Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 16, no. 2, pp. 447–481 (2012)
DOI: 10.7155/jgaa.00270

The Shortcut Problem – Complexity and Algorithms

Reinhard Bauer[1] *Gianlorenzo D'Angelo*[3] *Daniel Delling*[2]
Andrea Schumm[1] *Dorothea Wagner*[1]

[1]Faculty of Informatics, Karlsruhe Institute of Technology (KIT)
[2]Microsoft Research Silicon Valley
[3]MASCOTTE project, INRIA/I3S (CNRS/UNSA)

Abstract

We study a graph-augmentation problem arising from a technique applied in recent approaches for route planning. Many such methods enhance the graph by inserting *shortcuts*, i.e., additional edges (u,v) such that the length of (u,v) is the distance from u to v. Given a weighted, directed graph G and a number $c \in \mathbb{Z}_{>0}$, the *shortcut problem* asks how to insert c shortcuts into G such that the expected number of edges that are contained in an edge-minimal shortest path from a random node s to a random node t is minimal. In this work, we study the algorithmic complexity of the problem and give approximation algorithms for a special graph class. Further, we state ILP-based exact approaches and show how to stochastically evaluate a given shortcut assignment on graphs that are too large to do so exactly.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
March 2011	November 2011	April 2012	June 2012	July 2012
		Published:		
		August 2012		

Article type:	Communicated by:
Regular paper	S. Albers

Partially supported by the DFG (project WA654/16-1).

E-mail addresses: reinhard.bauer@kit.edu (Reinhard Bauer) gianlorenzo.d_angelo@inria.fr (Gianlorenzo D'Angelo) dadellin@microsoft.com (Daniel Delling) andrea.schumm@kit.edu (Andrea Schumm) dorothea.wagner@kit (Dorothea Wagner)

1 Introduction

Background. Computing shortest paths in graphs is used in many real-world applications like route-planning. Shortest paths from a given source to a given target can be computed by DIJKSTRA'S algorithm, but the algorithm is slow on huge datasets. Therefore, it can not be directly used for applications like car navigation systems or online working route-planners that require an instant answer to a source-target query. In the last decade various preprocessing-based techniques have been developed that yield much faster query-times (see [25, 31] for an overview).

One core part of some of these approaches is the insertion of *shortcuts* [7, 9, 13, 14, 15, 20, 24, 26, 28, 29], i.e., additional edges (u,v) whose length is the distance from u to v and that represent shortest u-v-paths in the graph. The strategies of assigning the shortcuts and of exploiting them during the query differ depending on the speedup-technique. Many techniques work as follows: In a preprocessing stage, the nodes of the input graph are assigned to a *level* and shortcuts between nodes of the same level are added to the graph. Afterwards, the query stage is similar to bidirectional Dijkstra's algorithm but omits some edges by preferring shortcut edges depending on the level. However, it is still guaranteed that correct distances are computed. Until now, all existing shortcut insertion strategies are heuristics and only few theoretical worst-case or average case results are known [1, 5].

In this context, an interesting new theoretical problem arises: Given a weighted, directed graph G and a number $c \in \mathbb{Z}_{>0}$, the *shortcut problem* asks how to insert c shortcuts into G such that the expected number of edges that are contained in an edge-minimal shortest path from a random node s to a random node t is minimal.

Contribution. In this work, we formally state the SHORTCUT PROBLEM and a variant of it, the REVERSE SHORTCUT PROBLEM. While we study the algorithmic complexity of both problems, the algorithmic contribution focuses on the SHORTCUT PROBLEM. We state exact, ILP-based solution approaches. We further describe two algorithms that give approximation guarantees on graphs in which, for each pair s,t of nodes, there is at most one shortest s-t-path. It turns out that this class is highly relevant as in road networks, most shortest paths are unique and only small modifications have to be made to obtain a graph having unique shortest paths. Finally, we show how to stochastically evaluate a given shortcut assignment on graphs that are too large to do so exactly. Besides its relevance as a step towards theoretical results on speedup-techniques, we consider the problem to be interesting and beautiful on its own right.

Related Work. Parts of this work have been published in [6]. The diploma thesis [30] experimentally examines heuristic algorithms to find shortcut assignments with high quality, including local search strategies and a betweenness-based approach. Furthermore, the GREEDY-step Algorithm 3 is proposed in this thesis. To the best of our knowledge, the problem of finding shortcuts as stated in this work has never been treated before.

Speedup-techniques that incorporate the usage of shortcuts are the following. Given a graph $G = (V,E)$ the multilevel overlay graph technique [19, 20, 21, 26, 27, 29] uses some centrality measures or separation strategies to choose a set of 'important' nodes V' in the graph and inserts the shortcuts S such that the graph (V',S) is edge-minimal among all graphs (V', E') for which the distances between nodes in V' are as in (V, E). Highway hierarchies [23, 24] and Reach Based Pruning [14, 15, 16, 17] iteratively sparsificate the graph according to the 'importance' of the nodes. After each sparsification step, nodes v with small in- and out-degree are deleted. Then for each pair of edges (u, v), (v, w) a shortcut (u, w) is inserted if necessary to maintain correct distances in the graph. SHARC-Routing [7, 8, 10, 11] and Contraction Hierarchies [13] use a similar strategy.

Overview. This paper is organized as follows. Section 2 introduces basic definitions. The SHORTCUT PROBLEM and the REVERSE SHORTCUT PROBLEM are stated in Section 3. Furthermore, results concerning complexity and non-approximability of the problems are given. The remainder of the paper focuses on the SHORTCUT PROBLEM. Section 4 proposes two exact, ILP-based approaches. In Section 5, a greedy algorithm is presented that gives an approximation guarantee on graphs in which shortest paths are unique. Section 6 states an approximation algorithm that works on graphs with bounded degree in which shortest paths are unique. A probabilistic approach to evaluate a given solution of the SHORTCUT PROBLEM is introduced in Section 7. The paper is concluded by a summary and possible future work in Section 8.

2 Preliminaries

Let $A \subseteq X$ be a subset of a set X. The indicator function of A and X is the function $1_A : X \to \{0, 1\}$ defined as $1_A(x) = 1$ if $x \in A$ and $1_A(x) = 0$ otherwise.

Common Graph Theory. Throughout this work, $G = (V, E, len)$ denotes a directed, weighted graph with positive length function $len : E \to \mathbb{R}_{>0}$. Given nodes u and v, we call u a neighbor of v if there is an edge (u, v) or (v, u). We denote by $N(v)$ the set of all neighbors of v. Given a set S of nodes, the *neighborhood* of S is the set $S \cup \bigcup_{u \in S} N(u)$.

We denote by $\overleftarrow{G}$ the *reverse graph* of G, i.e. the graph $(V, \overleftarrow{E}, \overleftarrow{\text{len}})$ with $\overleftarrow{E} := \{(v, u) \mid (u, v) \in E\}$ and $\overleftarrow{\text{len}}$ being defined by $\overleftarrow{\text{len}}(u, v) := \text{len}(v, u)$ for $(v, u) \in E$.

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 $(x_i, x_{i+1}) \in E$, $i = 1, \ldots, k-1$ and $x_i \neq x_j$ for each $i \neq j$. We say P *contains* an edge (u, v) if $(u, v) = (x_i, x_{i+1})$ for some $i \in \{1, \ldots, k-1\}$ and use the abbreviation $(u, v) \in P$. The *length* $\text{len}(P)$ of P is the sum of the lengths of all edges in P, i.e. $\text{len}(P) = \sum_{i=1}^{k-1} \text{len}(x_i, x_{i+1})$. A *shortest path* from node s to node t is a path from s to t of minimum length. Given two nodes s and t the *distance* $\text{dist}(s, t)$ from s to t is the length of a shortest path from s to t and ∞ if there is no path from s to t. The *diameter* of a graph G is the largest distance in G, i.e. $\max\{\text{dist}(s, t) \mid s, t \in V\}$. The *eccentricity* $\varepsilon_G(v)$ of a node v is the maximum distance between v and any other node u of G.

A *cycle* is a finite sequence $(x_1, x_2, \ldots, x_k)$ of nodes such that $(x_i, x_{i+1}) \in E$, $i = 1, \ldots, k-1$ and $x_i = x_k$. A *(rooted) tree with root (node) s* is a directed graph $T = (V', E')$ without cycles such that for each node $t \in V'$ there is exactly one path from s to t. We call v a descendant of t in T, if the path from s to v in T contains t. Note that each node is a descendant of itself.

