Karlsruhe Reports in Informatics 2010,4
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

Preprocessing Speed-Up Techniques is Hard

Reinhard Bauer, Tobias Columbus, Bastian Katz,
Marcus Krug, Dorothea Wagner

2010

KIT – University of the State of Baden-Wuerttemberg and National Research Center of the Helmholtz Association

Preprocessing Speed-Up Techniques is Hard *

Reinhard Bauer, Tobias Columbus, Bastian Katz, Marcus Krug, and Dorothea Wagner

Faculty of Informatics, Karlsruhe Institute of Technology (KIT)
{Reinhard.Bauer,Bastian.Katz,Marcus.Krug, Dorothea.Wagner}@kit.edu,
columbus@informatik.uni-karlsruhe.de

Abstract. During the last years, preprocessing-based techniques have been developed to compute shortest paths between two given points in a road network. These *speed-up techniques* make the computation a matter of microseconds even on huge networks. While there is a vast amount of experimental work in the field, there is still large demand on theoretical foundations. The preprocessing phases of most speed-up techniques leave open some degree of freedom which, in practice, is filled in a heuristical fashion. Thus, for a given speed-up technique, the problem arises of how to fill the according degree of freedom optimally. Until now, the complexity status of these problems was unknown. In this work, we answer this question by showing NP-hardness for the recent techniques. Part of this report has been published in [3]. However, this work includes all proofs omitted there.

1 Introduction

Computing shortest paths in graphs is used in many real-world applications like route-planning in road networks or for finding good connections in railway timetable information systems. In general, DIJKSTRA's algorithm computes a shortest path between a given source and a given target. Unfortunately, the algorithm is slow on huge datasets. Therefore, it cannot be directly used for applications like car navigation systems or online working route-planners that require an instant answer of a source-target query.

Often, this problem is coped with by dividing the computation of the shortest paths into two stages. In the *offline stage*, some data is precomputed that is used in the *online stage* to answer a source-target query heuristically faster than DIJKSTRA's algorithm. Such an approach is called a *speed-up technique* (for Dijkstra's algorithm). During the last years, speed-up techniques have been developed for road networks (see [6] for an overview), that make the shortest path computation a matter of microseconds even on huge road networks consisting of millions of nodes and edges.

Usually, the offline stage leaves open some degree of freedom, like the choice of how to partition a graph or of how to order a set of nodes. The decision taken to fill the respective degree of freedom has direct impact on the search space of the online stage and therefore on the runtime of a query. Currently, these decisions are made in a purely heuristical fashion. A common trade-off is between preprocessing time/space and query time/search space. In this paper we show the NP-hardness of preprocessing the offline stage such that the average search space of the query becomes optimal. For each technique, we demand that the size of the preprocessed data should be bounded by a given parameter. This model is used because practitioners in the field usually compare their results by absolute query times, size of the search space, size of preprocessing and time needed for the preprocessing. In practice, the basic technique can be enriched by various heuristic improvements. We will not consider such improvements and stick to the basic core of each technique. This implies that, for the sake of simplicity, some techniques are slightly altered.

The techniques considered are ALT [10], Arc-Flags [17, 15], SHARC [5], Highway Node Routing / Multilevel-Overlay Graph [23, 22, 16] and Contraction Hierarchies [9]. We left out Geometric Containers [25, 26], Highway Hierarchies [19, 20] and Reach-Based Pruning [13, 11, 12] as their offline stage only contains tuning parameters but no real degree of freedom. However, two interesting aspects of Reach-Based Pruning are included. We further did not work on Transit Node Routing [21, 2] as this is a framework for which also parts of the query-algorithm are to be specified.

* Partially supported by the DFG (project WAG54/16-1).

Related Work. There is a huge amount of work on speed-up techniques. An overview on experimental work can be found in [6]. There is large demand on a theoretical foundation for the field and there exists only few theoretical work: In [4] results are given for a problem that is related to the technique of inserting *shortcuts* to the underlying graph. Recently, a graph-generator for road networks was given [1] and it is shown that graphs evolving from that generator possess a property called *low highway dimension*. For graphs with this property, a special preprocessing technique is proposed and runtime guarantees for Reach-Based Routing, Contraction Hierarchies, Transit Node Routing and Sharc using that preprocessing technique are given.

2 Preliminaries

Throughout the work $G = (V, E, len)$ denotes a directed weighted graph with n nodes, m edges and positive length function $len : E \to \mathbb{R}^+$. Given an edge (u, v) we call u the *source node* and v the *target node* of (u, v). Further, (u, v) is an *incoming edge* of v and an *outgoing edge* of u. With $\overline{G}$ we denote the reverse graph, i.e. the graph $(V, \{(v, u) \mid (u, v) \in E\})$. A path P from x_1 to x_k in G is a finite sequence $x_1, x_2, \ldots, x_k$ of nodes such that $(x_i, x_{i+1}) \in E$, $i = 1, \ldots, k-1$. The *length* of a path P in G is the sum of the length of all edges in P. A shortest path between nodes s and t is a path from s to t with minimum length. Given two nodes s and t, the distance $dist(s, t)$ from s to t is the length of a shortest path between s and t and infinity if no s-t-path exists.

Dijkstra's algorithm. Given a graph $G = (V, E)$ with length function $len : E \to \mathbb{R}^+$ and a root $s \in V$, Dijkstra's algorithm finds the distances from s to all nodes in the graph. For each node v in the graph, the algorithm maintains a status marker, indicating exactly one of the states *unvisited, visited* or *settled* and a distance label $d(v)$. A min-based priority queue Q is provided that contains all visited nodes, keyed by the distance label.

Each node v is initialized to be unvisited and $d(v)$ is set to be infinity. Then, $d(s)$ is set to be 0, s is set to be visited and inserted into Q. While there are visited nodes, the algorithm extracts one node v with minimal distance label from Q, marks v as finished and *relaxes* all of its outgoing edges (v, w). An edge (v, w) is relaxed as follows: If $d(w) \leq d(v) + len(v, w)$ nothing is to be done. Otherwise $d(w)$ is set to be $d(v) + len(v, w)$. If w is already visited, its priority in the queue is updated, otherwise it is marked as visited and inserted into the queue.

There are many possibilities to break ties when extracting nodes from the queue. Throughout this work, we additionally identify every node uniquely with an integer between 1 and $|V|$. Among all nodes with minimal distance in the queue, the smallest integer gets extracted first.

In this work we focus on s-t-queries, i.e. queries for which only a shortest s-t is of interest. Hence, the algorithm can break after the node t has been marked as settled. The *search-space* of the query is the set of nodes, settled up to that point.

Bidirectional Search. This approach starts a Dijkstra's search rooted at s on G (the *forward search*) and one rooted at t on $\overline{G}$ (the *backward search*). Whenever a node has been settled it is to be decided if the algorithm changes to the opposite search. A simple approach is to swap the direction every time a node is settled. The *distance balanced* bidirectional search changes to the other direction iff the minimal distance label of nodes in the queue is greater than the minimal distance label of nodes in the contrary queue. There are different possible stopping criteria for bidirectional search which get specified for the particular speed-up technique applied. During the run of the algorithm, the *tentative distance* is $\min\{\text{dist}(s, u) + \text{dist}(u, t) | \ u \text{ has been settled by both searches}\}$. Finally, $\text{dist}(s, t) = \min\{\text{dist}(s, v) + \text{dist}(v, t)\}$ over all nodes v, that get settled from both directions. The search-space of a bidirectional search is the union of the search-spaces of forward and backward search. We consider the search space to be a multi-set, i.e. when computing the size of the search space we count nodes that get settled in both directions twice.

Speed-up techniques. The query of each speed-up technique we consider is either a modified Dijkstra's algorithm or a modified bidirectional search. The output of an s-t-query is $\text{dist}(s,t)$. We do not consider extra techniques like path-unpacking (see [6] for a description). For a given technique, we write $V_{\mathcal{F}}(s,t)$ for the size of the search space of an s-t-query when choosing $\mathcal{F}$ to fill the particular degree of freedom.

A Repeating Pattern. We consider Dijkstra's search and assume that there is a set $T \in V$ such that $\text{dist}(s,t)$ is equal for each $t \in T$ and an arbitrary but fixed $s \in V$. Our aim is to compute the sum $\sum_{t \in T} V(s,t)$ of the search-spaces of all s-t-queries with $t \in T$. We remember that, when deciding which node to settle next, ties are broken according to some predefined order on the vertices. Hence we can decompose $\sum_{t \in T} V(s,t)$ as $|T| \cdot |\{v \in V | \, \text{dist}(s,v) < \text{dist}(s,t)\}| + |T|(|T|+1)/2$.

3 Reach-Based Pruning

Reach is a centrality measure indicating whether a node lies in the middle of a long shortest path. More formally, the reach $\mathcal{R}_P(v_i)$ of a node v_i with respect to a path $P = (v_1, \ldots, v_k)$ is $\min\{len(v_1, \ldots, v_i), len(v_i, \ldots, v_k)\}$. The reach $\mathcal{R}(v)$ of a node with respect to a graph G is $\max_{\{P \in \mathrm{SP} \mid v \in P\}} \mathcal{R}_P(v)$ where SP denotes the set of all shortest paths in G. For ease of notation, we consider a single vertex to be a path of length 0.

There exist different variants of how to use reach for pruning the search-space of a bidirectional Dijkstra's search, all of them sharing the same main idea. We use the *self-bounding* query explained later. In practice the approach is mixed with other ingredients like ALT, contraction and the computation of upper bounds for reach-values which we do not consider here. Further, inspired by Contraction Hierarchies we relax the stopping criterion. This has been shown to be reasonable by experimental tests.

The reach-query is bidirectional Dijkstra's algorithm with the following two modifications: First, no stopping criterion is used, hence all vertices reachable from the source get settled. Second, a node v is not settled if $\mathcal{R}(v)$ is smaller than its priority in the queue. Note that v can still get visited. We denote by $d^+(v)$ and $d^-(v)$ the length of the shortest path from s and to t respectively, found by visiting or settling node v. Finally, $\mathrm{dist}(s,t)$ is given by $\min(d^+(v) + d^-(v))$ over all nodes v that are visited or settled from both directions.

Search-Space Minimal Reach. In case shortest paths are not unique the technique still computes correct distances even if only considering one shortest path for each source-target pair. The Problem MinReach is that of choosing these shortest paths such that the resulting average search-space becomes minimal. More formally, we choose a set $\mathcal{P}$ of shortest paths and compute $\mathcal{R}(v)$ by $\max_{\{P \in \mathcal{P} \mid v \in P\}} \mathcal{R}_P(v)$. We denote by $V_{\mathcal{P}}(s,t)$ the search space of the s-t-reach-query using this reach-values for pruning.

*Problem (*MinReach*).* Given a graph $G = (V, E, len)$, choose $\mathcal{P} \subseteq \mathrm{SP}$ such that $\mathcal{P}$ contains at least one shortest s-t path for each pair of nodes $s, t \in V$ for which there is an s-t-path and such that $\sum_{s,t \in V} V_{\mathcal{P}}(s,t)$ is minimal.

Theorem 1. *Problem* MinReach *is NP-hard (even for directed acyclic graphs).*

Proof. We make a reduction from Exact Cover by 3-Sets (X3C). W.l.o.g we may assume $\bigcup_{c \in C} = U$. Given an X3C-instance (U, C) with $|U| = 3q$ we construct a MinReach-instance $G = (V, E)$ as follows: $V = \{a\} \cup C \cup U$ where a is an additional vertex. There is an edge (a, c) for each $c \in C$. There is an edge $(c, u) \in C \times U$ if, and only if $u \in c$. All edge lengths are 1. The construction is polynomial, see Figure 1 for a visualization.

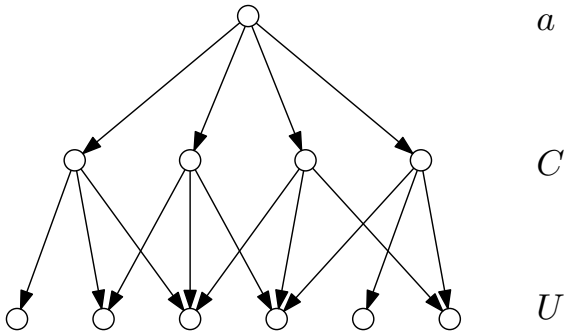


Fig. 1: Graph G constructed from the X3C-instance $\{1,2,3\}, \{2,3,4\}, \{3,4,6\}, \{4,5,6\}$

It is $\mathcal{R}(u) = 0$ for $u \in \{a\} \cup U$. Hence, these nodes only get settled as start nodes from the forward search or as target nodes from the backward search. Given the set $\mathcal{P}$, we denote the search space starting at node z by $V^+_{\mathcal{P}}(z)$ and $V^-_{\mathcal{P}}(z)$ for forward and backward search, respectively. We decompose

$$\sum_{s,t \in V} |V_{\mathcal{P}}(s,t)| = |V| \left(|V^+(a)| + \underbrace{\sum_{s \in U \cup C} |V^+(s)|}_{=|U \cup C|} + \underbrace{\sum_{t \in \{a\} \cup C} |V^-(t)|}_{=|C|+1} + \sum_{t \in U} |V^-(t)| \right)$$

Claim. There is a set $\mathcal{P}$ such that $\sum_{s,t \in V} V_{\mathcal{P}}(s,t) \leq |V|\,[(1+q) + |U \cup C| + |C| + 1 + 2|U|]$ if and only if there is an exact cover for (U, C).

"if" When computing the reach-values of nodes in C we only have to consider paths that start with a and end in U because paths consisting of only one edge do not contribute to reach-values greater than 0. Let $C' \subseteq C$ be an exact cover of (U, C). Further let $C'(u)$ denote the $c' \in C'$ with $u \in c'$. We set $\mathcal{P} = \{(a, C'(u), u) |\ u \in U\}$. Then, for each $c \in C$ we have $\mathcal{R}(c) = 1$ if there is a path (a, c, u) in $\mathcal{P}$ and $\mathcal{R}(c) = 0$ otherwise. Hence, $|V^+(a)| = 1 + q$ and $|V^-(u)| = 2$ for each $u \in U$. This yields the claimed bound.

