissection, we are able to compute contraction orders with sublinear search spaces for large classes of graphs. Under a worst-case assumption on the highway dimension, our results, even though our constructions ignore edge lengths, are comparable to those of Abraham et al. [1]. As a side result, we found the first connection of highway dimension to a more widely used graph parameter. Our main open questions are whether stronger bounds on the number of arcs that are spanned by search spaces can be proven, and whether there exist efficient algorithms to compute node orders that produce approximatively optimal search spaces in terms of the maximum or average search space size.

References

[1] I. Abraham, D. Delling, A. Fiat, A.V. Goldberg, R.F. Werneck, VC-dimension and shortest path algorithms, in: Proc. 38th International Colloquium on Automata, Languages, and Programming, ICALP'11, in: LNCS, vol. 6755, Springer, 2011, pp. 690–699.
[2] I. Abraham, A. Fiat, A.V. Goldberg, R.F. Werneck, Highway dimension, shortest paths, and provably efficient algorithms, in: Proc. 21st Annual ACM–SIAM Symp. Disc. Alg., SODA'10, SIAM, 2010, pp. 782–793.
[3] H. Bast, D. Delling, A.V. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, R.F. Werneck, Route planning in transportation networks, Technical Report abs/1504.05140, ArXiv e-prints, 2016.
[4] G.V. Batz, R. Geisberger, P. Sanders, C. Vetter, Minimum time-dependent travel times with contraction hierarchies, ACM J. Exp. Algorithmics 18 (1.4) (April 2013) 1–43.
[5] R. Bauer, M. Baum, I. Rutter, D. Wagner, On the complexity of partitioning graphs for arc-flags, in: Proc. 12th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, ATMOS'12, in: OpenAccess Series in Informatics (OASIcs), 2012, pp. 71–82.
[6] R. Bauer, T. Columbus, B. Katz, M. Krug, D. Wagner, Preprocessing speed-up techniques is hard, in: Proc. 7th Conference on Algorithms and Complexity, CIAC'10, in: LNCS, vol. 6078, Springer, 2010, pp. 359–370.
[7] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, T. Zündorf, Shortest feasible paths with charging stops for battery electric vehicles, in: Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM Press, 2015, pp. 44:1–44:10.
[8] H.L. Bodlaender, J.R. Gilbert, T. Kloks, H. Hafsteinsson, Approximating treewidth, pathwidth, and minimum elimination tree height, in: Proc. 17th Internat. Workshop Graph-Theoretic Concepts in Computer Science, WG'91, in: LNCS, vol. 570, Springer, 1991, pp. 1–12.
[9] S. Chaudhuri, C.D. Zaroliagis, Shortest paths in digraphs of small treewidth. Part I: sequential algorithms, Algorithmica 27 (3) (2000) 212–226.
[10] D. Delling, A.V. Goldberg, A. Nowatzyk, R.F. Werneck, PHAST: hardware-accelerated shortest path trees, J. Parallel Distrib. Comput. 73 (7) (2013) 940–952.
[11] J. Dibbelt, T. Pajor, D. Wagner, User-constrained multi-modal route planning, ACM J. Exp. Algorithmics 19 (April 2015) 3.2.
[12] J. Dibbelt, B. Strasser, D. Wagner, Customizable contraction hierarchies, in: J. Gudmundsson, J. Katajainen (Eds.), Proceedings of the 13th International Symposium on Experimental Algorithms, SEA'14, in: LNCS, vol. 8504, Springer, 2014, pp. 271–282.

[13] J. Fakcharoenphol, S. Rao, Negative weight edges, shortest paths, near linear time, in: Proceedings of the 42nd Symposium on Foundations of Computer Science, FOCS'01, 2001, pp. 232–241.
[14] S. Funke, S. Storandt, Polynomial-time construction of contraction hierarchies for multi-criteria objectives, in: Proceedings of the 15th Meeting on Algorithm Engineering and Experiments, ALENEX'13, SIAM, 2013, pp. 31–54.
[15] S. Funke, S. Storandt, Provable efficiency of contraction hierarchies with randomized preprocessing, in: K. Elbassioni, K. Makino (Eds.), Proceedings of the 26th International Symposium on Algorithms and Computation, ISAAC'15, in: LNCS, vol. 9472, Springer, 2015, pp. 479–490.
[16] R. Geisberger, P. Sanders, D. Schultes, C. Vetter, Exact routing in large road networks using contraction hierarchies, Transp. Sci. 46 (3) (2012) 388–404.
[17] A. George, Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. 10 (2) (1973) 345–363.
[18] J.R. Gilbert, R.E. Tarjan, The analysis of a nested dissection algorithm, Numer. Math. 50 (4) (1986) 377–404.
[19] P. Heggernes, Minimal triangulations of graphs: a survey, Discrete Math. 306 (3) (2006) 297–317.
[20] R.J. Lipton, D.J. Rose, R.E. Tarjan, Generalized nested dissection, SIAM J. Numer. Anal. 16 (2) (1979) 346–358.
[21] N. Milosavljević, On optimal preprocessing for contraction hierarchies, in: Proc. 5th Internat. Workshop Comput. Transport. Sci., IWCTS'12, 2012.
[22] S.V. Parter, The use of linear graphs in Gaussian elimination, SIAM Rev. 3 (2) (1961) 119–130.
[23] L.R. Planken, M.M. de Weerdt, R.P. van der Krogt, Computing all-pairs shortest paths by leveraging low treewidth, J. Artificial Intelligence Res. 43 (2012) 353–388.
[24] D.J. Rose, R.E. Tarjan, Algorithmic aspects of vertex elimination on directed graphs, SIAM J. Appl. Math. 34 (1) (1978) 176–197.
[25] A.A. Schäffer, Optimal node ranking of trees in linear time, Inform. Process. Lett. 33 (2) (1989) 91–96.
[26] R. Schreiber, A new implementation of sparse Gaussian elimination, ACM Trans. Math. Software (TOMS) 8 (3) (1982) 256–276.
[27] F. Schulz, D. Wagner, C.D. Zaroliagis, Using multi-level graphs for timetable information, in: Proceedings of the 4th Workshop on Algorithm Engineering and Experiments, ALENEX'02, in: LNCS, vol. 2409, Springer, 2002, pp. 43–59.
[28] M. Thorup, Compact oracles for reachability and approximate distances in planar digraphs, J. ACM 51 (6) (2004) 993–1024.