A *shortest-paths tree with root s* is a subgraph $T = (V', E')$ of G such that T is a tree, V' is the set of nodes reachable from s and such that for each edge $(u, v) \in E'$ we have $\text{dist}(s,u) + \text{len}(u,v) = \text{dist}(s,v)$. Note that each path in T is a shortest path. The *shortest-path subgraph with root s* is the subgraph $G_s = (V', E'')$ of G such that V' is the set of nodes reachable from s and E'' is the set of all edges with $\text{dist}(s,u) + \text{len}(u,v) = \text{dist}(s,v)$. Note that G_s contains exactly all shortest-paths in G that start with s. Further, G_s is directed acyclic in case all edge weights are strictly positive.

Specific Notation and Considered Graphs. Consider a path $P = (x_1, x_2, \ldots, x_k)$. We say P *contains* node u *before* node v if there are numbers i, j with $0 \leq i \leq j \leq k$ such that $u = x_i$ and $v = x_j$.

Given is a sequence $y_1, \ldots, y_k$ for $k \geq 2$. A y_1-y_2-…-y_k-path is a path P from y_1 to y_k such that P contains node y_i before node y_{i+1} for $i = 1, \ldots, k-1$. A shortest y_1-y_2-…-y_k-path is a y_1-y_2-…-y_k-path that is a shortest path from y_1 to y_k. Let

$$P^-(x,y) := \{s \in V \mid \exists \text{ shortest } s\text{-}y\text{-path containing } x\}$$
$$P^+(x,y) := \{t \in V \mid \exists \text{ shortest } x\text{-}t\text{-path containing } y\}$$

denote the sets of start- or end-vertices of shortest paths through x and y. Similarly, let

$$P(x,y) := \{(s,t) \in V \times V \mid \exists \text{ shortest } s\text{-}t\text{-path that contains } x \text{ before } y\}$$

consist of all pairs of nodes, for which a connecting shortest path containing first x and then y exists. Finally, let

$$P^{\bowtie}(x,y) := \{u \in V \mid \exists \text{ shortest } x\text{-}y\text{-path that contains } u\}$$

be the set of all nodes that lie on a shortest x-y-path.

We call a graph G *sp-unique* if, for any pair of nodes s and t in G, there is at most one, unique shortest s-t-path in G. Let $P = (x_1, x_2, \ldots, x_k)$ be a path. The *hop-length* $|P|$ of P is $k-1$. Given two nodes s and t, the *hop-distance* $h_G(s,t)$ from s to t is the minimum hop-length of any shortest s-t-path in G and 0 if there is no s-t-path in G or if $s = t$. We abbreviate $h_G(s,t)$ by $h(s,t)$ if the choice of the graph G is clear. We further assume that for each edge (u,v) in G it is $\text{len}(u,v) = \text{dist}(u,v)$. This can easily be assured by deleting edges (u,v) with $\text{len}(u,v) > \text{dist}(u,v)$ in a preprocessing step. This guarantees that, after the insertion of a shortcut (a,b), there is only one edge (a,b) in the graph.

3 Problem Statement and Complexity

In this section, we introduce the SHORTCUT PROBLEM and the REVERSE SHORTCUT PROBLEM. We show that both problems are NP-hard. Moreover, we show that there is no polynomial-time constant-factor approximation algorithm for the REVERSE SHORTCUT PROBLEM and no polynomial-time algorithm that approximates the SHORTCUT PROBLEM up to an additive constant unless P = NP. Finally, we identify a critical parameter of the SHORTCUT PROBLEM and discuss some monotonicity properties of the problem.

In the following, we augment a given graph G with *shortcuts*. These are edges (u,v) that are added to G such that $len(u,v) = dist(u,v)$. A set of shortcuts is called a *shortcut assignment*.

Definition (Shortcut Assignment). Consider a graph $G = (V,E,\text{len})$. A *shortcut assignment* for G is a set $E' \subseteq (V \times V) \setminus E$ such that, for any (u,v) in E', it is $\text{dist}(u,v) < \infty$. The notation $G[E']$ abbreviates the graph G with the shortcut assignment E' added, i.e., the graph $(V, E \cup E', \text{len}')$ where $\text{len}' : E \cup E' \to \mathbb{R}_{>0}$ equals $dist(u,v)$ if $(u,v) \in E'$ and equals $len(u,v)$ otherwise.

When working with shortcuts we are interested in the expected number of edges that are contained in an edge-minimal shortest path from a random node s to a random node t. The *gain* of a shortcut assignment E' measures how much this value decreases due to the graph-augmentation with E'.

Definition (Gain). Given a graph $G = (V,E,\text{len})$ and a shortcut assignment E', the *gain* $w_G(E')$ of E' is

$$w_G(E') := \sum_{s,t \in V} h_G(s,t) - \sum_{s,t \in V} h_{G[E']}(s,t) \; .$$

We abbreviate $w_G(E')$ by $w(E')$ in case the choice of the graph G is clear.

We briefly consider an augmented graph $G[E'] = (V, E \cup E', \text{len}')$ and choose nodes s and t uniformly at random. The expected number of edges on an edge-minimal shortest s-t-path is $\frac{1}{|V|^2} \sum_{s,t \in V} h_{G[E']}(s,t)$ when we count pairs s and t with $\text{dist}(s,t) = \infty$ by 0. The term $\sum_{s,t \in V} h_G(s,t)$ does not depend on E' and hence is constant. Consequently, maximizing the gain and minimizing the expected number of edges on edge-minimal shortest-paths are equivalent problems. The SHORTCUT PROBLEM consists of adding a number c of shortcuts to a graph, such that the gain is maximal.

Problem (SHORTCUT PROBLEM). Let $G = (V,E,\text{len})$ be a graph and $c \in \mathbb{Z}_{>0}$ be a positive integer. Given an instance (G,c), the SHORTCUT PROBLEM is to find a shortcut assignment E' with $|E'| \leq c$ such that the gain $w_G(E')$ of E' is maximal.

The REVERSE SHORTCUT PROBLEM searches for a shortcut assignment E' of minimum cardinality achieving at least some given gain k. We assure that such a solution exists by stating an upper bound on k. To obtain k, we first compute the number

$$\left| \{(u,v) \in V \times V \mid \text{dist}(u,v) < \infty, u \neq v\} \right| \; .$$

This is exactly the value of $\sum_{s,t \in V} h_{G[\overline{S}]}(s,t)$ when inserting all possible shortcuts $\overline{S}$ to G. Then we subtract this value from $\sum_{s,t \in V} h_G(s,t)$ to yield a sharp bound on the gain.

Problem (REVERSE SHORTCUT PROBLEM). Let $G = (V,E,\text{len})$ be a graph and $k \in \mathbb{Z}_{>0}$ be less than or equal to $\sum_{s,t \in V} h_G(s,t) - |\{(u,v) \in V \times V \mid \text{dist}(u,v) < \infty, u \neq v\}|$. Given an instance (G,k) the REVERSE SHORTCUT PROBLEM is to find a shortcut assignment E' such that $w_G(E') \geq k$ and such that $|E'|$ is minimal.

As an auxiliary problem to shorten proofs we also consider the SHORTCUT DECISION PROBLEM.

Problem (SHORTCUT DECISION PROBLEM). Let $G = (V, E, \mathrm{len})$ be a graph and $c, k \in \mathbb{Z}_{>0}$ be positive integers. Given an instance (G, c, k), the SHORTCUT DECISION PROBLEM is to decide if there is a shortcut assignment E' for $G = (V, E, \mathrm{len})$ such that $w_G(E') \geq k$ and $|E'| \leq c$.

In order to show the complexity of the problems we make a transformation from SET COVER and MIN SET COVER.

Definition (SET COVER and MIN SET COVER). Let C be a collection of subsets of a finite set U such that $\bigcup_{c \in C} c = U$ and let $k \in \mathbb{Z}_{>0}$ be a positive integer. A *set cover* of (C, U) is a subset C' of C such that every element in U belongs to at least one member of C'. Given an instance (C, U), the problem MIN SET COVER is to find a set cover C' of (C, U) of minimum cardinality. Given an instance (C, U, k), the problem SET COVER is to decide if there is a set cover C' of (C, U) of cardinality no more than k. The *size* of a MIN SET COVER instance (C, U) is $\sum_{c \in C} |c|$.