"only if" Let $\mathcal{P} \subseteq \text{SP}$ be such that $\sum_{s,t \in V} V_{\mathcal{P}}(s,t) \leq |V|\,[(1+q) + |U \cup C| + |C| + 1 + 2|U|]$. We show that $C' = \{c \in C |\ (a, c, u) \in \mathcal{P}\}$ is an exact cover of (U, C). As $\mathcal{P}$ has to include one shortest a-u path for each $u \in U$ we know C' covers of U. With the above decomposition of the search-space we know that $|V^+(a)| + \sum_{t \in U} |V^-(t)| \leq 1 + q + 2|U|$. It is $V^+(a) = \{a\} \cup \{c \in C |\ \mathcal{R}(c) \geq 1\} = \{a\} \cup C'$ and, for $u \in U$, $V^-(u) = \{u\} \cup \{c \in C |\ \mathcal{R}(c) \geq 1, u \in c\} = \{u\} \cup \{c \in C |\ (a, c, u) \in \mathcal{P}\} \geq 2$. Hence $|C'| \leq q$.

External Shortcuts for Reach-Based Pruning. This is an enhancement for reach-based pruning similar to problem ExtShortcutsArcFlags. We assume that, given the input graph G and a parameter k, we are allowed to insert a set $\mathcal{S}$ of k shortcuts to G. The resulting graph G' is the input of the search technique MinReach and we denote the resulting search-space of an s-t-query by $V_{\mathcal{S}}(s,t)$. One can solve the MinReach-part of the preprocessing-phase by a heuristic approach. In that case, one can show the NP-hardness of inserting shortcuts for a wide range of strategies. We show that it is NP-hard to insert the shortcuts even if we are given an oracle that optimally solves problem MinReach in constant time.

*Problem (*ExtShortcutsReach*).* Given a graph $G = (V, E, len)$ and a positive integer k, insert a set $\mathcal{S}$ of k shortcuts to G, such that $\sum_{s,t \in V} V_{\mathcal{S}}(s,t)$ is minimal.

Theorem 2. *Problem* ExtShortcutsReach *is NP-hard (even for directed acyclic graphs and even if there is an oracle that solves Problem* MinReach *in constant time).*

Proof. We make a reduction from X3C. Let (U, C) be an instance of X3C with $|U| = 3q$. W.l.o.g we may assume $\bigcup_{c \in C} = U$. We construct an instance $(G = (V, E, len), k = q)$ of ExtShortcutsReach as follows: the set V consists of two nodes c^- and c^+ for each $c \in C$, one node u for each $u \in U$, one additional node a and an additional set M with $|M|$ to be specified later. There is an edge (c^+, u) with length 2 iff $u \in c$. Further, there are edges (a, c^-) with length 2 and (c^-, c^+) with length 1 for each $c \in C$. Moverover, there is an edge (m, a) with length 1 for each $m \in M$. We set $k = q$. The transformation is polynomial as $|M|$ will be polynomial in the input size, see Figure 2 for a visualization.

Given the set $\mathcal{S}$, we denote the search space starting at node z by $V^+_{\mathcal{S}}(z)$ and $V^-_{\mathcal{S}}(z)$ for forward and backward search, respectively. It is $\mathcal{R}(a) \leq 1$ and $\mathcal{R}(m) = 0$ for $m \in M$. Hence

$$\sum_{s,t \in V} V_{\mathcal{S}}(s,t) = |V| \underbrace{\sum_{z \in \{a\} \cup C^- \cup C^+ \cup U} \left(V^-(z) + V^+(z)\right)}_{\leq 2|\{a\} \cup C^- \cup C^+ \cup U|^2} + |V| \sum_{z \in M} (\underbrace{V^-(z)}_{=|1|} + V^+(z))$$

We call a shortcut assignment $\mathcal{S}$ *set covering* if $\mathcal{S}$ contains, for each $u \in U$, a shortcut (a, c^+) such that $u \in c$.

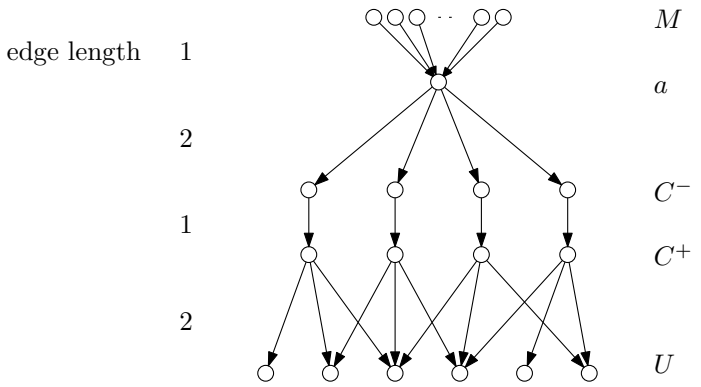


Fig. 2: Graph G constructed from the X3C-instance $\{1,2,3\},\{2,3,4\},\{3,4,6\},\{4,5,6\}$

Claim. Let $\mathcal{S}$ be set-covering. Then $\mathcal{P}$ can be chosen such that $\sum_{m\in M} V^+_{\mathcal{S}}(m) \leq 2|M|$.

Let m be in M. It is $\mathcal{R}(u) = 0$ for $u \in U$. Hence $V^+(m) \cap U = \emptyset$. Further, for $c^+ \in C^+$ we have $\mathcal{R}(c^+) \leq 2$ and $\text{dist}(m, c^+) = 4$. Hence $V^+(m) \cap C^+ = \emptyset$. As $\mathcal{S}$ is set-covering, we can choose $\mathcal{P}$ such that paths starting in $M \cup \{a\}$ and ending in U do not to contain a node in C^-. If we do so, $\mathcal{R}(c^-) \leq 2 < \text{dist}(m, c^-)$ for c^- in C^-. Hence $V^+(m) \subseteq \{m, a\}$.

Claim. Assume $|M| > k$. Let $\mathcal{S}$ and $u^* \in U$ be such that $(a, u^*) \notin \mathcal{S}$ and such that for all $c \in C$ with $u^* \in c$, we have $(a, c^+) \notin \mathcal{S}$. Then $\sum_{m\in M} V^+_{\mathcal{S}}(m) \geq 3|M|$.

As $|M| > k$ we have at least one node $m' \in M$ such that $(m', v) \notin \mathcal{S}$ for all $v \in V$. Hence, (m', a, c^-) is the only shortest path for $c^- \in C$ and $\mathcal{R}(a) \geq 1$. Therefore a gets settled from each $m \in M$. Further, a shortest m'-u^*-path starts with (m', a, c^-_*) for a $c^-_* \in C^-$. Hence, $\mathcal{R}(c^-_*) \geq 3$ and c^-_* gets settled from all $m \in M$.

Claim. We specify $|M| = \max\{k, 2|\{a \cup C^- \cup C^+ \cup U\}|^2\} + 1$. Then $\sum_{s,t\in V} V_{\mathcal{S}}(s,t) \leq |V|(2|\{a \cup C^- \cup C^+ \cup U\}|^2 + |M| + 2|M|)$ if and only if there is an exact cover for (U, C).

Let C' be an exact cover of (U, C). Then $\{(a, c^+) |\ c \in C'\}$ is set-covering and the bound on the search-space holds with the above claims. On the other hand let $\sum_{s,t\in V} V_{\mathcal{S}^*}(s,t) \leq |V|(2\cdot|\{a\}\cup C^-\cup C^+\cup U|^2+|M|+2|M|)$. With the last claim we know that for each $u \in U$ there must be either a shortcut $(a, u) \in \mathcal{S}^*$ or a shortcut (a, c^+) with $u \in c$. We gain a shortcut assignment $\mathcal{S}'$ out of $\mathcal{S}^*$ by copying all shortcuts of the form (a, c^+) in $\mathcal{S}$ and taking, for each shortcut of the form $(a, u) \in \mathcal{S}^*$, one arbitrary shortcut (a, c^+) with $u \in c$. The set $\{c|\ (a, c^+) \in \mathcal{S}'\}$ is a cover of (U, C) of size q.

4 Highway Node Routing (Min-Overlay Graph)

Given the input graph $G = (V, E)$ this technique chooses a sequence $V := V_0 \supseteq V_1 \supseteq \ldots \supseteq V_L$ of sets of nodes. Then, a sequence $(G_0, G_1, \ldots, G_L)$ of graphs is computed which is defined by $G_0 := G$ and for each $l > 0$ by $G_l = (V_l, E_l)$ with

$$E_l := \{(s,t) \in V_l \times V_l \mid \forall \text{shortest } s\text{-}t\text{-paths } P = (s, u_1, u_2, \ldots, u_k, t) \text{ in } G_{l-1} \text{ it is } u_1, \ldots, u_k \notin V_l\}.$$

The length of an edge (u, v) in G_{i+1} is the length of a shortest u-v-path in G_i. Note that, given nodes $u, v \in V_1$ for which the the edge (u, v) is the only shortest u-v path in G_0, (u, v) is contained in E_1. The *level* i of a node v, is the highest index i such that $v \in V_i$. The *multi-level overlay graph* $\mathcal{G}$ is given by $\mathcal{G} = (V, E_1 \cup \ldots \cup E_L)$.

The query is a modified distance-balanced bidirectional DIJKSTRA's algorithm in $\mathcal{G}$. From a node of level i, only edges in $E_i \cup \ldots \cup E_L$ are relaxed. The forward (backward) search is aborted when all keys in the forward (backward) priority queue are greater than or equal to the tentative distance. For the NP-completeness proof, we restrict to 2-level min-overlay graphs:

*Problem (*HIGHWAYNODEPREPROCESS*).* Given a graph $G = (V, E)$ and an integer $F \geq |E|$, choose $V_1 \subseteq V$, such that $|E \cup E_1| \leq F$ and $\sum_{s,t \in V} V_{V_1}(s,t)$ is minimal.

We observe that a feasible solution always must exist as we could choose $V_1 = V$.

Theorem 3. *Problem* HIGHWAYNODEPREPROCESS *is NP-hard.*

Proof. We make a reduction from Exact Cover by 3-Sets (X3C). Given an instance (U, C) of X3C with $|U| = 3q$, we construct an instance $(G = (V, E), F = |E|)$ of HIGHWAYNODEPREPROCESS as follows. The set V consists of a node b, one node c for each $c \in C$, one node u for each $u \in U$ and a set M_v of M additional nodes for each node v in $\{b\} \cup U$ where M will be specified later. We set $D := \bigcup_{v \in \{b\} \cup U} M_v$. For each $c \in C$, there is a directed edge (b, c). For each $u \in U$ and each $c \in C$, such that $u \in c$, there is a directed edge (c, u). Finally, for each $v \in \{b\} \cup U$ and each $w \in M_v$, there is an undirected edge $\{v, w\}$. All edges have length 1, except edges from b to C which have length 2 and the edges from C to U which have length 10. The transformation is polynomial as M will be polynomial in the input size (see Figure 3 for a visualization). Note that, as $F = |E|$, no new edges may be introduced to the overlay-graph, i.e. E_1 is empty.

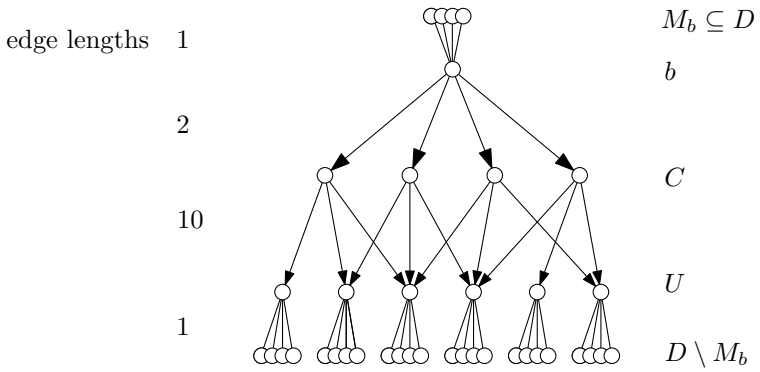


Fig. 3: Graph G constructed from the X3C-instance $\{1,2,3\}, \{2,3,4\}, \{3,4,6\}, \{4,5,6\}$, some nodes in D are omitted

Requisite. Throughout the remaining proof, we write $X := (|U| + |C| + 2)$ and assume $M > 2^5 X^5$. We will see that, under this assumption, the maximum search space of an s-t-query is at most $2X$ (when V_1 is chosen optimally).

Claim. Let V_1 be an optimal solution, then $\{b\} \cup U \subseteq V_1$.

If $\{b\} \cup U \subseteq V_1$ and $D \cap V_1 = \emptyset$, then $\sum_{s,t \in V} V_{V_1}(s,t) \leq 2X|V|^2$ as, in this case, for each s-t-pair and for forward and backward search at most the start node and all nodes in $\{b\} \cup C \cup U$ get settled. We now show that $\sum_{s,t \in V} V_{V_1}(s,t) > 2X|V|^2$ if there is a $v \in \{b\} \cup U$ with $v \notin V_1$.

If $\sum_{s,t \in V} V_{V_1}(s,t) \leq 2X|V|^2$, then $\{b\} \cup U \subseteq V_1$: Assume that $b \notin V_1$. Then $|M_b \cap V_1| \leq 1$ as $F = |E|$. It is

$$\sum_{s \in M_b, t \in D \setminus M_b} V_{V_1}(s,t) \geq \underbrace{(M-1)}_{\text{sources}} \cdot \underbrace{|U|M}_{\text{targets}} \cdot \underbrace{M}_{\text{search space}}$$

as for each according query all nodes in M_b get settled. Assume that $u \in U$ is such that $u \notin V_1$. Then $|M_u \cap V_1| \leq 1$ as $F = |E|$. It is

$$\sum_{s \in M_b, t \in M_u} V_{V_1}(s,t) \geq \underbrace{M}_{\text{sources}} \cdot \underbrace{(M-1)}_{\text{targets}} \cdot \underbrace{M}_{\text{search space}}$$

as the backward search settles all nodes in M_u. The claim follows with $M > 2^5 X^5$ from

$$M^2(M-1) \geq (M/2)^3 > 2X(2X^2M)^2 \geq 2X(|C| + |U| + 1 + (M+1)|U|)^2 = 2X|V|^2$$

which contradicts the optimality of V_1.

Claim. Let V_1 be an optimal solution. Then $V_1 \setminus D$ is also an optimal solution.

We already know that $\{b\} \cup U \in V_1$. Given a solution V_1, we show that we can iteratively remove elements v of D from V_1 without increasing the search-space. Let v be a node with only one neighbour w. Let the edge (v,w) be undirected, i.e. $len(v,w) = len(w,v)$ and let $v, w \in V_1$. Further, let v have minimal distance from w and be maximal among all nodes in V_1 with minimum distance from w (remember that we uniquely identify vertices with integers). We show that, for each $s, t \in V$ it holds $V_{V_1}(s,t) \geq V_{V_1 \setminus \{v\}}(s,t)$.

This is clear if $s, t \neq v$. Now, let w.l.o.g $s = v$. Note, that both search directions are finished, as soon as s gets settled by the backward or t gets settled by the forward direction. We denote the search with $v \in V_1$ by σ and the other one by σ'. The size of the according search spaces is denoted by $|\sigma|$ and $|\sigma'|$.

Let $t \neq w$ and assume that $|\sigma| \neq |\sigma'|$. The searches are equal up to the point at which the backward part of σ settles v. As the search is not finished at that moment, the forward part of σ may not have yet settled w. Hence, $\text{dist}(t,v) \leq dist(v,w)$ which contradicts the assumption that $t \neq w$. It follows $\sigma = \sigma'$ for $s = v$ and $t \neq w$.

Let $t = w$. Again, the searches are equal up to the point where the backward part of σ settles v. In σ the computation is finished after this step. In σ', as v is maximal among all neighbours of w with minimal distance that are in V_1, the search alters direction, settles w by the forward direction and is also finished. Hence, $|\sigma| = |\sigma'|$.

Note that this claim can also be proven slightly simpler using a similar argument as the last claim.

Claim. Let V_1 be an optimal solution. Then, for each $u \in U$, there is at least one $c \in C$ with $u \in c$ and $c \in V_1$.

Recall that $\{b\} \cup U \subseteq V_1$. Assume, that there is a u with $\{(c \in C \cap V_1 |\ (c,u) \in E\} = \emptyset$. Then, the set E_1 would contain the edge (u,b) which contradicts $F = |E|$.

Claim. Let $M > 2X^3 + 4(|U|+1)X^2)$ and V_1 be an optimal solution. Then, there is an integer B, such that $\sum_{s,t \in V} V_{V_1}(s,t) \leq B$ if and only if (C,U) has an exact cover.

Let V_1 be an optimal solution with $V_1 \cap D = \emptyset$. It is (with some abuse of notation)

$$\sum_{s,t \in V} V_{V_1}(s,t) = \Big(\underbrace{\sum_{s,t \in V \setminus D}}_{\leq \alpha} + \underbrace{\sum_{\substack{s \in V \setminus D, t \in D \\ s \in D, t \in V \setminus D}}}_{\leq \beta} + \underbrace{\sum_{\substack{u \in \{b\} \cup U \\ s,t \in M_u}} + \sum_{s \in D \setminus M_b, t \in M_b}}_{=:\gamma} + \underbrace{\sum_{\substack{a,c \in U, a \neq c \\ s \in M_a, t \in M_c}}}_{=:\delta_1} + \underbrace{\sum_{s \in M_b, t \in D \setminus M_b}}_{=:\delta_2} \Big) V_{V_1}(s,t)$$

The bounds $\alpha := 2X^3$ and $\beta := 2(|U|+1)MX \cdot 2X$ derive from multiplying the according number of sources, number of targets and maximum search-space.

The value of γ is independent of the choice of V_1 (if V_1 sticks to the structure ensured by the above claims) and computable in polynomial time (and of polynomial size). This is due to the fact, that in this case, for all s-t pairs, the according forward and backward search-space have size 2. Further, if $C \cap V_1$ is an exact cover of U,

$$\delta_1 = \delta_1' := \underbrace{|U|M}_{\text{source}} \cdot \underbrace{(|U|-1)M}_{\text{target}} \cdot (\underbrace{2}_{\text{forward search}} + \underbrace{4}_{\text{backward search}})$$

$$\delta_2 = \delta_2' := \underbrace{M}_{\text{source}} \cdot \underbrace{|U|M}_{\text{target}} \cdot (\underbrace{2+|U|+q}_{\text{forward search}} + \underbrace{4}_{\text{backward search}})$$

and

$$\delta_1 \geq \delta_1' + M^2$$
$$\delta_2 \geq \delta_2'$$

otherwise. Concluding, we set $B := \alpha + \beta + \gamma + \delta_1 + \delta_2$. With the assumption $M > 2X^3 + 4(|U|+1)X^2)$ follows $\alpha + \beta + \gamma + \delta_1 + \delta_2 < \gamma + \delta_1 + \delta_2 + M^2$ which proves the claim.

5 ALT

Goal-directed search is a variant of Dijkstra's algorithm which assigns a different priority to the nodes in the queue. For a node v, let $p(v)$ denote the priority of v in the queue when applying Dijkstra's algorithm (that means $p(v)$ is the tentative distance to v). Goal-directed search adds a potential $\Pi_t(v)$ depending on the target t to the nodes priority, i.e. the priority of v (when applying goal-directed search) is $\Pi_t(v) + p(v)$. An equivalent formulation (that implicitely substracts the constant $\Pi_t(s)$ from the priority) is as follows. Given the potential function $\Pi_t : V \to \mathbb{R}^+$, goal-directed search is Dijkstra's algorithm applied on G with altered edge lengths $\overline{len}(u,v) = len(u,v) - \Pi_t(u) + \Pi_t(v)$ for all edges $(u,v) \in E$. A potential is called *feasible* if $\overline{len}(u,v) \geq 0$ for every edge (u,v). The ALT-algorithm [10] is goal-directed search with a special potential function. Initially, a set $L \subset V$ of 'landmarks' is chosen. For a landmark $l \in L$ we define

$$\Pi_t^{l+}(v) := \text{dist}(v,l) - \text{dist}(t,l) \tag{1}$$
$$\Pi_t^{l-}(v) := \text{dist}(l,t) - \text{dist}(l,v) \tag{2}$$

We use the convention $\infty - \infty := 0$. Accordingly, for a set L of landmarks, the potential is

$$\Pi_t^L(v) := \max_{l \in L} \left\{ \Pi_t^{l+}(v), \Pi_t^{l-}(v), 0 \right\}.$$

Note that this potential is feasible and that $\Pi_t^L(t) = 0$. We denote by $V_L(s,t)$ and $V_\Pi(s,t)$ the search space of an s-t-ALT-query using landmarks L and potential Π, respectively.

The search space of an ALT-query can be expressed as

$$\begin{aligned} V_L(s,t) = \{ v \in V \mid & \text{dist}(s,v) + \Pi_t^L(v) < \text{dist}(s,t) \text{ or} \\ & \text{dist}(s,v) + \Pi_t^L(v) = \text{dist}(s,t) \text{ and } v < t \} \end{aligned}$$

The next lemma shows that, when using landmarks, the potential of a node is a lower bound of its distance to the target.

Lemma 1. *Let Π be a feasible potential, t be a vertex and $\Pi(t) = 0$. Then, for any vertex v, $\Pi(v) \leq \text{dist}(v,t)$ holds.*

Proof. Let $v = v_1, v_2, \ldots, v_k = t$ be a shortest v-t-path. Because of $len(v_i, v_{i+1}) - \Pi(v_i) + \Pi(v_{i+1}) \geq 0$ for each $i = 1, \ldots, k-1$ it holds $\sum_{i=1}^{k-1} len(v_i, v_{i+1}) \geq \sum_{i=1}^{k-1} \Pi(v_i) - \Pi(v_{i+1})$. This implies $\text{dist}(v,t) \geq \Pi(v) - \Pi(t) = \Pi(v)$.

Goldberg et al [10] show that stronger potentials result in tighter search spaces:

Lemma 2 (Goldberg et al). *Given arbitrary nodes s, t and feasible potential functions Π and Π' with $\Pi(t) = \Pi'(t) = 0$ and $\Pi(v) \leq \Pi'(v)$ for any vertex v. Then $V_{\Pi'}(s,t) \subseteq V_\Pi(s,t)$.*

Proof. We show that from $v \notin V_\Pi(s,t)$ it follows $v \notin V_{\Pi'}(s,t)$. Since v gets not settled using Π it holds $\text{dist}(s,v) + \Pi(v) > \text{dist}(s,t) + \Pi(t)$ or $(\text{dist}(s,v) + \Pi(v) = \text{dist}(s,t) + \Pi(t)$ and $v > t)$. Because of the assumptions it follows that $\text{dist}(s,v) + \Pi'(v) > \text{dist}(s,t) + \Pi'(t)$ or $(\text{dist}(s,v)) + \Pi'(v) = \text{dist}(s,t) + \Pi'(t)$ and $v > t)$ which means that v does not get settled using Π'.

The problem MinALT is that of assigning a given number of landmarks to a graph (and thus using only a given amount of preprocessing space), such that the expected number of settled nodes gets minimal.

*Problem (*MinALT*).* Given a directed graph $G = (V, E, len)$ and an integer r, find a set $L \subset V$ with $|L| = r$ such that $\sum_{s,t \in V} V_L(s,t)$ is minimal.

Theorem 4. *Problem* MinALT *is NP-hard.*

Proof. We make a reduction from the NP-complete problem 3MinimumCover [8].

*Problem (*3MinimumCover*).* Given a collection C of subsets of a finite set S with $|c| \leq 3$ for each $c \in C$ and a positive integer k. Does C contain a cover for U of size k, i.e. is there a subset $C' \subseteq C$ with $|C'| = k$ and $\bigcup_{c \in C'} c = U$.

We say, a *set* $c \in C$ *covers* an *element* $u \in U$ if $u \in c$. As a preparatory step, we may assume that $|c| = 3$ for each set c and that, for each element u, there is a set c that does not cover u. To assure this, we first remove each element that is contained in every set, as such elements do not affect the solvability of the instance. In case each set c of the remaining instance has size of at most 2, we can solve the problem in polynomial time [8]. Otherwise, we transform the resulting instance (C', U', k') to a new instance (C, U, k) by setting $k = k' + 1$, $U = U' \cup \{x, y, z\}$ where x, y, z are new elements and $C = \{x, y, z\} \cup \{c \mid c \in C', |c| = 3\} \cup \{c \cup \{x\} \mid c \in C', |c| = 2\} \cup \{c \cup \{x, y\} \mid c \in C', |c| = 1\}$. This instance fulfills our claims.

Now, we construct an instance $(G = (V, E, len), r)$ of MinAlt the following way: We introduce a set $M = \{m_1, \ldots, m_{|M|}\}$ of nodes, with $|M|$ to be specified later and set $V = C \cup U \cup M$. There is an edge (c, u) with length 1 for each $u \in c$. Moreover, there is an edge (u_i, u_j) with length $\epsilon = 0.5$ for each pair of elements $u_i, u_j \in U$, an edge (u, m) with length 1 for each pair $u \in U, m \in M$ and an edge (m_i, m_{i+1}) for each $i = 1, \ldots, |M| - 1$. Nodes are ordered such that $c_1 \leq \ldots \leq c_{|C|} \leq u_1 \leq \ldots \leq u_{|U|} \leq m_1 \leq \ldots \leq m_{|M|}$. The number of landmarks r is set to be $k + 1$. The transformation (and the following computation of M is polynomial).

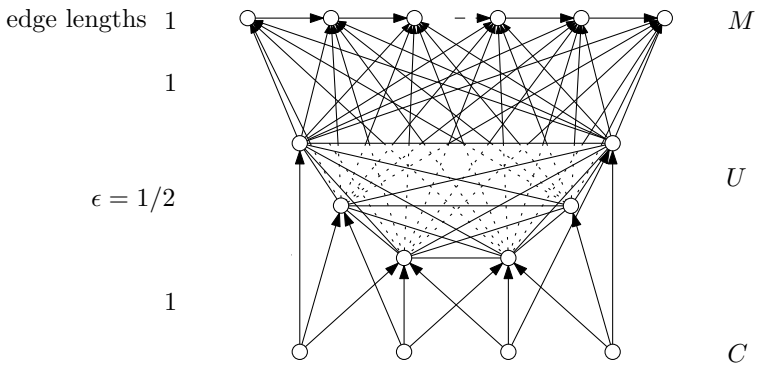


Fig. 4: Graph G constructed from the 3MinimumCover-instance $\{1, 2, 3\}, \{2, 3, 4\}, \{3, 4, 6\}, \{4, 5, 6\}$

Let L be a set of landmarks. Then

$$\sum_{s,t \in V} V_L(s,t) = \underbrace{\sum_{s \in V, t \in C} V_L(s,t) + \sum_{s \in M, t \in U \cup M} V_L(s,t)}_{=:\alpha} + \underbrace{\sum_{s \in U \cup C, t \in U} V_L(s,t)}_{\leq \beta} + \underbrace{\sum_{s \in U \cup C, t \in M} V_L(s,t)}_{=:\gamma(L)}$$

Claim 1. The value of α is independent of the choice of L and can be computed in polynomial time.

For the cases $s \in V, t \in C$ and $s \in M, t \in U$, the target t is not reachable from the source s (unless $s = t$). Therefore, in these cases, the search space is exactly the number of nodes that are reachable from s (or 1 if $s = t$). If $s, t \in M$, either the target is again not reachable from the source or the search space is a direct path to the target.

Claim 2. It holds $0 \leq \sum_{s \in U \cup C, t \in U} V_L(s,t) \leq \beta := |U||C|(1 + |U|) + |U|^3$.

In case $s \in C$, it holds $\text{dist}(s,t) + \Pi(t) = \text{dist}(s,t) \leq 1 + \epsilon < 2 = \text{dist}(s,m)$ for $m \in M$. Hence, for each of the $|C||U|$ s-t-pairs in $C \times U$ at most $1 + |U|$ nodes get settled. In case $s \in U$, it holds $\text{dist}(s,t) + \Pi(t) =$

$\text{dist}(s,t) \leq \epsilon < 1 = \text{dist}(s,m)$ for $m \in M$. Hence, for each of the $|U|^2$ s-t-pairs in $U \times U$ at most $|U|$ nodes get settled.

We call a set L of landmarks *set-covering* if $m_1 \in L$ and for each $u \in U$, either u or a node $c \in C$ with $u \in c$ is contained in L.

Claim 3. Let $L \subseteq V$ be set-covering. Then $\sum_{s \in U \cup C, t \in M} V_L(s,t) \leq |M|(5|C| + 2|U|)$.