Notation (Solution). Given a $\{$SHORTCUT PROBLEM, REVERSE SHORTCUT PROBLEM, MIN SET COVER$\}$-instance I, we denote by $\mathrm{opt}_{\{\mathrm{SP,RSP,MSC}\}}(I)$ an arbitrary (optimal) solution of I of the according problem.

We now show a relationship between SET COVER and the SHORTCUT PROBLEM.

Lemma 1. Let (C, U, k) be a SET COVER-instance. There is a graph $G = (V, E, \mathrm{len})$ such that there is a set cover C' for (C, U) of cardinality $|C|' \leq k$ if, and only if there is a shortcut assignment E' for G of cardinality $|E|' \leq k$ and gain $w(E') \geq (2|C| + 1)|U|$. Further, the size of G and the time to compute G is polynomial in the size of (C, U). Finally, given a shortcut assignment E' with $w(E') \geq (2|C| + 1)|U|$, we can compute a set cover of cardinality at most $|E'|$ in time polynomial in the size of (C, U, k).

Proof: Given an instance (C, U, k) of SET COVER, we construct a graph $G = (V, E, \mathrm{len})$ as follows, see Figure 1 for an illustration: We denote the value $2|C| + 1$ by Δ. We introduce a node s to G. For each $u \in U$, we introduce a set of nodes $U_u = \{u_1, \ldots, u_\Delta\}$ to G. For each c in C, we introduce nodes c^-, c^+ and edges (c^-, c^+), (c^+, s) to G. The graph furthermore contains, for each $u \in U$ and each $c \in C$ with $u \in c$, the edges $(u_r, c^-), r = 1, \ldots, \Delta$. All edges are directed and have length 1. We abbreviate $\overline{U} := \bigcup_{u \in U} U_u$, $C^- := \{c^- | c \in C\}$ and $C^+ := \{c^+ | c \in C\}$.

We first observe that shortcuts in G are always contained in one of the following three sets: $\overline{U} \times \{s\}, C^- \times \{s\}$ and $\overline{U} \times C^+$. Given $u \in U$, we say u is *covered* by a shortcut $(c^-, s) \in C^- \times \{s\}$ if $u \in c$.

Claim. Let C' be a set cover of (C, U). Then, the shortcut assignment $E' = \{(c^-, s) \mid c \in C'\}$ fulfills $|E'| = |C'|$ and $w(E') \geq \Delta|U|$.

Obviously $|E'| = |C'|$ holds. For each node $v \in \overline{U}$ the hop-distance to node s decreases by 1 due to the insertion of E'. As $|\overline{U}| = \Delta|U|$, it is $w(E') \geq \Delta|U|$.

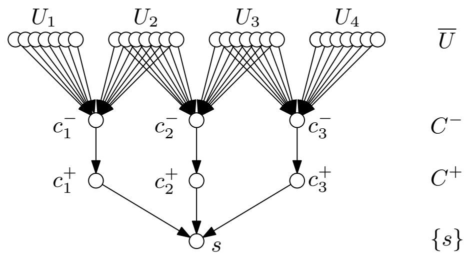


Figure 1: Graph $G = (V, E)$ constructed from the SET COVER-instance $\{c_1 = \{1,2\}, c_2 = \{2,3\}, c_3 = \{3,4\}\}$.

Claim. Let E' be a shortcut assignment of G with $w(E') \geq \Delta|U|$. Then, we can construct a shortcut assignment $E'' \subseteq C^- \times \{s\}$ of G with cardinality $|E''| \leq |E|$ and $w(E'') \geq \Delta|U|$ in polynomial time.

We first check if $|E'| > |C|$. If this is the case, we set $E'' := \{(c^-, s) | c \in C\}$ and terminate. Otherwise, we proceed as follows until $E' \subseteq C^- \times \{s\}$ or each $u \in U$ is covered by a shortcut (c^-, s): We choose a shortcut (x, y) in $E' \cap (\overline{U} \times C^+ \cup \overline{U} \times \{s\})$. We further choose a shortcut $(c^-, s) \in V \times V$ such that there is a $u \in c$ which is not covered by any shortcut in E'. Then, we set $E' := (E' \cup \{(c^-, s)\}) \setminus \{(x, y)\}$.

The removal of a shortcut in $\overline{U} \times C^+ \cup \overline{U} \times \{s\}$ decreases the gain by at most 2. Let $u \in U$ be an element that is not covered by a shortcut in E' and let $u \in c \in C$. The insertion of (c^-, s) in E' improves the hop distance $h(v, s)$ for each node in $v \in U_u$ which is not part of a shortcut in E' by 1. As there are $2|C| + 1$ nodes in U_u and we have at most $|C|$ shortcuts, the gain increases by at least $2|C| + 1 - |C|$. Summarizing, at each step $w(E')$ increases at least by $2|C| + 1 - |C| - 2 = |C| - 1 \geq 0$. Any shortcut assignment that covers all $u \in U$ results in the desired gain. Hence, after termination $E'' := E' \cap (C^- \times \{s\})$ gives a solution to the claim.

Claim. Let E' be a shortcut assignment of G with $w(E') \geq \Delta|U|$. Then, we can compute in polynomial time a set cover C' for (C, U) of cardinality at most $|E'|$.

We use the last claim to transform E' such that $E' \subseteq C^- \times \{s\}$ and $w(E') \geq \Delta|U|$. It is $w(E') = |E'| + \Delta|\{u \in U \mid u \text{ is covered by a shortcut in } E'\}| \geq \Delta|U|$. This implies that each $u \in U$ is covered by a shortcut in E' and $\{c | (c^-, s) \in E'\}$ is a set cover of (C, U). □

Theorem 1. The SHORTCUT DECISION PROBLEM is NP-complete.

Proof: Let (C, U, k) be a SET COVER-instance and G be constructed as described in Lemma 1. It is (C, U, k) a yes-instance if and only if the SHORTCUT DECISION PROBLEM-instance $(G, |k|, (|2|C| + 1)|U|)$ is a yes-instance, and the transformation is polynomial. □

We remember that an optimization problem P is NP-hard if there is an NP-hard decision problem P' such that following holds: Problem P' can be solved by a polynomial-time algorithm which uses an oracle that, for any instance of P, returns –in constant time– an optimal solution along with its value.

Corollary. The SHORTCUT PROBLEM and the REVERSE SHORTCUT PROBLEM are NP-hard.

The transformation applied in Lemma 1 also preserves part of the non-approximability of MIN SET COVER.

Theorem 2. Unless P = NP, no polynomial-time constant-factor approximation algorithm exists for the REVERSE SHORTCUT PROBLEM, i.e., there is no combination of an algorithm apx and an *approximation ratio* $\alpha > 0$ such that

- $\mathrm{apx}(G,k)$ is a shortcut assignment for G of gain at least k
- $|\mathrm{apx}(G,k)|/|\mathrm{opt}_{\mathrm{RSP}}(G,k)| \leq \alpha$ for all instances (G,k) of the REVERSE SHORTCUT PROBLEM
- the runtime of $\mathrm{apx}(G,k)$ is polynomial in the size of (G,k).

Proof: Given a MIN SET COVER-instance (C,U), assume to the contrary that there is a polynomial-time constant-factor approximation apx of the REVERSE SHORTCUT PROBLEM with approximation ratio α. Using apx, we construct a constant-factor approximation algorithm for MIN SET COVER, contradicting the fact that MIN SET COVER is not contained in the class APX unless P = NP [4]:

As described in Lemma 1, we first construct the graph G. Then we compute $E' = \mathrm{apx}(G,(2|C|+1)|U|)$ and finally transform E' to a set cover C' of (C,U) of size at most $|E'|$. With Lemma 1 we have that

$$|\mathrm{opt}_{\mathrm{MSC}}(C,U)| = |\mathrm{opt}_{\mathrm{RSP}}(G,(2|C|+1)|U|)| \ .$$

Hence it is

$$|C'|/|\mathrm{opt}_{\mathrm{MSC}}| \leq |E'|/|\mathrm{opt}_{\mathrm{RSP}}(G,(2|C|+1)|U|)| \leq \alpha$$

which shows the theorem. □

Using a stronger result on the inapproximability of the MIN SET COVER-problem, we get an asymptotically tighter lower bound on the approximation factor of the REVERSE SHORTCUT PROBLEM.