Remember that we identify nodes by integers. It is $V_L(s,t) \cap M = \{t\}$ because $\text{dist}(s,m)$ is equal for all $m \in M$ and $\Pi_t^L(m) > 0$ for $m < t, m \in M$. As each element $u \in U$ is covered by a landmark l, we have $\Pi_t^L(u) \geq \text{dist}(l,t) - \text{dist}(l,u) = 2 - 1 = 1$ for those u. Hence, for $s \in U$, $V_L(s,t) = \{s,t\}$ and for $s \in C$, $V_L(s,t) = \{s,t\} \cup \{u \in U \mid s \text{ covers } u\}$.

Claim 4. Let $s \in C, t \in M$ and Π_t be a feasible potential with $\Pi_t(t) = 0$, then $V_{\Pi_t}(s,t) \geq 5$ and $V_{\Pi_t}(s,t) \supseteq \{s,t\} \cup \{u \in U \mid s \text{ covers } u\}$.

Let Π_t be the potential such that $\Pi_t(v) = dist(v,t)$. Then, $V_{\Pi_t}(s,t) = \{s,t\} \cup \{s_i \in S \mid s \text{ covers } s_i\}$. The claim now follows from Lemmata 1 and 2.

Claim 5. Let L be a set of landmarks with $M \cap L = \emptyset$. Then $\sum_{s \in C \cup U, t \in M} V_L(s,t) \geq (|C|+|U|)|M|(|M|-1)/2$.

If there is no landmark in M, then all nodes in M have the same potential due to symmetry reasons. Hence, before settling a node $t \in M$, all nodes $m \in M$ with $m < t$ are getting settled. The claim follows by summing over all possible sources and targets.

Claim 6. Let L be a set of landmarks such that there is an $x \in U$ that is not a landmark and not covered by a landmark. Then $\sum_{s \in U \cup C, t \in M} V_L(s,t) \geq |M|(5|C| + 2|U|) + |M| - |L|$.

Let s be in U. Then $|V_L(s,t)| \geq 2$ as source and target differ. Let s be in C, by Claim 4 we know that $|V_L(s,t)| \geq 5$. Because of the preparatory step, we know that there is a set $c_x \in C$ with $x \notin c_x$. Furthermore, there are at least $|M| - |L|$ nodes in M that are no landmarks. Let t_x denote such a node. For a landmark $l \in C$, $\Pi_{t_x}^l(x) = 1 - \epsilon$. For a landmark $l \in U$, $\Pi_{t_x}^l(x) = 1 - \epsilon$. For a landmark $l \in M$, $\Pi_{t_x}^l(x) = 0$ as t_x is not a landmark. Therefore $x \in V_L(c_x, t_x)$ and $|V_L(c_x, t_x)| \geq 6$. The claim now follows together with Claim 4 by summing over all possible sources and targets.

Claim 7. Let $|M|$ be $\max\{\beta + r, 2(\beta + 5|C| + 2|U| + 1)\} + 1$ and L be optimal. Then $\sum_{s,t \in V} V_L(s,t) \leq \alpha + \beta + |M|(5|C| + 2|U|) =: q$ if and only if (C,U) has a cover of size at most k.

We observe that there is a set-covering set of landmarks of size $k+1$ if and only if (C,U) admits a set-cover of size k. Because of Claim 3 a set-covering set of landmarks fulfills $\sum_{s,t \in V} V_L(s,t) \leq q$. Let L be a set of landmarks such that $\sum_{s,t \in V} V_L(s,t) \leq q$. Because of Claims 5 and 6 $L' = \{m_1\} \cup (L \setminus M)$ is set-covering and of size at most $k+1$.

6 Arc-Flags

Main Technique. This approach partitions the set of nodes V into k cells $\mathcal{V} = (V_1, V_2, \ldots, V_k)$. For a node w, we write $\mathcal{V}(w) = V_i$ iff $w \in V_i$. To each edge (u, v) a k-bit vector $\mathcal{F}_{(u,v)}$ is attached, such that $\mathcal{F}_{(u,v)}(V_i)$ is true iff a shortest path starting with the edge (u, v) and ending at a node $t \in V_i$ exists. The Arc-Flags s-t-query is the variant of Dijkstra's algorithm which only relaxes edges (u, v) for which $\mathcal{F}_{(u,v)}(\mathcal{V}(t))$ is true.

*Problem (*ArcFlags*).* Given a graph $G = (V, E, len)$ and an integer k, find a k-partition $\mathcal{V} = \{V_1, \ldots, V_k\}$ of V such that $\sum_{s,t \in V} V_{\mathcal{V}}(s, t)$ is minimal.

In order to prove NP-completeness for ArcFlags we will use the following technical lemmata.

Lemma 3. *Given $L, m \in \mathbb{Z}_{>1}$, we consider the problem (P) of how to minimize $\sum_{i=1}^{m} c_i(c_i+1)/2$ such that $c_1 + c_2 + \ldots + c_m = Lm$ and $c_1, c_2, \ldots, c_m \in \mathbb{Z}^+$.*

The only optimal solution of (P) is $c_1^* = c_2^* = \ldots = c_m^* = L$ with objective value $D = mL(L+1)/2$. For any other feasible solution $c_1', c_2', \ldots, c_n'$ it is $\sum_{i=1}^{m} c_i'(c_i'+1)/2 \geq D + 1$.

Proof. Given $c' = (c_1', c_2', \ldots, c_m')$ with $c_1' + c_2' + \ldots + c_m' = Lm$, we can construct c' by starting with $c = (c_1 = L, c_2 = L, \ldots, c_m = L)$ and iteratively proceeding as follows. At each step we choose some i and j fullfilling $c_i > c_i'$, $c_j < c_j'$ and set $c_i := c_i - 1$ and $c_j := c_j + 1$ until no such i and j exist anymore.

We define $\Delta^+ := \sum_{i | c_i' > c_i} c_i' - c_i$ and $\Delta^- := \sum_{i | c_i' < c_i} c_i - c_i'$. Throughout the construction we have the invariant $\Delta^+ = \Delta^-$. Hence, $\Delta^+ = \Delta^- = 0$ and c' is constructed when the algorithm terminates (which is guaranteed due to monotonicity Δ^+). Further, throughout the construction we have $c_i \leq c_j$ if $c_i > c_i'$ and $c_j < c_j'$ as a second invariant. Finally, when performing a step, the objective value increases by

$$1/2\,[(c_i - 1)c_i + (c_j + 1)(c_j + 2) - c_i(c_i + 1) - c_j(c_j + 1)] \geq 1$$

which proves the lemma.

Lemma 4. *Given are a set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$ and a size w_a for each $a \in A$ such that $B/4 < w_a < B/2$ and such that $\sum_{a \in A} w_a = mB$. Consider the problem (P') of how to partition A in m disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{i=1}^{m} \left(|A_i| \cdot \sum_{a \in A_i} w_a\right)$ is minimal. A solution of (P') has to fullfill $|A_i| = 3$ for each $i = 1, \ldots m$ and has objective value $D = 3\sum_{a \in A} w_a = 3mB$. Further, for each other partition of A the objective value is at least $D + 1$.*

Proof. Analogous to the proof of Lemma 3 we may assume that there is a starting partition $A' = (A_1', A_2', \ldots A_m')$ with $|A_i'| = 3$ such that we can construct an arbitrary partition $A^* = (A_1^*, A_2^*, \ldots A_m^*)$ out of A' by iteratively moving one element from a set A_r with $|A_r| \leq 3$ to a set A_s with $|A_s| \geq 3$. Let $A_r = \{1, \ldots, k\}$ and $A_s = \{k+1, \ldots, k+\ell\}$. When moving element k to A_s the objective function increases by $(k-1)\sum_{i=1}^{k-1} w_i + (\ell+1)\sum_{i=k}^{k+\ell} w_i - k\sum_{i=1}^{k} w_i - \ell\sum_{i=k+1}^{k+\ell} w_i$. This simplifies to $(\ell + 1 - k)w_k + \sum_{i=k+1}^{k+\ell} w_i - \sum_{i=1}^{k-1} w_i > B/4 + 3 \cdot B/4 - 2B/2 = 0$ because of $k \leq 3$ and $\ell \geq 3$ and the bounds on the w_i. The claim follows with the objective function always being integer-valued.

Theorem 5. *Problem* ArcFlags *is NP-hard.*

Proof. We make a reduction from the strongly NP-complete problem 3-Partition [8].

*Problem (*3-Partition*).* Given a set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$ and a size w_a for each $a \in A$ such that $B/4 < w_a < B/2$ and such that $\sum_{a \in A} w_a = mB$, can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} w_a = B$.

Given a 3-Partition-instance $(A, \{w_a |\ a \in A\})$ with $|A| = 3m$ and $B = \sum_{a \in A} w_a/m$, we construct an ArcFlags-instance $(G = (V, E, len), m + 1)$ as follows: We introduce a directed cycle (Z, U) to G with $len(u, v) = 1/(|Z| + 1)$ for each $(u, v) \in U$. The cardinality $|Z|$ will be specified later. We further introduce A to V and for each $a \in A$ a set W_a of w_a nodes. We denote by W the set $\bigcup_{a \in A} W_a$. There is a directed edge (z, w) of length 1 for each $z \in Z$ and $w \in W$ and a directed edge (w, a) of length 1 for each $a \in A$ and $w \in W_a$. A visualization can be seen in Figure 5.

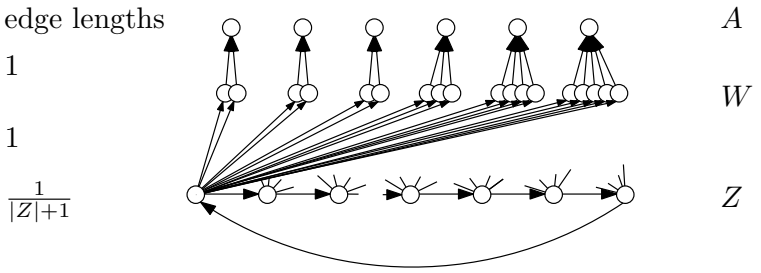


Fig. 5: Graph G constructed from the 3-PARTITION-instance 2,2,2,3,4,5

Claim. Let $Z > 6|A \cup W|^3$. Then there is an optimal $(m+1)$-partition $\mathcal{V} = (V_1, \ldots, V_{m+1})$ with $V_{m+1} = Z$.

Let $\mathcal{V}^*$ be an $m+1$ partition such that there is a $J \subseteq \{1, 2, \ldots, m+1\}$ with $Z = \bigcup_{j \in J} V_j$. Then

$$\sum_{s,t \in V} V_{\mathcal{V}^*}(s,t) = \underbrace{\sum_{s,t \in Z} V_{\mathcal{V}^*}(s,t)}_{=|Z|^2(|Z|+1)/2} + \underbrace{\sum_{s \in Z, t \in A \cup W} V_{\mathcal{V}^*}(s,t)}_{\substack{\leq |Z| \cdot (|A \cup W|+1)^2 \\ \leq |Z| \cdot |A \cup W|^3}} + \underbrace{\sum_{s,t \in A \cup W} V_{\mathcal{V}^*}(s,t)}_{\leq |A \cup W|^3} + \underbrace{\sum_{s \in A \cup W, t \in Z} V_{\mathcal{V}^*}(s,t)}_{=|Z| \cdot |A \cup W|}$$

W.l.o.g let $\max J$ be $(m+1)$. Let $\mathcal{V}^{**} = (V_1^{**}, \ldots, V_{m+1}^{**})$ with $V_i^{**} = V_i^*$ for $i \notin J$, $V_i^{**} = \emptyset$ for $i \in J \setminus \{m+1\}$ and $V_{m+1}^{**} = \bigcup_{j \in J} V_j^*$. Then $\sum_{s,t \in V} V_{\mathcal{V}^*}(s,t) = \sum_{s,t \in V} V_{\mathcal{V}^{**}}(s,t)$ which can be seen by the above decomposition of $\sum_{s,t \in V} V_{\mathcal{V}^*}(s,t)$.

In the other direction, let $\mathcal{V}'$ be an $(m+1)$-partition such that there are vertices $w \in A \cup W$ and $z \in Z$ with $w \in V_i'$ and $z \in V_i'$ for some i. Then

$$\sum_{s,t \in V} V_{\mathcal{V}'}(s,t) \geq \sum_{s,t \in Z} V_{\mathcal{V}'}(s,t) + \sum_{s \in Z} V_{\mathcal{V}'}(s,w) \geq \frac{|Z|^2(|Z|+1)}{2} + \frac{|Z|(|Z|+1)}{2}$$

With the assumption $Z > 6|A \cup W|^3$ we have

$$|Z|(|Z|+1)/2 > |Z| \cdot |A \cup W|^3 + |A \cup W|^3 + |Z| \cdot |A \cup W|$$

which yields $\sum_{s,t \in V} V_{\mathcal{V}^*}(s,t) < \sum_{s,t \in V} V_{\mathcal{V}'}(s,t)$. Hence, for optimal $\mathcal{V}^*$ there must be a $J \subseteq \{1, 2, \ldots, m+1\}$ with $Z = \bigcup_{j \in J} V_j$ and the claim follows with the above restructuring of $\mathcal{V}^*$ to $\mathcal{V}^{**}$.

Requisite. In the remainder we assume $Z > 6|A \cup W|^3$. Further $\mathcal{V}^*$ denotes an optimal partition with $V_{m+1}^* = Z$. We decompose the objective function as follows.

$$\sum_{s,t \in V} V_{\mathcal{V}^*}(s,t) = \underbrace{\sum_{s,t \notin Z} V_{\mathcal{V}^*}(s,t)}_{\leq |W \cup A|^3} + \underbrace{\sum_{s,t \in Z} V_{\mathcal{V}^*}(s,t) + \sum_{s \notin Z, t \in Z} V_{\mathcal{V}^*}(s,t)}_{=:\alpha} + \underbrace{\sum_{s \in Z, t \in A} V_{\mathcal{V}^*}(s,t)}_{=:\beta} + \underbrace{\sum_{s \in Z, t \in W} V_{\mathcal{V}^*}(s,t)}_{=:\gamma}$$

The value of α is independent of $\mathcal{V}^*$ as it equals $|Z|^2(|Z|+1)/2 + |W \cup A||Z|$. We write $A_i := A \cap V_i^*$ and $\overline{W}_i := W \cap V_i$. Further, with $H_i = \{w \in W | w \in W_a \text{ and } a \in V_i\}$ we denote the set of nodes in W that are direct predecessors of a node in $A \cap V_i$. Note that $|H_i|$ is exactly the weight of all elements in $A \cap V_i^*$. With this notation we have

$$\beta = |Z| \sum_{i=1}^{m} \left(|A_i|(|A_i|+1)/2 + |A_i|(|\overline{W}_i \cup H_i|) + 1 \right)$$

$$\gamma \geq |Z| \sum_{i=1}^{m} \left(|\overline{W}_i|(|\overline{W}_i|+1)/2 + 1 \right)$$

Further, $\gamma = |Z| \sum_{i=1}^{m} \left(|\overline{W}_i|(|\overline{W}_i|+1)/2+1\right)$ in case $\overline{W_i} = H_i$ for all i.