Proposition. Unless P = NP, no polynomial-time algorithm exists that approximates the REVERSE SHORTCUT PROBLEM to a factor $\Omega\big(\log(\log|V|)\big)$.

Proof: By [3], MIN SET COVER is not approximable within a factor $\eta \cdot \ln|U|$, for a certain constant η. Assume that there is a polynomial-time approximation algorithm apx for the REVERSE SHORTCUT PROBLEM such that $|\mathrm{apx}(G,k)|/|\mathrm{opt}_{\mathrm{RSP}}(G,k)| \leq \eta \cdot \ln\big(\frac{\log(|V|)}{2} - 2\big)$ for all instances $(G=(V,E),k)$ of the REVERSE SHORTCUT PROBLEM.

Let (C,U) be an arbitrary instance of MIN SET COVER. Analogous to the proof of Theorem 2, we construct a graph $G=(V,E)$ with $(2|C|+1)|U|+2|C|+1$ nodes and a set cover C' in polynomial-time such that $|C'|/|\mathrm{opt}_{\mathrm{MSC}}(C,U)| \leq \eta \cdot \ln\big(\frac{\log(|V|)}{2} - 2\big)$.

Our goal is to obtain an upper bound on $\log(|V|)$ depending on $|U|$, thus we aim to get an upper bound on $|V|$ in the form 2^x. As $|C| \leq 2^{|U|}$ and $|U| \leq 2^{|U|}$, it is

$$\begin{aligned}|V| &\leq (2^{|U|+1}+1)|U| + 2^{|U|+1} + 1 \leq (2^{|U|+1}+1)2^{|U|+1} + 2^{|U|+1} + 1\\ &= 2^{2|U|+2} + 2^{|U|+2} + 1 \leq 2^{2|U|+4}\end{aligned}$$

Hence, it is $2|U|+4 \geq \log(|V|)$ and thus $|C'|/|\operatorname{opt}_{\text{MSC}}(C,U)| \leq \eta \cdot \ln |U|$, contradicting the inapproximability of Min Set Cover. □

Theorem 3. Unless P = NP, no polynomial-time algorithm exists that approximates the Shortcut Problem up to an additive constant, i.e., there is no combination of an algorithm apx and a *maximum error* $\alpha \in \mathbb{R}_{>0}$ such that

- $\operatorname{apx}(G,c)$ is a shortcut assignment for G of cardinality at most c
- the runtime of $\operatorname{apx}(G,c)$ is polynomial in the size of (G,c)
- $w_G(\operatorname{opt}_{\text{SP}}(G,c)) - w_G(\operatorname{apx}(G,c)) \leq \alpha$ for all instances (G,c) of the Shortcut Problem.

Proof:

Assume to the contrary that there is an polynomial-time algorithm apx that approximates the Shortcut Problem up to an additive constant maximum error α and let $(G = (V,E,\text{len}),c,k)$ be a Shortcut Decision Problem-instance. To assure $\alpha \in \mathbb{Z}^+$, we set $\alpha := \lceil \alpha \rceil$. We construct an instance $(\overline{G} = (\overline{V},\overline{E},\overline{\text{len}}),c)$ of the Shortcut Problem by adding to G, for each node $v \in V$, exactly $\chi := \alpha + 1 + |V|^2$ nodes $v_1,\ldots,v_\chi$ and directed edges $(v_1,v),\ldots,(v_\chi,v)$. We further set $\overline{\text{len}}(v_i,v) = 1$ for $i = 1\ldots\chi$. This construction can be done in polynomial time. Let E' denote $\operatorname{apx}(\overline{G},c)$.

Our aim is to solve $(G = (V,E,\text{len}),c,k)$ in polynomial time. We can insert at most $c_{\max} := |\{(u,v) \in V \times V \setminus E \,|\, \operatorname{dist}(u,v) < \infty, u \neq v\}|$ shortcuts into G. If $c \geq c_{\max}$ we can decide the problem in polynomial time by adding all possible shortcuts and computing the according gain. Hence, in the following we may assume $c < c_{\max}$.

Claim. The endpoints of all shortcuts inserted by apx in $\overline{G}$ lie in V, i.e $E' \subseteq V \times V$.

If a shortcut in $\overline{G}$ is not contained in $V \times V$, it must be contained in $\overline{V} \times V$ because of the edge directions in $\overline{G}$. Assume that there is a shortcut $(\overline{u},v) \in E'$ such that $(\overline{u},v) \in (\overline{V} \setminus V) \times V$. Removing $(\overline{u},v)$ from E' will decrease the gain $w_{\overline{G}}(E')$ by at most $|V|^2$ (as it represents only paths starting from $\overline{u}$ of length at most $|V|+1$). Afterwards inserting an arbitrary shortcut $(x,y) \in V \times V$ increases the gain $w_{\overline{G}}(E' \setminus \{(\overline{u},v)\})$ by at least χ (as it represents at least χ paths ending at y of length at least 2). Summarizing,

$$w_{\overline{G}}((\{(x,y)\} \cup E') \setminus \{(\overline{u},v)\}) - w_{\overline{G}}(E') \geq \chi - |V|^2 > \alpha$$

contradicting the approximation guarantee of apx.

Claim. We can use apx to decide $(G = (V,E,\text{len}),c,k)$ in polynomial time contradicting the assumption P $\neq$ NP.

An exact algorithm can be seen as an approximation algorithm with maximum error $\alpha = 0$. We can show in a similar fashion as in the last claim that an optimal solution of

$(\overline{G},c)$ only consists of shortcuts in $V \times V$, i.e., $\text{opt}_{\text{SP}}(\overline{G},c) \subseteq V \times V$. Given a shortcut assignment $E'' \in V \times V$, it is $w_{\overline{G}}(E'') = (1+\chi) \cdot w_G(E'')$. Given an optimal solution E^* for (G,c) and $(\overline{G},c)$, it follows

$$(1+\chi)\left(w_G(E^*) - w_G(E')\right) = w_{\overline{G}}(E^*) - w_{\overline{G}}(E') \leq \alpha.$$

Hence, $w_G(E^*) - w_G(E') \leq \alpha/(1+\chi) < 1$ which implies $w_G(E^*) = w_G(E')$ as both $w_G(E^*)$ and $w_G(E')$ are integer valued. This shows the claim and finishes the proof. □

To obtain a better intuition on the SHORTCUT PROBLEM, we report some properties of the problem.

Trivial approximation bounds. Consider an arbitrary non-empty shortcut assignment E'. It is $0 \leq \sum_{s,t \in V} h_G(s,t) \leq |V|^3$ for any graph $G = (V, E, \text{len})$ and hence $w_G(E') \leq |V|^3$. As each shortcut in E' decreases the hop-distance from its start to its end-node by at least one, we have that each E' is a trivial factor $|V|^3/|E'|$-approximation of the SHORTCUT PROBLEM. Furthermore, any shortcut assignment achieving the desired gain is a trivial factor $|V|^2$-approximation of the REVERSE SHORTCUT PROBLEM.

Bounded number of shortcuts. If the number of shortcuts we are allowed to insert is bounded by a constant k_{max}, the number of possible solutions of the SHORTCUT PROBLEM is at most

$$\binom{|V|^2}{k_{max}} = \frac{|V|^2!}{(|V|^2 - k_{max})!k_{max}!} \leq |V|^{2k_{max}}.$$

This is polynomial in the size of the input graph $G = (V, E, \text{len})$. We can evaluate a given shortcut assignment by basically computing all-pairs shortest-paths, hence this can be done in time $O(|V|^2 \log |V| + |V||E|)$ using Dijkstra's algorithm. For this reason, the case with bounded number of shortcuts can be solved in polynomial time by a brute-force algorithm.

Monotonicity. In order to show the hardness of working with the problem beyond the complexity results, Figure 2 gives an example that, given a shortcut assignment S and an additional shortcut $s \notin S$, the following two inequalities do not hold in general:

$$w(S \cup \{s\}) \geq w(S) + w(s) \quad (1)$$
$$w(S \cup \{s\}) \leq w(S) + w(s). \quad (2)$$

It is easy to verify that in Figure 2 the inequalities $w(\{s_1, s_2\}) > w(s_1) + w(s_2)$ and $w(\{s_1, s_2, s_3\}) < w(\{s_1, s_2\}) + w(s_3)$ hold.