Claim. Let $\beta^* := |Z| \sum_{i=1}^{m} (3(3+1)/2+3mB+1)$ and $\gamma^* := |Z| \sum_{i=1}^{m} (B(B+1)/2+1)$. Then $\sum_{s,t\in V} V_{\mathcal{V}^*}(s,t) \leq |W \cup A|^3 + \alpha + \beta^* + \gamma^*$ if, and only if the 3-PARTITION-instance $(A, \{w_a |\ a \in A\})$ is a yes-instance.

"if": Let $\overline{A}_1, \overline{A}_2, \ldots, \overline{A}_m$ be a 3-partition of $(A, \{w_a |\ a \in A\})$. We partition G such that for each $a \in \overline{A}_i$ it holds $a \in V_i$ and $W_a \subseteq V_i$. Then, for each i, $\overline{W}_i = H_i$, $|W_i| = B$ and $|A_i| = 3$. This implies $\sum_{s,t\in V} V_{\mathcal{V}^*}(s,t) \leq |W \cup A|^3 + \alpha + \beta^* + \gamma^*$.

"only if": On the other hand, let $\sum_{s,t\in V} V_{\mathcal{V}^*}(s,t) \leq |W \cup A|^3 + \alpha + \beta^* + \gamma^*$. We will show that $A_1, \ldots, A_m$ is a 3-partition of A. We apply Lemma 3 on $\sum_{i=1}^{m} |A_i|(|A_i|+1)/2$ and $\sum_{i=1}^{m} |\overline{W}_i|(|\overline{W}_i|+1)/2$ and Lemma 4 on $\sum_{i=1}^{m} |A_i||H_i|$. By optimizing these terms separately we know that $\sum_{s,t\in V} V_{\mathcal{V}^*}(s,t) \geq \alpha + \beta^* + \gamma^*$. Further it is $|A_i| = 3$, $|\overline{W}_i| = B$ and $\overline{W}_i \subseteq H_i$ for all i, since otherwise $\sum_{s,t\in V} V_{\mathcal{V}^*}(s,t) \geq \alpha + \beta^* + \gamma^* + Z$. From $\overline{W}_i \subseteq H_i$ follows $\overline{W}_i = H_i$ as both $\overline{W}_1, \ldots, \overline{W}_m$ and $H_1, \ldots H_m$ partition W. Hence $|H_i| = B$ for all $i = 1, \ldots m$ and $A_1, \ldots A_m$ is a 3-Partition of A.

Search-Space Minimal Arc-Flags. This problem models a special aspect of the arc-flag technique. In case shortest paths are not unique, the situation may occur that one can improve the search-space by changing some flags from true to false while still guaranteeing the query to compute the correct distance. We consider the partition $\mathcal{V} = (V_1, V_2, \ldots, V_k)$ of the graph $G = (V, E)$ to be already given and change the rule of how to compute the vectors $\mathcal{F}_{(u,v)}$: For each pair of nodes s and t there shall be at least one shortest path $s = v_1, \ldots, v_\ell = t$ such that $\mathcal{F}_{(v_i,v_{i+1})}(\mathcal{V}(t))$ is true for $i = 1, \ldots, \ell - 1$. We may assume that for each edge (u, v) with $\mathcal{F}_{(u,v)}(V_i) = \mathit{true}$ there is at least one shortest path starting with (u, v) that leads to V_i. The problem MINFLAGS is that of how to assign values to the vectors $\mathcal{F}_{(u,v)}$ such that the resulting average search-space of an arc-flags query becomes minimal.

*Problem (*MINFLAGS*).* Given a graph $G = (V, E, \mathit{len})$, a partition $\mathcal{V} = (V_1, \ldots, V_k)$ of V, compute an arc-flag assignment $\mathcal{F} = (\mathcal{F}_{(u,v)})_{(u,v)}$ for G such that $\sum_{s,t\in V} V_{\mathcal{F}}(s,t)$ is minimal.

Theorem 6. *Problem* MINFLAGS *is NP-hard (even for directed acyclic graphs).*

Proof. We make a reduction from X3C.

*Problem (*EXACT COVER BY 3-SETS (X3C)*).* Given a set U with $|U| = 3q$ and a collection C of 3-element subsets of U, does C contain an exact cover for U, i.e., a subcollection $C' \subseteq C$ such that every element of U occurs in exactly one member of C'?

Let (U, C) be an instance of X3C, w.l.o.g we may assume $\bigcup_{c\in C} = U$. Let $(G = (V, E), \mathcal{V} = \{V_1, V_2\})$ be an instance of MINFLAGS as follows: the set V consists of two nodes c^- and c^+ for each $c \in C$, one node u for each $u \in U$ and one additional node a. The partition is given by $V_2 = U$, $V_1 = V \setminus V_2$. There are edges (a, c^-) and (c^-, c^+) for each $c \in C$, and an edge (c^+, u) iff $u \in c$. All edges have equal length, the node ordering is arbitrary, (see Figure 6 for a visualization). The transformation is polynomial.

Given an arc-flag assignment $\mathcal{F}$, the objective function can be decomposed as

$$\sum_{s,t\in V} V_{\mathcal{F}}(s,t) = \underbrace{\sum_{s\in V\setminus\{a\}, t\in V} V_{\mathcal{F}}(s,t) + \sum_{t\in V\setminus U} V_{\mathcal{F}}(a,t)}_{\alpha} + \sum_{t\in U} V_{\mathcal{F}}(a,t)$$

Claim. The value of α is independent of $\mathcal{F}$ and can be computed in polynomial time.

Shortest paths that start with a node $u \neq a$ are unique. Therefore, an arc-flag assignment for G is relatively fixed: $\mathcal{F}_{(v,w)}(V_1)$ is true iff $w \notin V_2$ and $\mathcal{F}_{(v,w)}(V_2)$ is true if $v \neq a$. The remaining degree of freedom is to decide for arbitrary c^- if $\mathcal{F}(a, c^-)$ is true. For each node $s \neq a$, queries starting from s are not affected by the actual choice of $\mathcal{F}$. Further, queries for which $s, t \in V_1$ are also not influenced by $\mathcal{F}$.

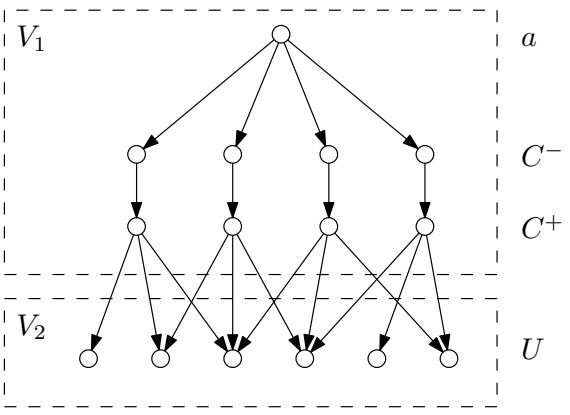


Fig. 6: Graph G constructed from the X3C-instance $\{1,2,3\},\{2,3,4\},\{3,4,6\},\{4,5,6\}$

Claim. There is an integer B such that (U,C) contains an exact cover if, and only if the objective function for an optimal solution of $(G,\mathcal{V})$ is smaller than B.

We call an arc-flag assignment an *exact cover* iff for each $u \in U$, the value $\mathcal{F}_{(a,c^-)}(V_2)$ is true for exactly one $c \in C$ with $u \in c$. Obviously an exact cover on G induces one on (U,C) and vice versa. Let $\mathcal{F}^*$ be an exact cover, then

$$\sum_{t\in U} V_{\mathcal{F}^*}(a,t) = \beta := \underbrace{|U|}_{\#\text{ targets}} \Big(\underbrace{1}_{a} + \underbrace{2q}_{\text{nodes in } C^-\cup C^+}\Big) + \underbrace{|U|(|U|+1)/2}_{\text{overall sum nodes in } U}$$

holds. The term $|U|(|U|+1)/2$ derives from the fact that the nodes in U are settled in some fixed order and before settling u all nodes v with $v < u$ get settled. We set $B := \alpha + \beta$. Further, as

$$\sum_{t\in U} V_{\mathcal{F}}(a,t) = \underbrace{|U|\cdot 1}_{a} + 2|U|\#\{(a,c^-)|\ \mathcal{F}_{(a,c^-)}(V_2) = true\} + \underbrace{|U|(|U|+1)/2}_{\text{nodes in } X}$$

for arbitrary $\mathcal{F}$ from $\sum_{t\in U} V_{\mathcal{F}}(a,t) \le \beta$ follows that $\mathcal{F}$ is an exact cover.

External Shortcuts for Arc-Flags. This problem models an enhancement of the arc-flag technique that is used within the SHARC-algorithm. We are in the situation that the graph $G = (V,E,len)$, a h-partition $\mathcal{V} = (V_1,\dots V_h)$ of G and an integer k are already given. A shortcut is an edge (u,v) that is added to the graph for which $len(u,v) = \text{dist}(u,v)$. A shortest path $v_1, v_2, \dots, v_\ell$ is called *canonical* if it is edge-minimal among all shortest v_1-v_ℓ-paths and if $(v_1,\dots,v_\ell)$ is lexicographically minimal among all edge-minimal shortest v_1-v_ℓ-paths.

The query is similar to the arc-flag query but the preprocessing stage differs. We are allowed to add k shortcuts to the graph, afterwards the vectors $\mathcal{F}_{(u,v)}$ get computed as follows: $\mathcal{F}_{(u,v)}(V_i)$ is true iff a *canonical* shortest path starting with the edge (u,v) and ending at a node $t \in V_i$ exists. W.l.o.g we do not insert shortcuts that are already present in the graph.

*Problem (*ExtShortcutsArcFlags*).* Given a graph $G = (V,E,len)$, a partition $\mathcal{V} = (V_1,\dots,V_h)$ of V and a positive integer k, insert a set $\mathcal{S}$ of k shortcuts to G, such that $\sum_{s,t\in V} V_{\mathcal{S}}(s,t)$ is minimal.

Theorem 7. *Problem* ExtShortcutsArcFlags *is NP-hard (even for directed acyclic graphs).*

Proof. We make a reduction from X3C. Given an instance (U,C) with $|U| = 3q$, we construct an instance $(G, k = q)$ of ExtShortcutsArcFlags as follows. Starting with the empty graph, we introduce a path

(h_1, h_2, a) to G. For each $u \in U$ we insert an edge (u, h_1) to G. For each $c \in C$ we insert a path (c, d_c^1, d_c^2, a) and M edges $(m_c^1, c), \ldots, (m_c^M, c)$ to G with M to be specified later. We denote by $\widetilde{M}$ the set $\{m_c^i \mid c \in C, 1 \leq i \leq M\}$ and by D_i the set $\{d_c^i \mid c \in C\}$. Finally, there is an edge (u, c) iff $u \in c$. The edge lengths are adjusted such that each path in G is a shortest path, i.e. all edge lengths are 1 except the length of edge (h_2, a) which equals 2. The partition $\mathcal{V} = (V_1, V_2)$ is given by $V_2 = \{a\}, V_1 = V \setminus V_2$. The transformation is polynomial (see Figure 7 for a visualization).

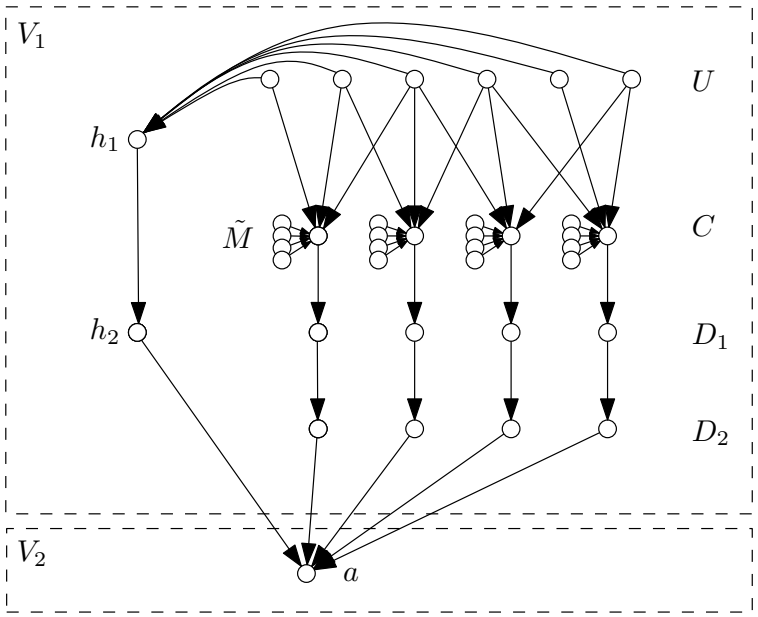


Fig. 7: Graph G constructed from the X3C-instance $\{1,2,3\}, \{2,3,4\}, \{3,4,6\}, \{4,5,6\}$

Let S be a shortcut assignment. It is

$$\sum_{s,t \in V} V_{\mathcal{S}}(s,t) = \underbrace{\sum_{s \in V, t \in V \setminus \{a\}} V_{\mathcal{S}}(s,t)}_{\alpha} + \underbrace{\sum_{s \in V \setminus \widetilde{M}} V_{\mathcal{S}}(s,a)}_{\leq \beta} + \underbrace{\sum_{s \in \widetilde{M}} V_{\mathcal{S}}(s,a)}_{\gamma}$$

The value of α is independent of S and can be computed in polynomial time: Each edge (v, w) in the graph induced by $V \setminus \{a\}$ is the only path from v to w. This does not change due to a shortcut insertion. Therefore $\mathcal{V}_{(v,w)}(V_1)$ is true iff $w \neq a$.

The value $\beta := |V \setminus \widetilde{M}|^2$ is a correct bound as the corresponding part of the search space consists of $|V \setminus \widetilde{M}|$ different queries that settle at most the entire subgraph reachable from the source. Note that β is independent of M. We now fix $M := \max\{\beta + 1, 4\}$.

Claim. Let $\mathcal{S}$ be an optimal solution. Then, each shortcut in $\mathcal{S}$ is of the form (c, a) for an $c \in C$.

W.l.o.g $q < |C|$ which ensures the existence of such a shortcut-assignment. Further, $\gamma = 5M|C|$ in the original graph without shortcuts. We can bound the value of γ on a graph with shortcuts by subtracting the maximal yield for each type of shortcut:

$$\begin{aligned} \gamma \geq 5M|C| &- 1|S \cap \widetilde{M} \times D_1| \\ &- 2|S \cap \widetilde{M} \times D_2| \; - M|S \cap C \times D_2| \\ &- 3|S \cap \widetilde{M} \times \{a\}| - 2M|S \cap C \times \{a\}| - M|S \cap D_1 \times \{a\}| \end{aligned}$$

Let $\mathcal{S}^*$ be such that each shortcut is the form (c, a) for a $c \in C$. For each shortcut (c, a), the flag $\mathcal{V}_{(c,d_c^1)}(V_2)$ is false. Hence $\gamma = 5M|C| - 2Mq := \gamma_{opt}$ which is optimal for γ. If, for a shortcut assignment $\mathcal{S}'$, at least one shortcut is not of this form we have $\gamma \geq \gamma_{opt} + M$. It holds $\sum_{s,t} V_{\mathcal{S}^*}(s,t) \leq \alpha + \beta + \gamma_{opt} < \alpha + \gamma_{opt} + M \leq \sum_{s,t} V_{\mathcal{S}'}(s,t)$ which implies that $\mathcal{S}^*$ is better than $\mathcal{S}'$.

Claim. There is an integer B, such that an optimal solution of (G, q) is smaller than B if, and only if (U, C) contains an exact cover.

Let $\mathcal{S}^*$ be optimal. We write

$$\sum_{s,t \in V} V_{\mathcal{S}^*}(s,t) = \underbrace{\sum_{s \in V, t \in V \setminus \{a\}} V_{\mathcal{S}^*}(s,t)}_{\alpha} + \underbrace{\sum_{s \in \substack{D_1 \cup D_2 \\ \cup \{h_1, h_2\}}} V_{\mathcal{S}^*}(s,a)}_{\alpha'} + \sum_{s \in U} V_{\mathcal{S}^*}(s,a) + \underbrace{\sum_{s \in C} V_{\mathcal{S}^*}(s,a)}_{=4(|C|-q)+2q} + \underbrace{\sum_{s \in \widetilde{M}} V_{\mathcal{S}^*}(s,a)}_{=5M|C|-2qM}$$

The value of α' is equal for all optimal $\mathcal{S}$ and computable in polynomial time. We call a shortcut assignment $\mathcal{S}$ *set covering* iff for each $u \in U$, there is a $c \in C$ with $u \in C$ such that $(c, a) \in \mathcal{S}$.

It is $\sum_{s \in U} V_S(s, a) = 3|U|$ if an optimal shortcut assignment $\mathcal{S}$ is set covering and greater otherwise: Let u be in U. If S is setcovering, a canonical shortest u-a path in $(V, E \cup \mathcal{S})$ is of the form s-c-a for a $c \in C$. Therefore, $\mathcal{V}_{(u,v)}(V_2)$ is true, only for one node v which must be in C and for which there is a shortcut (v, a). Because of (v, a) the flag $\mathcal{V}_{(v,d_v^1)}$ is false which implies $V_{\mathcal{S}}(u, a) = 3$ which is minimal for every u and optimal $\mathcal{S}$. If u is not covered by a shortcut, the canonical shortest u-a-path is u-h_1-h_2-a and $V_{\mathcal{S}}(u, a) = 4$.

We set $B := \alpha + \alpha' + 3|U| + 4(|C| - q) + 2q + 5|M||C| - 2qM$. If (G, q) has optimal solution $\mathcal{S}$ with objective at most B, we know that $\mathcal{S}$ is set covering and that $\{c \mid (c, a) \in \mathcal{S}\}$ is a set-cover for (X, C). The other way around, an solution $\mathcal{S}'$ for (G, q) with objective value at most B can be constructed out of a set-cover C' for (X, C) by setting $\mathcal{S}' := \{(c', a) \mid c' \in C'\}$.

7 Contraction Hierarchies

Throughout this section we work on undirected graphs. This is no restriction as the results also hold for directed graphs with edges always being symmetric. Given the input graph $G = (V, E)$, the preprocessing of Contraction Hierarchies (CH) consists of distuingishing a total order $\prec$ on V and iteratively *contracting* the $\prec$-least node until G is empty. A node v is contracted as follows: For each pair of edges $\{u, v\}$, $\{v, w\}$ such that (u, v, w) is the only shortest u-w-path, a new edge (u, w) called a *shortcut* is introduced with length $len\{u, v\} + len\{v, w\}$ to G. Afterwards, v and all of its adjacent edges are removed from G. The output of the preprocessing is $\prec$ and the graph $H_\prec(G) = (V, E \cup E')$ where E' is the set of all edges that got inserted due to node contraction. We call $H_\prec := H_\prec(G)$ the *contraction hierarchy* of G and denote by $|H_\prec|$ the number of edges $|E'|$.

The CH-query is bidirectional Dijkstra's algorithm on $H_\prec(G)$ applying two changes. Firstly, no special stopping criterion exists, the whole reachable subgraph gets settled. Secondly, when settling node u, only edges $\{u, v\}$ with $u \prec v$ get relaxed.

*Problem (*CH Preprocessing*).* Given a graph $G = (V, E)$, a length function $len\colon E \to \mathbb{R}^+$ and a number $K \in \mathbb{Z}_{\geq 0}$, find an order $\prec$ on V, such that $|H_\prec(G)| \leq K$ and $\sum_{s,t \in V} V_\prec(s, t)$ is minimal?

Theorem 8. *Problem* CH Preprocessing *is* NP*-hard.*

Proof. For this problem, it is not assured that a feasible solution exists. We show that already the problem of minimizing $|H_\prec|$ is NP-complete. To that end we make a reduction from VertexCover:

*Problem (*VertexCover*).* Given an undirected graph $G = (V, E)$ and a positive integer $K \leq |V|$, is there a vertex cover of size K or less, i.e., a subset $C' \subseteq V$ of G with $|C'| \leq K$ such that for each edge $\{u, v\} \in E$ at least one of u and v belongs to C'.

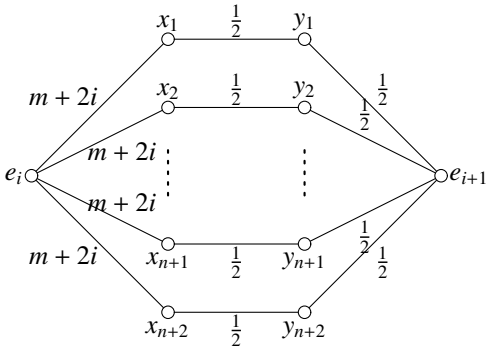
Given a Vertex Cover instance $(G = (V, E), K)$, we construct a weighted graph $G' = (V', E')$, which admits a contraction hierarchy H with at most $|E'| + K$ arcs, if and only if G has a vertex cover of size at most K. From now on let $m = |E|$ and $n = |V|$ and w.l.o.g we assume that each vertex is adjacent to at least one edge.

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

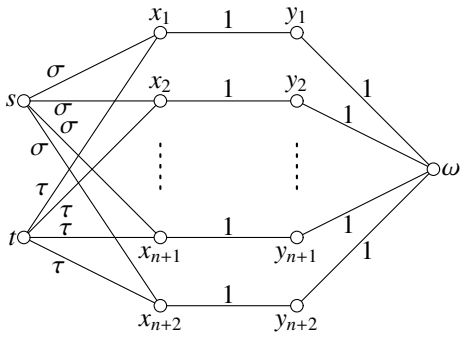
Now we fix an arbitrary order $e_1, \ldots, e_m$ on E and connect each e_i to e_{i+1} by a *honeycomb gadget* H_i that enforces the contraction order $e_i \prec e_{i+1}$. The gadget H_i can be seen in Figure 8a. Additionally we have a *final gadget* F connecting s and t, which is depicted in Figure 8b. Finally we have to fix the edge-lengths in G'. We let $len(t, v) = \frac{1}{2}m$, $len(e_i, v) = 2m$ and $len(s, e_i) = m + i$ for $e_i \in E$ and $v \in V$. The edge-lengths in the gadgets are chosen according to Figure 8a and Figure 8b. The whole construction is summarized in Figure 9. Note that G' can be computed in polynomial time, as K is polynomial in $|V|$.

Direction "only if". There is a vertex cover $C \subseteq V$ in G of at most K nodes only if G' admits a contraction hierarchy H with at most $K + |E'|$ arcs: Let $C \subseteq V$ be a vertex cover with at most K vertices. Consider the following contraction order of V':

1. Contract all $v \in V \setminus C$. This does not insert any shortcuts into the hierarchy: Paths of the form (t, v, e) are no unique shortest paths as there must be a path (t, c, e) of same length with $c \in C$. Paths of the form (e_i, v, e_j) have length $4m$ and are no shortest paths as the path (e_i, s, e_j) has length $2m + i + j < 4m$.
2. Contract all edge-vertices $e \in E$ in the chosen order $e_1, \ldots, e_m$. Note that by contraction of e_i the contraction of the gadget connecting e_i to its successor e_{i+1} is implicitly included. This step inserts at most K shortcuts into the hierarchy. We use the notation from Figure 10a.
   (a) The path $p = (x_r, e_i, x_s)$ has length $2m+4i$ and thus is no shortest path, as the path $(x_r, y_r, e_{i+1}, y_s, x_s)$ has length 2.



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



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

Fig. 8: The gadgets used in the reduction from Vertex Cover to CH Preprocessing

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

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

(d) A shortcut (s, v) may be introduced replacing the path $p = (s, e_i, v)$. At this step at most $|C| = K$ vertices are left in V. Hence at most K such shortcuts are introduced.

After contraction of e_i the remaining part of the gadget H_i consists only of the vertex e_{i+1} and simple paths (x_r, y_r, e_{i+1}). Thus it can be contracted without introducing any shortcuts.

To exactly know the structure of G' after this step we additionally proof that there *is* a shortcut (s, v) for each $v \in V$: Let e_i be the first edge-vertex adjacent to v in our fixed order $e_1, \ldots, e_m$, then $p = (v, e_1, s)$ is a unique shortest path of length $3m + 1$ because of the following case distinction.

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

(b) $p' = (s, e_j, u, t, v)$ for some edge-vertex $e_j \neq e_i$ and some vertex $u \in V$. Then p' has length $4m + j$, which is greater than $3m + 1$.

(c) $p' = (s, x_r, t, v)$ for some vertex x_r in the final gadget F. Then p' has length at least $10m$, which is greater than $3m + 1$, too.

3. Contract the special vertex s. This does not insert any shortcut in the hierarchy: The remaining graph consists of $\{s, t\} \cup C$, the final gadget F and the set of shortcuts that were inserted $\{\{s, v\} : v \in C\}$, where the edge $\{s, v\}$ has weight $3m + i$, if e_i is the first edge-vertex in the order $e_1, \ldots, e_m$ that is adjacent to v. Hence, the graph like the one shown in Figure 10b, from which we borrow notation for the following considerations. We have to take the following paths into account:
   (a) The path $p = (v, s, v')$ between two vertices $v, v' \in C$ has length greater than $6m$. As the path (v, t, v') has length m the path p is clearly no shortest path.
   (b) The path $p = (x_r, s, x_s)$ between two vertices x_r and x_s of the final gadget F has length $10m + \frac{1}{4}$, while the path (x_r, t, x_s) has length $10m$. Therefore p is no shortest path.
   (c) The path $p = (x_r, s, v)$ has length $8m + i + \frac{1}{8}$. Again, p is no shortest path as the path (x_r, t, v) has length $5m + \frac{1}{2}m$.
4. Contract all $v \in C$. This does not insert any shortcut into the hierarchy as after Step 3 all $v \in C$ have degree one.
5. After Step 4 the remaining graph consists only of the final gadget F without s and its incident edges. For each two distinct vertices x_r, x_s in the final gadget F the path (x_r, t, x_s) has length $10m$. Hence it is no shortest path as $(x_r, y_r, \omega, y_s, x_s)$ has length 4. Thus t can be contracted without introducing any additional shortcuts. After contraction of t the remaining part of F is the vertex ω with paths (x_r, y_r, ω) attached to it. This, too, can be contracted without inserting any new shortcuts into the hierarchy.

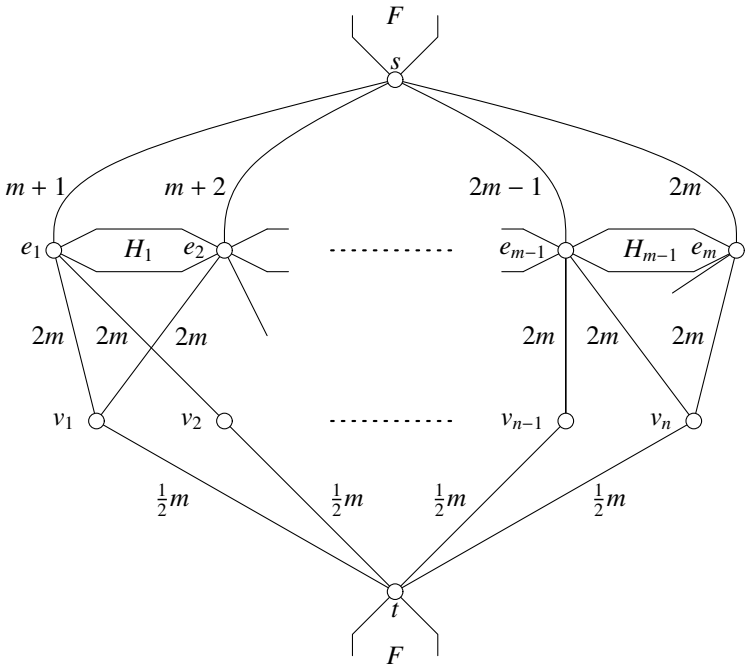


Fig. 9: Schematic picture of G'. The honeycomb gadgets H_i are depicted by small hexagons between e_i and e_{i+1}. For readability reasons the final gadget F is only shown as half hexagons at s and t.

Direction "if". On the other hand suppose there is an order $\prec$ on the vertices of G', such that the corresponding contraction hierarchy has at most $|E'|+K$ arcs – or equivalently at most K shortcuts. We will first show some simpler properties that the contraction order $\prec$ must possess and then construct a vertex cover in G using these properties and the contraction order.