Note that Inequality 2 holds if, for any pair of nodes (s,t) of graph G, there is at most one, unique shortest s-t-path in G. We call such a graph *sp-unique* and prove that fact in the following lemma.

Lemma 2. Given an sp-unique graph $G = (V, E, \text{len})$ and a set of shortcuts S with $S = \{s_1, s_2, \ldots, s_k\}$. Then, $w_G(S) \leq \sum_{i=1}^{k} w_G(s_i)$ and $w_G(S) \leq w_G(\{s_1, \ldots s_{k-1}\}) + w_G(s_k)$.

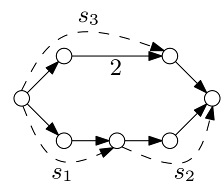


Figure 2: Example Graph G with shortcuts s_1, s_2, s_3. All edges for which no weight is given in the picture have weight 1.

Proof: Given arbitrary but fixed $a, b \in V$ we denote by $w_G^{ab}(S)$ the gain of S on graph G restricted to shortest a-b-paths, i.e., $w_G^{ab}(S) = h_G(a,b) - h_{G[S]}(a,b)$. Because of $w_G(S) = \sum_{u,v \in V} w_G^{uv}(S)$ it suffices to show $w_G^{ab}(S) \leq w_G^{ab}(\{s_1, \ldots s_{k-1}\}) + w_G^{ab}(s_k)$. The inequality $w_G^{ab}(S) \leq \sum_{i=1}^{k} w_G^{ab}(s_i)$ then follows by induction. We write $s_k = (x,y)$. It is

$$w_G^{ab}(S) = w_G^{ab}(\{s_1, \ldots, s_{k-1}\}) + w_{G[s_1,\ldots,s_{k-1}]}^{ab}(\{(x,y)\}).$$

If $(a,b) \in P(x,y)$, we have

$$w_{G[s_1,\ldots,s_{k-1}]}^{ab}(\{(x,y)\}) \leq h_{G[s_1,\ldots,s_{k-1}]}(x,y) - 1 \leq h_G(x,y) - 1 = w_G^{ab}(s_k).$$

Further, if $(a,b) \notin P(x,y)$ we have $w_{G[s_1,\ldots,s_{k-1}]}^{ab}(\{(x,y)\}) = 0 = w_G^{ab}(s_k)$. Hence, we have

$$w_G^{ab}(S) \leq w_G^{ab}(\{s_1, \ldots s_{k-1}\}) + w_G^{ab}(s_k)$$

which shows the lemma. □

Later, we use these results to present an approximation algorithm for sp-unique graphs.

4 ILP-Approaches

In this section we present two exact, ILP-based approaches for the SHORTCUT PROBLEM. Throughout this section, we are given an instance $(G = (V, E, \text{len}), c)$ of the SHORTCUT PROBLEM that is to be solved optimally.

For a vertex $x \in V$, we denote by P_x the set of all vertices $u \in V$ for which an x-u-path exists. Remember that we denote by $P^+(x,y)$ the set of all vertices $u \in V$ for which a shortest x-u path containing y exists and that we denote by $P^{\bowtie}(x,y)$ the set of all vertices that lie on a shortest x-y-path. We assume that all distances in the graph are precomputed and hence that the sets P_x, $P^{\bowtie}(x,y)$ and $P^+(x,y)$ are known for all $x, y \in V$.

Simple ILP-Formulation. The following ILP-formulation (SLSP) is straightforward and simple but has the drawback to incorporate $O(|V|^4)$ variables and constraints. The interpretation of the ILP is as follows: The variables $k_t^s(\cdot,\cdot)$ represent an edge-minimal shortest s-t-path in the augmented graph. It is $k_t^s(u,v) = 1$ if and only if the edge (u,v)

is used in this path. We characterize all edges or possible shortcuts (u,v) that can be used for a shortest s-t-path by introducing the set

$$A := \{(s,u,v,t) \in V^4 \mid \text{dist}(s,u) + \text{dist}(u,v) + \text{dist}(v,t) = \text{dist}(s,t) < \infty,\ u \neq v\}.$$

Consequently, for fixed $s,v,t \in V$, the set $\{u \in V \mid (s,u,v,t) \in A\}$ contains each node u such that the edge or shortcut (u,v) can be used in a shortest s-t-path. The variable $c(u,v)$ equals 1 if the computed shortcut assignment contains (u,v). Instead of maximizing the gain, our aim is to minimize the sum of all hop-distances in the augmented graph. This value equals the sum of all variables $k_t^s(u,v)$ with $(s,u,v,t \in A)$.

$$\text{(SLSP)} \qquad \text{minimize} \sum_{(s,u,v,t)\in A} k_t^s(u,v) \tag{3}$$

such that

$$\sum_{\{v\in V \mid (s,v,t,t)\in A\}} k_t^s(v,t) = 1 \qquad s \in V,\ t \in P_s \setminus \{s\} \tag{4}$$

$$\sum_{\{u\in V \mid (s,u,v,t)\in A\}} k_t^s(u,v) = \sum_{\{w\in V \mid (s,v,w,t)\in A\}} k_t^s(v,w) \qquad \begin{matrix} s \in V,\ t \in P_s \setminus \{s\} \\ v \in P^{\bowtie}(s,t),\ v \neq s,t \end{matrix} \tag{5}$$

$$k_t^s(u,v) \leq c(u,v) \qquad (s,u,v,t) \in A,\ (u,v) \notin E \tag{6}$$

$$\sum_{(u,v)\in (V\times V)\setminus E} c(u,v) \leq c \tag{7}$$

$$k_t^s(u,v) \in \{0,1\} \qquad (s,u,v,t) \in A \tag{8}$$

$$c(u,v) \in \{0,1\} \qquad (u,v) \in V \times V \setminus E \tag{9}$$

Constraint 4 and Constraint 5 ensure that a shortest path is considered for every s-t-pair: Constraint 4 requires that each target t owns exactly one incoming edge on an s-t-path while Constraint 5 guarantees that, for each node $v \neq s,t$, there is an incoming edge (on an s-t-path) if there is an outgoing edge (on such a path). Constraint 6 forces shortcuts to be present whenever edges are used that are not present in the graph. Finally, Constraint 7 limits the number of shortcuts to be inserted. Consequently, a solution of model (SLSP) gives an optimal solution of (G,c): The set $\{(u,v) \in V \times V | c(u,v) = 1\}$ is a shortcut assignment for G of maximum gain and cardinality at most c.

Obviously, there can be more than one edge-minimal shortest path from a given source to a given target. Hence, the model may incorporate unwanted symmetries. In order to break these symmetries one could use additional constraints. We did not further pursue this direction because of the huge number of constraints that would be necessary. Note that the model stays correct when relaxing Constraint 8 to

$$k_t^s(u,v) \in [0,1] \qquad (s,u,v,t) \in A.$$

Flow-Based ILP-Formulation. The number of variables and constraints of the following integer linear program (LSP) is cubic in $|V|$. The model exhibits two types of variables. It is $c(u,v) = 1$ if and only if the solution found uses the shortcut (u,v). Instead of directly counting the hop-distance for each pair of nodes, we use a flow-like formulation that counts, for each edge, how often it is used in the solution. In detail, the value of $f^s(u,v)$ can be interpreted as the number of vertices t for which the hop-minimal shortest s-t-path found by (LSP) includes the edge or shortcut (u,v). To characterize all possible combinations of $s,u,v \in V$ such that (u,v) could be an edge or a shortcut in the shortest-paths subgraph with root s, we introduce the set

$$B := \{(s,u,v) \in V^3 \mid \mathrm{dist}(s,u) + \mathrm{dist}(u,v) = dist(s,v) < \infty,\ \ u \neq v\}\ .$$

The flow outgoing from source s is exactly the number of vertices reachable from s (Constraint 11). As each node consumes exactly one unit of flow (Constraint 12), it is assured that a shortest path from s to any reachable node is considered. Constraint 13 forces shortcuts to be present whenever edges are used that are not present in the graph. Finally, Constraint 14 limits the number of shortcuts to be inserted. Again, instead of maximizing the gain, our aim is to minimize the sum of all hop-distances in the augmented graph which is given as $\mathrm{obj}(f,c)$.