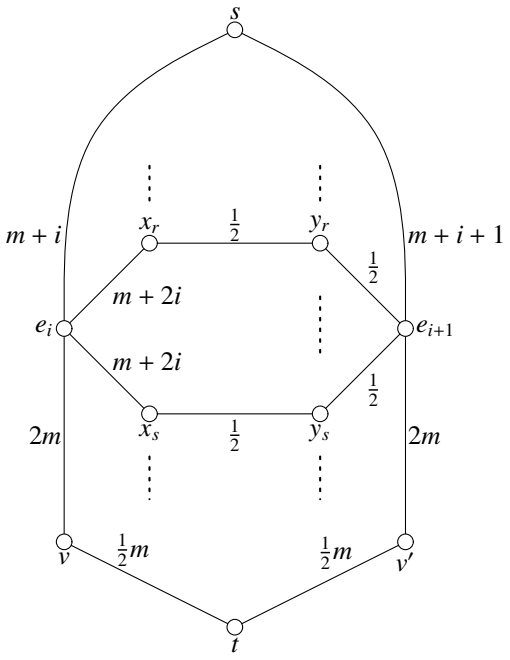
Claim. Each edge-vertex e_i gets contracted before its successor e_{i+1} in the fixed order $e_1, \ldots, e_m$.

Assume the contrary and consider the honeycomb gadget H_i between e_i and e_{i+1}. Without loss of generality let $(x_1, y_1), \ldots, (x_L, y_L)$ be the pairs of vertices (x_r, y_r) in H_i, such that $e_{i+1} \prec x_r$ or $e_{i+1} \prec y_r$. Then there are $K + 2 - L$ pairs $(x_{L+1}, y_{L+1}), \ldots, (x_{K+2}, y_{K+2})$, where $x_r, y_r \prec e_{i+1} \prec e_i$ for $r > L$ which we consider first:

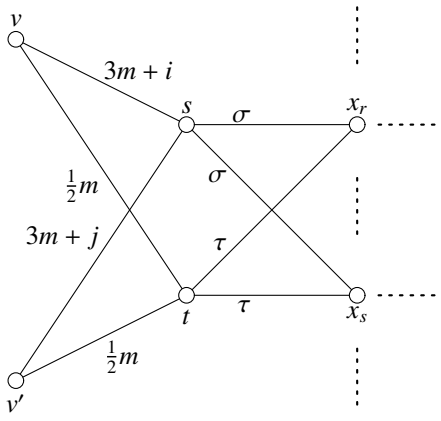
1. $y_r \prec x_r, e_i, e_{i+1}$
   The path $p = (x_r, y_r, e_{i+1})$ is a unique shortest path of length 1:
   (a) The paths $(x_r, e_i, x_s, y_s, e_{i+1})$, where $s \neq r$, have length $2m + 4i + 1$.
   (b) The paths (x_r, e_i, v, e_{i+1}), where v is some vertex $v \in V$ incident to e have length $5m + 2i$.
   (c) The path (x_r, e_i, s, e_{i+1}) has length $3m + 4i + 1$.
2. $x_r \prec y_r, e_i, e_{i+1}$
   The path $p = (e_i, x_r, y_r)$ is a unique shortest path of length $m + 2i + \frac{1}{2}$:
   (a) The paths $(e_i, x_s, y_s, e_{i+1}, y_r)$, where $s \neq r$, have length $m + 2i + \frac{3}{2}$.
   (b) The path (e_i, s, e_{i+1}, y_r) has length $2m + 2i + \frac{3}{2}$.
   (c) The path (e_i, v, e_{i+1}, y_r) has length $4m + \frac{1}{2}$.

Therefore contraction of x_r and y_r before e_i and e_{i+1} results in at least one additional edge being inserted into the hierarchy. This sums up to at least $K + 2 - L$ additional edges.

Now consider the pairs $(x_1, y_1), \ldots, (x_L, y_L)$, where at least one of x_r, y_r gets contracted after e_{i+1}. For $1 \leq s \leq L$ let z_s be the vertex $z_s \in \{x_s, y_s\}$ that is a neighbour of e_{i+1} when e_{i+1} gets contracted. For



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



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

Fig. 10: Important steps during contraction of G' given a vertex cover $C \subseteq V$

distinct z_r, z_s the path $p = (z_s, e_{i+1}, z_r)$ has length at most 2, while paths $(z_r, \ldots, e_i, \ldots, z_s)$ have length at least $m + 2i$, as they include the edge $\{x_r, e_i\}$ of length $m + 2i$ or the shortcut $\{y_r, e_i\}$ of length $m + 2i + \frac{1}{2}$. Hence p is a unique shortest path and contraction of e_{i+1} before z_s, z_r and e_i inserts an additional shortcut $\{z_r, z_s\}$. As there are $\frac{1}{2}L(L-1)$ such pairs $\{z_r, z_s\}$, contraction of e_{i+1} leads to the insertion of $\frac{1}{2}L(L-1)$ shortcuts.

Altogether, contraction of e_{i+1} before e_i results in at least $K + 2 - L + \frac{1}{2}L(L-1) > K$ shortcuts contradicting the assumption of at mosthttp://portal.acm.org/citation.cfm?id=502098 K inserted edges.

Claim. The special vertices s and t gets contracted before the vertex ω in the final gadget F.

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

1. $y_r \prec x_r, \omega, \alpha$. In this situation (x_r, y_r, ω) is a unique shortest path.
2. $x_r \prec y_r, \omega, \alpha$. In this situation (α, x_r, y_r) is a unique shortest path.

Contraction of x_r and y_r before α and ω inserts in any case at least one shortcut which sums up to at least $K + 2 - L$ additional edges in the hierarchy.

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

Altogether, contraction of ω before α results in at least $K+2-L+\frac{1}{2}L(L-1) > K$ additional shortcuts being inserted. This is a contradiction and thus $\alpha \prec \omega$.

Claim. The special vertex t gets contracted after all $v \in V$.

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

1. $y_r \prec x_r, t, \omega$. In this situation (x_r, y_r, ω) is a unique shortest path of length 2.
2. $x_r \prec y_r, t, \omega$. In this situation (t, x_r, y_r) is a unique shortest path of length $5m+1$.

Contraction of x_r and y_r before t hence inserts at least one additional edge into the hierarchy which sums up to at least $K+2-L$ additional shortcuts. For $1 \leq r \leq L$ now let z_r be the vertex $z_r \in \{x_r, y_r\}$ that is adjacent to t, when t gets contracted. We have $\text{dist}(v_0, s) \geq 3m+1)$, $\text{dist}(s, x_r) \geq 5m+\frac{1}{8}$ and $\text{dist}(s, y_r) \geq 5m+\frac{9}{8}$.

1. Let $z_r = x_r$ then $p = (v_0, t, x_r)$ is a unique shortest path of length $\frac{11}{2}m$.
2. Let $z_r = y_r$ then $p = (v_0, t, x_r, y_r)$ is a unique shortest path of length $\frac{11}{2}m+1$.

Contraction of t therefore results in the insertion of an additional shortcut $\{v_0, z_r\}$. As there are L such neighbours z_r of t, contraction of t inserts at least L additional edges. Altogether contraction of v after t results in $K+2-L+L > K$ additional shortcuts, which is a contradiction.

Claim. All edge-vertices $e_i \in E$ get contracted before s.

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

1. $y_r \prec x_r, s, \omega$. In this situation (x_r, y_r, ω) is a unique shortest path of length 2.
2. $x_r \prec y_r, s, \omega$. In this situation (s, x_r, y_r) is a unique shortest path of length $5m+\frac{1}{8}+1$.

Contraction of x_r and y_r before s hence inserts at least one additional edge into the hierarchy which sums up to at least $K+2-L$ additional edges into the hierarchy.

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

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

Subsumption. The following observations subsume the above claims about possible pairwise contraction orders.

1. E gets contracted in order $e_1, \ldots, e_m$.
2. $V \prec t$ and $E \prec s$, that is whenever we encounter vertices $v \in V$ or edge-vertices $e \in E$, we may assume that the vertex t or the vertex s respectively are not contracted yet.

In the final step of this proof we will construct a vertex cover for the original graph G and prove that it contains at most K vertices.

For each vertex $v \in V$ let $e_{\min}(v)$ be the first edge-vertex in order $e_1, \ldots, e_m$ that is incident to v, that is $e_{\min}(v) = e_M$, where $M = \min\{i : e_i \text{ is incident to } v\}$. We partition E into two sets. The set E_1 contains those edges that are incident to some vertex v that gets contracted after $e_{\min}(v)$.

$$E_1 = \{e = \{u, v\} \in E |\ e_{\min}(u) \prec u \text{ or } e_{\min}(v) \prec v\}$$

Secondly we let E_2 be the set of edges that are incident to two vertices u and v that get both contracted before $e_{\min}(v)$.

$$E_2 = \{e = \{u, v\} \in E |\ u \prec e_{\min}(u) \text{ and } v \prec e_{\min}(v)\}$$

Obviously it is $E = E_1 \dot\cup E_2$. Now we define for each edge $e \in E$ the *cover vertex* $v(e)$ of e as follows:

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

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

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

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

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

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

First consider a vertex $v = v(e) \in C$ with $e \in E_1$. Then, by definition of E_1, $v \succ e_i = e_{\min}(v)$. By the subsumption we now that s and t are still present in the graph when e_i gets contracted. Now consider possible paths between s and v at this step.

1. The path $p = (s, e_i, v)$ has length $3m + i$. For any other e_j the path (s, e_j, v) has length $3m + j$ and since $e_i = e_{\min}(v)$ the path p is a unique shortest path among the paths (s, e_j, v).
2. For some vertex $u \neq v$ and some edge-vertex $e_j \neq e_i$ the path (s, e_j, u, t, v) has length $4m + j$
3. The path (s, x_r, t, v), where x_r is a vertex of the final gadget F, has length $10m + \frac{5}{8}$.

Hence, (s, e_i, v) is a unique shortest path when e_i gets contracted and thus contraction of e_i inserts a shortcut $\{s, v\}$. We let $M(v) = \{s, v\}$.

Next we account for the vertices in $v(E_2)$. Let $v \in v(E_2)$. Choose arbitrary $e = \{u, v\}$ in E_2 such that $v = v(e)$. By definition of E_2, we have $u \prec e_{\min}(u)$ and $v \prec e_{\min}(v)$. In particular $u \prec v \prec e$. By the subsumption, the vertices s and t are not contracted, when u or v get contracted. Consider the paths $p_u = (t, u, e)$ and $p_v = (t, v, e)$, each of length $\frac{5}{2}m$. Apart from p_u and p_v the only relevant path between t and e is $p' = (t, x_r, s, e)$, where x_r is some vertex of the final gadget F. The length of p' is greater than $10m$ and thus p_u and p_v are shortest paths.

When v gets contracted, u and the path p_u are already contracted and p_v is a unique shortest path. Contraction of v hence inserts a shortcut $\{t, e\}$ into the hierarchy. We let $M(v) = \{t, e\}$. Observe that M is injective on $v(E_2)$, as $M(x) = M(y)$ implies $x = y = v(e')$ for some e'. This finishes the proof.

We now consider the problem of minimizing the preprocessing size without regarding the search-space size.

*Problem (*CHPSO*).* Given a graph $G = (V, E)$, a length function $len\colon E \to \mathbb{R}^+$ and a number $K \in \mathbb{Z}_{\geq 0}$, find an order $\prec$ on V, such that $|H_\prec(G)| - |E|$ is minimal.

Using essentially the same reduction as for the last problem, we can give some non-approximability results for CHPSO.

Corollary 1. *Problem* CHPSO *is APX-hard and, for any $\epsilon > 0$, it is NP-hard to approximate* CHPSO *within a ratio of $7/6 - \epsilon$.*

Proof. The problem MinVertexCover is to find a vertex cover of minimal cardinality. The APX-hardness of MinVertexCover has been shown in [18]. In [14] it is shown that it is NP-hard to approximate MinVertexCover within a ratio of $7/6 - \epsilon$.

Let G be a MinVertexCover-Instance and G' be a CHPSO instance constructed like in the proof of Theorem 8. Let $\mathrm{opt}(G)$ and $\mathrm{opt}(G')$ denote the values of optimal solutions of G and G', respectively. According to the proof of Theorem 8 we know that $\mathrm{opt}(G) = \mathrm{opt}(G')$.

Further, let $\prec$ be a feasible solution for G' such that $(|H_\prec(G')| - |E|)/\mathrm{opt}(G') \leq 7/6 - \epsilon$ for an $\epsilon > 0$. Let $C = \{v(e) : e \in E\}$ be like in the proof of Theorem 8. Then C is a feasible solution for G and $C \leq (|H_\prec(G')| - |E|$. Hence, $|C|/\mathrm{opt}(G) \leq 7/6 - \epsilon$.

The proof of APX-hardness works analogous.

Lower Bounds for Search-Space Guarantees. We now consider the following question: Preprocessing size and space are unrestricted, what guarantee can Contraction Hierarchies give for the size of the average search space? Obviously the size of the search space is bounded from above by the number n of nodes. For arbitrary graphs, no bound outside $\Omega(n)$ is possible as the complete graph with n vertices has average search space size $(n+1)/2$.

We give a lower bound of $\Omega(\mathrm{ld}\, n)$ for guarantees on the average search-space size on sparse graphs (especially for graphs with bounded degree). Throughout the remainder of this section, we consider the input-graph $P_n = (V_n, E_n)$ to be a path, i.e. $V_n = \{1, \ldots, n\}$ and $E_n = \{\{i, i+1\} \mid 1 \leq i < n\}$. Given a graph $G = (V, E)$, an order $\prec$ on V and the contraction hierarchy $(V, E^*) := H_\prec(G)$, the *directed contraction hierarchy* $\overrightarrow{H}(G, \prec)$ is defined as

$$\overrightarrow{H}(G, \prec) := (V, \{(u, v) \mid \{u, v\} \in E^*, u \prec v\}).$$

For simplicity, we consider only one direction of the actual query. Remember that $\mathrm{dist}_G(u, v)$ denotes the distance from vertex u to vertex v in graph G. Consequently,

$$\mathcal{V}_{G,\prec}(u) := \left|\{v \in V_n \mid \mathrm{dist}_{\overrightarrow{H}(G,\prec)}(u, v) < \infty\}\right|$$

is the number of nodes visited during a query starting at vertex u and

$$\mathcal{V}(G, \prec) := \sum_{u \in V} \mathcal{V}_{G,\prec}(u)$$

is $n/2$ times the average search space for the contraction hierarchy $H_\prec(P_n)$. For all definitions, we will leave out the order $\prec$ whenever the choice of $\prec$ is clear.

Lemma 5. *For all $n \in \mathbb{Z}^+$ and all orders $\prec$ on V_n, it is*

$$\mathcal{V}(P_n, \prec) \geq B(n+1)$$

where $B(k) = \sum_{i=1}^{k} \lceil \mathrm{ld}\, i \rceil$ and $P_n = (V_n, E_n)$ with $V_n = \{1, \ldots, n\}$ and $E_n = \{\{i, i+1\} \mid 1 \leq i < n\}$.

In order to proof this result we first give two facts on the sequence $B(k)$. The *sorting numbers* $B(k)$ are sequence A001855 in [24]. The following recursive formula is a key to the proof of Lemma 5.

Lemma 6. *Let $B(k) = \sum_{i=1}^{k} \lceil \mathrm{ld}\, i \rceil$ for $k \in \mathbb{Z}^+$. Then*

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

Proof. We do induction on n. For $n = 2$ we have

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

For $n > 2$, we get by induction hypothesis

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

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

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

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

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

This finishes the proof.

Further, from the monotonicity of ld we have the following inequality.

Lemma 7. *Let $B(k) = \sum_{i=1}^{k} \lceil \operatorname{ld} i \rceil$ for $k \in \mathbb{Z}^+$. Further, let $n_1, n_2, n \in \mathbb{Z}^+$ with $n_1 + n_2 = n$. Then*

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

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

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

Equipped with these two lemmata on the sorting numbers we now approach the proof of Lemma 5:

Proof (of Lemma 5). We do induction on n. If $n = 1$, there is nothing to show, as $B(2) = 1$. Now let $n > 1$. Furthermore let $\prec$ be an order on V_n and v be the $\prec$-largest vertex in V_n. Removal of v splits P_n into graphs P_1, P_2 and $H = H_{\prec}(p_n)$ into graphs H_1, H_2 of n_1 and n_2 vertices, where $n_1 + n_2 = n - 1$. It is $H_1 = H_{\prec}(P_1)$ and $H_2 = H_{\prec}(P_2)$. Furthermore $\mathcal{V}_{P_i}(v) = \mathcal{V}_{P_n}(v) - 1$ for all vertices $v \in V(P_i)$ and thus

$$\mathcal{V}(P_n) = \mathcal{V}(P_1) + n_1 + \mathcal{V}(P_2) + n_2 + 1$$

By induction hypothesis we have $\mathcal{V}(P_i) \geq B(n_i + 1)$. Hence we may apply Lemmata 6 and 7 to obtain

$$\mathcal{V}(P_n) \geq B\left(\left\lceil \frac{n+1}{2} \right\rceil\right) + B\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) + n = B(n+1)$$

which was to show.

The lower bound of $B(n+1)$ is tight as Algorithm 1 computes an order $\prec$ on V_n, such that $\mathcal{V}(H_{\prec}) = B(n+1)$. The proof is by induction analogous to the proof of Lemma 5.

Corollary 2. *For all $n \in \mathbb{N}$ and all orders $\prec$ on V_n it is $\mathcal{V}(H_{\prec}) = \Omega(n \operatorname{ld} n)$ and there is an order $\prec$ on V_n, such that $\mathcal{V}(H_{\prec}) = \Theta(n \operatorname{ld} n)$.*

Algorithm 1: OPTIMALPATHORDER

Input : Path $P_n = (V_n, E_n)$ of n vertices
Output: Order $\prec$ on V_n, such that $\mathcal{V}(H_\prec) = B(n+1)$
if $n = 0$ **then**
  return $\langle\ \rangle$
else
  Pick vertex $v \in V$ separating P_n into paths Q and R of length $\lfloor \frac{n-1}{2} \rfloor$ and $\lceil \frac{n-1}{2} \rceil$
  return OPTIMALPATHORDER(Q) $\circ$ OPTIMALPATHORDER(R) $\circ \langle v \rangle$
end

Proof. The corollary is an immediate consequence of the equation

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

which can be shown by induction on n. If $n = 1$we have $B(1) = 0 = 0 - 1 + 1$. If $n > 1$, then

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

If $\lceil \operatorname{ld} n - 1 \rceil = \lceil \operatorname{ld} n \rceil$, the claimed equality follows immediately. On the other hand, $\lceil \operatorname{ld} n - 1 \rceil < \lceil \operatorname{ld} n \rceil$, if and only if $n = 2^k + 1$ for some $k \in \mathbb{Z}^+$. In that case we have

$$\begin{aligned} B(n) &= (n-1) \lceil \operatorname{ld} n - 1 \rceil - 2^{\lceil \operatorname{ld} n - 1 \rceil} + 1 + \lceil \operatorname{ld} n \rceil \\ &= (n-1) \cdot k - 2^k + 1 + k + 1 \\ &= nk - n + 3 \\ &= n(k+1) - 2(n-1) + 1 \\ &= n(k+1) - 2^{k+1} + 1 \\ &= n \lceil \operatorname{ld} n \rceil - 2^{\lceil \operatorname{ld} n \rceil} + 1 \end{aligned}$$

8 Conclusion

Speed-up techniques have been widely studied experimentally for the last years, especially for road-networks. There is large interest in a theoretical foundation of the techniques developed and some first work on the topics have been published [1, 4]. In this work we focused on the preprocessing phases of the recent techniques. These usually incorporate a degree of freedom that, in practice, is filled in a heuristical manner.1 Until now, the complexity status of filling the according degree of freedom was unknown. We settled this question by showing that all variants considered are NP-hard to optimize.

There are numerous open questions for the topic. A reasonable next step to enhance this work is the development of approximation- or fixed parameter tractable algorithms for the preprocessing phase. Efficient algorithms for special graph classes would help to show the bounds of intractability.

When working with special graph classes (either for giving algorithms or showing the complexity status) modeling the two main applications for speed-up techniques, road-networks and public transportation networks would be helpful. Until now there is no experimentally verified model for these two applications (but some first work [1, 7]). From a more theoretical point of view we have the question which of the problems can be solved efficiently on trees.

Another interesting question is the following: We assume preprocessing time and space is unbounded, how good can a speed-up technique actually get? Obviously, ALT and Arc-Flags can encode All-Pairs Shortest-Path in a way that a shortest s-t path and $\text{dist}(s, t)$ can be queried in time that is linear in the size of the shortest-path subgraph which yields optimal search-space for these techniques (but of course, the runtime of ALT would not be optimal in this case). The situation is not so clear for Highway-Node Routing, Reach, Highway-Hierarchies and Contraction Hierarchies. We have shown that the average CH-search space of an s-t-query can not guarantee to be better than $\Omega(n)$ for arbitrary graphs and $\Omega(\log n)$ for graphs with bounded degree. Is the second bound tight?

Finally, another interesting model can be obtained by also including the number of relaxed edges in the search-space. However, we expect to obtain the same results for that model with slightly modified proofs.

References

1. I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 782–793, 2010.
2. H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes. In Transit to Constant Shortest-Path Queries in Road Networks. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, pages 46–59. SIAM, 2007.
3. R. Bauer, T. Columbus, B. Katz, M. Krug, and D. Wagner. Preprocessing Speed-Up Techniques is Hard. In *Proceedings of the 7th Conference on Algorithms and Complexity (CIAC'10)*, Lecture Notes in Computer Science. Springer, 2010.
4. R. Bauer, G. D'Angelo, D. Delling, and D. Wagner. The Shortcut Problem – Complexity and Approximation. In *Proceedings of the 35th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'09)*, volume 5404 of *Lecture Notes in Computer Science*, pages 105–116. Springer, January 2009.
5. R. Bauer and D. Delling. SHARC: Fast and Robust Unidirectional Routing. *ACM Journal of Experimental Algorithmics*, 14:2.4, May 2009. Special Section on Selected Papers from ALENEX 2008.
6. D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering Route Planning Algorithms. In J. Lerner, D. Wagner, and K. A. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer, 2009.
7. D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. ACM Press, 2008.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness*. W. H. Freeman and Company, 1979.
9. R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In C. C. McGeoch, editor, *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, June 2008.
10. A. V. Goldberg and C. Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 156–165, 2005.
11. A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*, pages 129–143. SIAM, 2006.
12. A. V. Goldberg, H. Kaplan, and R. F. Werneck. Better Landmarks Within Reach. In C. Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 38–51. Springer, June 2007.
13. R. J. Gutman. Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*, pages 100–111. SIAM, 2004.
14. J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, July 2001.
15. M. Hilger, E. Köhler, R. H. Möhring, and H. Schilling. Fast Point-to-Point Shortest Path Computations with Arc-Flags. In C. Demetrescu, A. V. Goldberg, and D. S. Johnson, editors, *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 of *DIMACS Book*, pages 41–72. American Mathematical Society, 2009.
16. M. Holzer, F. Schulz, and D. Wagner. Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. In *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*, pages 156–170. SIAM, 2006.
17. U. Lauther. An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background. In *Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung*, volume 22, pages 219–230. IfGI prints, 2004.
18. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC'88)*, pages 229–234. ACM Press, 1988.
19. P. Sanders and D. Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA'05)*, volume 3669 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2005.
20. P. Sanders and D. Schultes. Engineering Highway Hierarchies. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA'06)*, volume 4168 of *Lecture Notes in Computer Science*, pages 804–816. Springer, 2006.

21. P. Sanders and D. Schultes. Robust, Almost Constant Time Shortest-Path Queries in Road Networks. In C. Demetrescu, A. V. Goldberg, and D. S. Johnson, editors, *9th DIMACS Implementation Challenge - Shortest Paths*, November 2006.
22. D. Schultes and P. Sanders. Dynamic Highway-Node Routing. In C. Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 66–79. Springer, June 2007.
23. F. Schulz, D. Wagner, and K. Weihe. Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. In *Proceedings of the 3rd International Workshop on Algorithm Engineering (WAE'99)*, volume 1668 of *Lecture Notes in Computer Science*, pages 110–123. Springer, 1999.
24. N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, 2008.
25. D. Wagner and T. Willhalm. Geometric Speed-Up Techniques for Finding Shortest Paths in Large Sparse Graphs. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 776–787. Springer, 2003.
26. D. Wagner, T. Willhalm, and C. Zaroliagis. Geometric Containers for Efficient Shortest-Path Computation. *ACM Journal of Experimental Algorithmics*, 10:1.3, 2005.

A Proofs of Correctness

Highway-Node Routing

Lemma 8. *Given a multilevel overlay graph $\mathcal{G} = (V, E \cup E_1 \cup \ldots \cup E_L)$ of a graph $G = (V, E)$, then bidirectional, distance-balanced* DIJKSTRA*'s algorithm in $\mathcal{G}$, where from a node of level i only edges in $E_i \cup \ldots \cup E_L$ are relaxed, returns the correct distance in G.*

Proof. Let $s, t \in V$. The construction of the multilevel overlay graph $\mathcal{G}$ obviously does not change $\text{dist}(s, t)$. Hence, it suffices to show, that for each shortest s-t-path in G, there exists a shortest s-t-path in $\mathcal{G}$, whose edges fulfill the relaxation criterion.

Let $p = (s = v_1, \ldots, v_m = t)$ be a shortest path between s and t in G. Then we can choose maximal subsequences $p_1 = (s = v_{i_1}, \ldots, v_{i_k})$ and $p_2 = (v_{j_l}, \ldots, v_{j_1} = t)$ of p, such that the levels of v_{i_r} and v_{j_r} are less or equal than the level of $v_{i_{r+1}}$ and $v_{j_{r+1}}$ respectively. Note that p_1 and p_2 are maximal and therefore have non-empty intersection, i.e. $j_l < i_k$. By definition of multilevel overlay graphs, $E \cup E_1 \cup \ldots \cup E_L$ contains edges $(v_{i_r}, v_{i_{r+1}})$ and $(v_{j_r}, v_{j_{r+1}})$ of length $len(v_{i_r}, v_{i_r+1}, \ldots, v_{i_{r+1}})$ and $len(v_{j_{r+1}}, v_{j_{r+1}+1}, \ldots, v_{j_r})$ respectively. Hence the sequences p_1 and p_2 are shortest paths in $\mathcal{G}$ and the edges $(v_{i_r}, v_{i_{r+1}})$ and $(v_{j_{r+1}}, v_{j_r})$ fulfill the relaxation condition, i.e. if v_{i_r} or $v_{j_{r+1}}$ is on level K, then $v_{i_{r+1}}$ or v_{j_r} are at least on level K, too.

Reach To see that this query algorithm is correct consider a shortest path $(s = v_1, \ldots, v_i, \ldots v_l = t)$ between s and t. If there is some vertex v_i such that for all $1 \leq j < i$ the vertices v_j get settled in the forward search and for all $i < k \leq l$ the vertices v_k get settled in the backward search, then $d^+(v_i) = len(v_1, \ldots, v_i), d^-(v_i) = len(v_i, \ldots, v_l)$ and the query is correct. Assume the contrary, i.e. that there are vertices v_j and v_k with j minimal, k maximal, $j < k$ and such that v_j does not get settled in the forward search and v_k does not get settled in the backward search. After settling v_{j-1} the priority of v_j in the forward queue is $len(v_1, \ldots, v_j)$ and after settling v_{k+1} the priority of v_k in the backward queue is $len(v_k, \ldots, v_l)$. As both v_j and v_k do not get settled, the definition of $\mathcal{R}$ and the settling criterion imply

$$\min\{len(v_1, \ldots, v_j), len(v_j, \ldots, v_l)\} \leq \mathcal{R}(v_j) < len(v_1, \ldots, v_j)$$
$$\min\{len(v_1, \ldots, v_k), len(v_k, \ldots, v_l)\} \leq \mathcal{R}(v_k) < len(v_k, \ldots, v_l)$$

Hence $len(v_j, \ldots, v_l) < len(v_1, \ldots, v_j)$ and $len(v_1, \ldots, v_k) < len(v_k, \ldots, v_l)$, which is a contradiction, as

$$len(v_1, \ldots, v_l) < len(v_1, \ldots, v_k) + len(v_j, \ldots, v_l) < len(v_k, \ldots, v_l) + len(v_1, \ldots, v_j) < len(v_1, \ldots, v_l)$$