$$\text{(LSP)} \qquad \text{minimize} \ \ \mathrm{obj}(f,c) := \sum_{(s,u,v)\in B} f^s(u,v) \tag{10}$$

such that

$$\sum_{\{v\in V \mid (s,s,v)\in B\}} f^s(s,v) = |P_s| - 1 \qquad s \in V \tag{11}$$

$$\sum_{\{u\in V \mid (s,u,v)\in B\}} f^s(u,v) = 1 + \sum_{\{w\in V \mid (s,v,w)\in B\}} f^s(v,w) \qquad s \in V,\ v \in P_s,\ v \neq s \tag{12}$$

$$f^s(u,v) \leq |P^+(s,v)| \cdot c(u,v) \qquad (s,u,v) \in B,\ (u,v) \notin E, \tag{13}$$

$$\sum_{(u,v)\in(V\times V)\setminus E} c(u,v) \leq c \tag{14}$$

$$f^s(u,v) \in \mathbb{Z}_{\geq 0} \qquad (s,u,v) \in B \tag{15}$$

$$c(u,v) \in \{0,1\} \qquad (u,v) \in V \times V \setminus E \tag{16}$$

We now prove the correctness of model (LSP). The proof of the following preparatory lemma shows that a solution of (LSP) can be converted into a solution of equal objective value that, for each node, induces a shortest-paths tree.

Lemma 3. There exists an optimal solution (f,c) of (LSP), such that for each $s \in V$, the subgraph induced by $T_s := \{(u,v) \in V \times V \mid f^s(u,v) > 0\}$ is a tree.

Proof: Let (f,c) be a solution of (LSP). Then T_s is a directed acyclic graph with root s as T_s is contained in the shortest-paths subgraph of G with root s. As long as T_s is not a tree proceed as follows:

First, consider an arbitrary node y such that there are two edges (v,y) and (w,y) in T_s. Let x be an arbitrary node such that there are disjoint x-y-paths P_1 and P_2 in T_s. Such a node x has to exist as there is more than one shortest s-y-path in T_s and we can take any topologically maximal node x for which there is more than one x-y-path. Let (y',y) be the last edge on P_1 and $\Delta := f^s(y',y)$. For each edge e on P_1 we set $f^s(e) := f^s(e) - \Delta$, for each edge e on P_2 we set $f^s(e) := f^s(e) + \Delta$.

It is easy to verify that this does not change the feasibility of the solution. Obviously, the objective function cannot decrease because of this operation as (f,c) is optimal. Further, the objective function may not increase: Assume the contrary. Then P_2 contains more edges than P_1. Let (y'',y) be the last edge of P_2 and $\Delta' := f^s(y'',y)$. We would obtain a better feasible solution by setting $f^s(e) := f^s(e) - \Delta'$ for each edge $e \in P_2$ and $f^s(e) := f^s(e) + \Delta'$ for each edge $e \in P_1$, contradicting the optimality of (f,c). □

The following theorem shows that model (LSP) and the SHORTCUT PROBLEM are equivalent with regard to exact solutions.

Theorem 4. Given an optimal solution E' of the SHORTCUT PROBLEM, the assignment

$$c'(u,v) := \begin{cases} 1 & ,(u,v) \in E' \\ 0 & ,\text{otherwise} \end{cases}$$

can be extended to an optimal solution of (LSP). Further, given an optimal solution (f,c) of (LSP), the set

$$E'' := \{(u,v) \in V \times V \mid c(u,v) = 1\}$$

is an optimal solution for the SHORTCUT PROBLEM.

Proof: Let $(G = (V,E,\text{len}),c)$ be a SHORTCUT PROBLEM-instance. As we have observed before, maximizing the gain is equivalent to finding a shortcut assignment E' that minimizes $\text{obj}(E') := \sum_{s,t \in V} h_{G[E']}(s,t)$. Throughout this proof, we use this point of view.

Let E' be a shortcut assignment of $(G = (V,E,\text{len}),c)$. Consider an arbitrary vertex $s \in V$. There is a shortest-paths tree $T_s \subseteq G[E']$ such that, for each $t \in V$ with $\text{dist}(s,t) < \infty$, the number of edges on the s-t-path in T_s equals $h_{G[E']}(s,t)$. Such a tree T_s can be computed using Dijkstra's algorithm by altering the distance labels to be tuples (edge length, hop distance) and applying lexicographical ordering. Let

$$c'(u,v) = \begin{cases} 1 & ,(u,v) \in E' \\ 0 & ,\text{otherwise} \end{cases}$$

and

$$f'^s(u,v) = \begin{cases} 0 & ,(u,v) \notin T_s \\ |\{w \mid w \text{ is descendant of } v \text{ in } T_s\}| & ,\text{ otherwise} \end{cases}$$

The pair (c', f') is a feasible solution of (LSP). We denote by $P_{T_s}(s,t)$ the s-t-path in T_s and by $|P_{T_s}(s,t)|$ the number of edges on this path. It is

$$\sum_{t\in P_s} h_{G[E']}(s,t) = \sum_{t\in P_s} |P_{T_s}(s,t)| = \sum_{t\in P_s}\sum_{e\in T_s} 1_e(P_{T_s}(s,t)) = \sum_{e\in T_s}\sum_{t\in P_s} 1_e(P_{T_s}(s,t))$$
$$= \sum_{(u,v)\in T_s} |\{w \mid w \text{ is descendant of } v \text{ in } T_s\}| = \sum_{u\in P_s,\ v\in P^+(s,u),\ u\neq v} f'^s(u,v)$$

Consequently, $\mathrm{obj}(f', c') = \mathrm{obj}(E')$.

On the other hand, let (f,c) be a feasible solution of (LSP). With Lemma 3 we may assume that, for each node s, the subgraph induced by $T_s := \{(u,v) \in V \times V \mid f^s(u,v) > 0\}$ is a tree. Hence, we can show by induction that $f^s(u,v) = |\{w \mid w \text{ is descendant of } v \text{ in } T_s\}|$ for each edge $(u,v) \in T_s$. Further, the set

$$E'' = \{(u,v) \in V \times V \mid c(u,v) = 1\}$$

is a feasible solution of the SHORTCUT PROBLEM. Finally, we show that $\mathrm{obj}(E'') \leq \mathrm{obj}(f,c)$. We consider each root $s \in V$ separately. To bound the hop-distances in $G[E'']$ starting at s from above we use the shortest-paths tree T_s as a witness. This yields

$$\sum_{t\in P_s} h_{G[E'']}(s,t) \leq \sum_{t\in P_s} |P_{T_s}(s,t)|$$

With the same computation as above, we derive

$$\sum_{t\in P_s} h_{G[E'']}(s,t) \leq \sum_{t\in P_s} |P_{T_s}(s,t)| = \sum_{u\in P_s,\ v\in P^+(s,u),\ u\neq v} f^s(u,v)$$

which shows the claim. □

Tuning the Flow-Based Formulation. In order to simplify model (LSP), we relax Constraint 15 to

$$f^s(u,v) \in \mathbb{R}_{\geq 0} \qquad (s,u,v) \in B \tag{17}$$

and denote the resulting model (10, 11, 12, 13, 14, 16, 17) by (RLSP).

Lemma 4. Let (f,c) be a solution of (RLSP). Then there is a solution (f', c) of (LSP) with same objective value.

Proof: Note that Lemma 3 also holds for (RLSP). Hence, we assume that, for each node s, the subgraph induced by $T_s := \{(u,v) \in V \times V \mid f^s(u,v) > 0\}$ is a tree. The integrality of f now follows by induction on the nodes in reverse topological order and Constraint 12. □

In order to heuristically speedup the solving process we may add the following constraints that give bounds on the f-variables.

$$f^s(u,v) \leq |P^+(s,v)| \qquad (s,u,v) \in B \tag{18}$$

An additional heuristic improvement works as follows: The sum $\sum_{s,t \in V} h_G(s,t)$ is the value of the objective function of model (LSP) in case no shortcuts are allowed. The value $(h_G(a,b)-1) \cdot |P(a,b)|$ is an upper bound for the amount that shortcut (a,b) improves the objective function. We precompute $\sum_{s,t \in V} h_G(s,t)$ and, for each pair (a,b) of connected nodes, the value $(h_G(a,b)-1) \cdot |P(a,b)|$. Then we can add the constraint

$$\underbrace{\sum_{(s,u,v)\in B} f^s(u,v)}_{=\text{obj}(f,c)} \geq \underbrace{\sum_{s,t\in V} h_G(s,t)}_{\text{lower bound of obj}(f,0)} - \sum_{\substack{a,b\in V\\ \text{dist}(a,b)<\infty}} c(a,b) \cdot \underbrace{(h_G(a,b)-1)\cdot|P(a,b)|}_{\substack{\text{upper bound of improvement}\\ \text{because of shortcut } (a,b)}} \tag{19}$$

to additionally tighten the model.

Case Study. While our main interest on the problem is of theoretical nature, we report some experimental results of the ILP-based approaches. This shall allow for a brief comparison of both formulations and for assessing the heuristic improvements. Our implementation is written in Java using CPLEX 11.2 as ILP-Solver and was compiled with Java 1.6. The tests were executed on one core of an AMD Opteron 6172 Processor, running SUSE Linux 10.3. The machine is clocked at 2.1 GHz and has 16 GB of RAM per processor.

We tested on four different graphs. The graph G_{disk} is a unit-disk graph and generated by randomly assigning 100 nodes to a point in the unit square of the Euclidean plane. Two nodes are connected by an edge if their Euclidean distance is below a given radius. This radius is adjusted such that the resulting graph has approximately 1000 edges. The graph G_{ka} represents a part of the road network of Karlsruhe. It contains 102 nodes and 241 edges. The graph G_{grid} is based on a two-dimensional 10×10 square grid. The nodes of the graph correspond to the crossings in the grid. There is an edge between two nodes if they are neighbors on the grid. Finally, the graph G_{path} is a path consisting of 30 nodes. In each graph, edge weights are randomly chosen integer values between 1 and 1000. For each experiment, the computation time has been limited to 60 minutes. The integrality constraints of the variables $k_s^t(\cdot,\cdot)$ of the simple model and the variables $f^s(\cdot,\cdot)$ of the flow model have been relaxed. Some example outcomes are depicted in Figure 3.

The results are summarized in Table 1. Columns mean the following: Columns *Eq19* and *Eq18* indicate if Equation 19 and Equation 18 are incorporated in the model. For the simple model, we adapted Equation 19 in a straightforward fashion. Columns *opt* show if an optimal solution has been found and proven to be optimal. Columns *gap* give the guaranteed approximation ratio of the best feasible solution found within 60 minutes, i.e., the value (*best feasible solution found - best proven lower bound*) / *best proven lower bound*. The value of gap is ∞ if no feasible solution has been found in 60 minutes. Finally, columns *time* give the computation time in minutes.

We observe that the simple model does not benefit from Equation 19 and the plain version without this enhancement is always superior. For the flow formulation, it turned out that the version enriched with Equation 18 is best: This version is always better than the plain model without improvement and than the formulation enhanced only with Equation 19. Further, it is most times better than the version enriched with Equa-

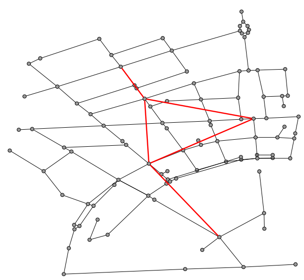
				G_{grid}			G_{ka}			G_{path}			G_{disk}		
shortcuts	model	Eq19	Eq18	opt	gap	time	opt	gap	time	opt	gap	time	opt	gap	time
1	flow			✓	0	2	✓	0	5	✓	0	1	✓	0	2
1	flow		✓	✓	0	2	✓	0	3	✓	0	0	✓	0	1
1	flow	✓		✓	0	4	✓	0	8	✓	0	0	✓	0	3
1	flow	✓	✓	✓	0	2	✓	0	7	✓	0	0	✓	0	2
1	simple			✓	0	16	✓	0	29	✓	0	1	✓	0	14
1	simple	✓		✓	0	18	✓	0	49	✓	0	1	✓	0	24
2	flow				0.02	60		0.09	60		0.2	60	✓	0	12
2	flow		✓	✓	0	10	✓	0	35	✓	0	8	✓	0	2
2	flow	✓		✓	0	17		0.01	60		0.06	60	✓	0	2
2	flow	✓	✓	✓	0	3	✓	0	40	✓	0	9	✓	0	2
2	simple			✓	0	20	✓	0	26	✓	0	2	✓	0	12
2	simple	✓		✓	0	21	✓	0	48	✓	0	2	✓	0	20
5	flow				0.16	60		0.53	60		0.4	60		0.06	60
5	flow		✓	✓	0	28	✓	0	46		0.16	60	✓	0	4
5	flow	✓			0.05	60		0.12	60		0.39	60	✓	0	55
5	flow	✓	✓		0	60		0.01	60		0.17	60	✓	0	9
5	simple			✓	0	30	✓	0	40		0.04	60	✓	0	15
5	simple	✓		✓	0	58		∞	60		∞	60	✓	0	38
10	flow				0.58	60		0.83	60		0.45	60		0.11	60
10	flow		✓		0.03	60		0.49	60		0.27	60	✓	0	27
10	flow	✓			0.14	60		0.49	60		0.49	60		0.07	60
10	flow	✓	✓		0.05	60		0.34	60		0.32	60	✓	0	25
10	simple				∞	60		∞	60		0.47	60	✓	0	22
10	simple	✓			∞	60		∞	60		2.08	60	✓	0	39

Table 1: Experimental results of the ILP-approaches.

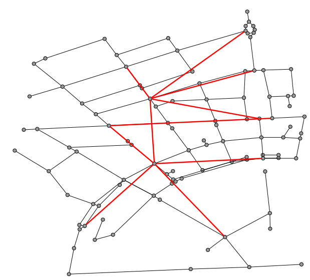
tion 19 and 18. Finally, we see that Equation 19 was an improvement to the plain model if more than one shortcut was to be inserted.

Comparing the two formulations we obtain that the flow formulation is superior. The flow formulation enhanced with Equation 18 was most times better than the simple model, sometimes with a big gap. With one exception, the difference was small when the simple model was better. Concluding, in this testset the flow formulation enhanced with Equation 18 performed best.

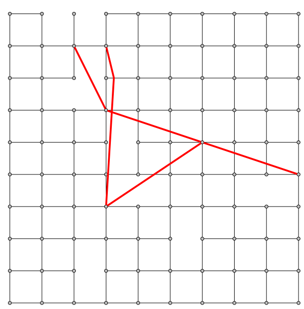
In our experiments, we did not take memory consumption into account as the limiting factor was computation time. However, to enable a vague comparison of the memory consumption, we report in Table 2 the number of nonzeros reported by CPLEX after the presolve routine. Note that, this number turned out to be almost independent from the number of shortcuts to be inserted.



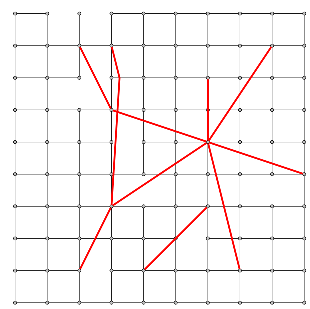
graph *ka* with 5 optimal shortcuts



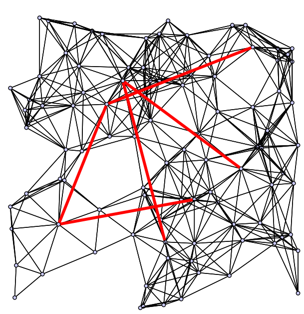
graph *ka* with 10 optimal shortcuts



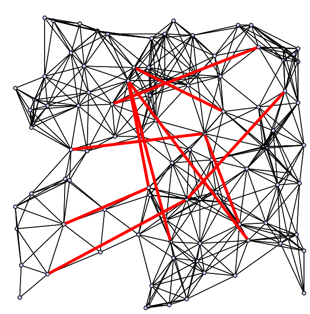
graph *grid* with 5 optimal shortcuts



graph *grid* with 10 optimal shortcuts



graph *disk* with 5 optimal shortcuts



graph *disk* with 10 optimal shortcuts

Figure 3: Optimal shortcut assignments for some example graphs.

model	Eq19	Eq18	G_{grid}	G_{ka}	G_{path}	G_{disk}
flow			274.818	328.102	34.391	249.564
flow		✓	327.022	392.422	41.849	295.460
flow	✓		342.689	409.157	43.029	311.547
flow	✓	✓	394.737	473.390	50.379	357.146
simple			1.241.560	1.724.034	259.211	1.005.390
simple	✓		1.551.052	2.165.022	324.256	1.250.583

Table 2: Number of nonzeros reported by CPLEX after the presolve routine for each model and graph.

5 Approximation using the GREEDY-Strategy

In this section, we propose a polynomial-time algorithm that approximatively solves the SHORTCUT PROBLEM in a greedy fashion. Given the number c of shortcuts to insert, the approach finds a c-approximation of the optimal solution if the underlying graph is sp-unique. While the algorithm works on arbitrary graphs, we restrict our description to strongly connected graphs to improve readability. The restriction to sp-unique graphs is only needed for achieving the approximation guarantee.

Description. Given the instance (G,c), the GREEDY approximation scheme consists of iteratively constructing a sequence $G = G_0, G_1, \ldots, G_c$ of graphs where G_{i+1} results from solving the SHORTCUT PROBLEM on G_i with only one shortcut allowed to insert. The pseudocode for the approach is given as Algorithm 1. The following theorem shows the approximation ratio for GREEDY.

Algorithm 1: GREEDY(G,c)

```
input : graph G = (V, E, len), number of shortcuts c
output: shortcut assignment E'
E' ← ∅; for i = 1, 2, ..., c do
    (x, y) ← argmax_{(x,y)∈(V×V)\(E∪E'), dist(x,y)<∞} {w_{G[E']}({(x, y)})}
    E' ← E' ∪ {(x, y)}
output E'.
```

Theorem 5. Consider an sp-unique graph $G = (V,E,\text{len})$ together with a positive integer $c \in \mathbb{Z}_{>0}$. The solution $E' := \text{GREEDY}(G,c)$ of the GREEDY-approach is a c-approximation of an optimal solution E^*, i.e., $w_G(E^*)/w_G(E') \leq c$.

Proof: Let $e_1 \in E'$ be the first shortcut inserted by GREEDY. Then, $w_G(e) \leq w_G(e_1)$ for each $e \in E^*$. Moreover by Lemma 2, $w(E^*) \leq \sum_{e \in E^*} w(e)$. This yields

$$w_G(E^*) \leq \sum_{e \in E^*} w_G(e) \leq \sum_{i=1}^{c} w_G(e_1) = c \cdot w_G(e_1) \leq c \cdot w_G(E')$$

which shows $w(E^*)/w(E') \leq c$. □

Basic Runtime Issues. The runtime of GREEDY crucially depends on how the next shortcut to be inserted is found. A straightforward approach would be to first precompute the distance $\text{dist}(s,t)$ for each pair $s,t \in V$ as well as the shortest-paths subgraph G_s for each node $s \in V$. Then, the evaluation of a possible shortcut can be done by running breadth-first searches on the $|V|$ graphs G_s. After insertion of a shortcut (a,b) to G, the shortest-paths subgraphs G_s can be adapted by adding (a,b) to each subgraph G_s for which $\text{dist}(s,a)+\text{dist}(a,b)=\text{dist}(s,b)$. Hence G_s contains at most $|E|+c$ edges and the time needed for evaluating one shortcut is $O(|V|\cdot(|V|+|E|+c))$. This leads to a runtime in $O(|V|^2\cdot|V|\cdot(|V|+|E|+c))$ for evaluating all $|V|^2$ possible shortcuts. The runtime $O(|V|^2\log|V|+|V|\cdot|E|)$ of precomputing the shortest-paths subgraphs can be neglected.

In the remainder of this section, we show how to perform this step in time $O(|V|^3)$ using a dynamic program. Consequently, the GREEDY-strategy can be implemented to work in time $O(c\cdot|V|^3)$.

Greedily finding one optimal shortcut in sp-unique graphs. In sp-unique graphs each shortest path is edge-minimal. Hence, we can compute the gain of a shortcut (a,b) restricted to a pair of nodes $(s,t)\in P(a,b)$ by the equation

$$h_G(s,t)-h_{G[(a,b)]}(s,t)=h_G(a,b)-1. \tag{20}$$

Furthermore, for general graphs, the following lemma holds.

Lemma 5. $(s,t)\in P(a,b)$ if and only if $s\in P^-(a,b)$ and $t\in P^+(s,b)$.

Proof: $\Rightarrow$: Let $(s,t)\in P(a,b)$, then there is a shortest s-t-path p containing first a and then b. Obviously, this path also shows that $t\in P^+(s,b)$ and as subpaths of shortest paths are again shortest paths, the subpath of p from s to b is a witness that s is in $P^-(a,b)$.

$\Leftarrow$: Let $s\in P^-(a,b)$ and $t\in P^+(s,b)$. Then, $\text{dist}(s,b)=\text{dist}(s,a)+\text{dist}(a,b)$ and hence $\text{dist}(s,t)=\text{dist}(s,b)+\text{dist}(b,t)=\text{dist}(s,a)+\text{dist}(a,b)+\text{dist}(b,t)$. This shows that $(s,t)\in P(a,b)$. □

Exploiting Lemma 5 and Equation 20, we obtain

$$w(a,b)=(h_G(a,b)-1)\cdot|P(a,b)|=(h_G(a,b)-1)\cdot\sum_{s\in P^-(a,b)}|P^+(s,b)|. \tag{21}$$

This equation directly leads to Algorithm 2 that finds one optimal shortcut for sp-unique graphs. The runtime of the algorithm lies in $O(|V|^3)$ as the computation of $|P^+(s,b)|$ is linear in $|V|$: For each $v\in V$ we have to check if $\text{dist}(s,b)+\text{dist}(b,v)=dist(s,v)$.

The problem of this approach is that we can not apply Algorithm 2 for the GREEDY-strategy, even when the input graph is sp-unique: After insertion of the first shortcut, the augmented graph is not sp-unique any more and hence we can not use Equation 20.

An $O(|V|^3)$-implementation for greedily finding one optimal shortcut. In the following, we generalize the above approach to work with arbitrary graphs. The *offset*

$$\omega_{sb}(t):=h_G(s,b)+h_G(b,t)-h_G(s,t)$$

Algorithm 2: GREEDY STEP ON SP-UNIQUE GRAPHS

input : graph $G = (V, E, \text{len})$, distance table $\text{dist}(\cdot,\cdot)$ of G
output: optimal shortcut (a,b)

initialize $w(\cdot,\cdot) \equiv 0$
compute $h_G(\cdot,\cdot)$
for $s \in V$ **do**
  for $b \in V$ **do**
    compute $|P^+(s,b)|$
    for $a \in V$ **do**
      if $\text{dist}(s,a) + \text{dist}(a,b) = \text{dist}(s,b)$ **then**
        $w(a,b) \leftarrow w(a,b) + (h_G(a,b) - 1)|P^+(s,b)|$
output arbitrary (a,b) with maximum $w(a,b)$

reflects the increase of the hop-distance between given nodes s and t, if we restrict ourselves to shortest paths containing b. We define the *potential gain* $g_s(a,b)$ of a shortcut from a to b with respect to s as

$$g_s(a,b) := h_G(a,b) - 1 - \omega_{sa}(b) \ .$$

This is an upper bound for the decrease of the hop-distance between s and any t in the graph $G[(a,b)]$.

Lemma 6. For all vertices $s,t,a,b \in V$ such that $(s,t) \in P(a,b)$ it holds that

$$h_G(s,t) - h_{G[(a,b)]}(s,t) = \max\{g_s(a,b) - \omega_{sb}(t),\, 0\}.$$

Proof: Directly from the definition of potential gain and offset we obtain

$$g_s(a,b) - \omega_{sb}(t) > 0 \Longleftrightarrow h_G(s,t) > h_G(s,a) + 1 + h_G(b,t) \tag{22}$$

Case $[g_s(a,b) - \omega_{sb}(t) > 0]$. Then $h_G(s,t) > h_G(s,a) + 1 + h_G(b,t)$. The presence of shortcut (a,b) decreases the s-t-hop-distance to $h_{G[(a,b)]}(s,t) = h_G(s,a) + 1 + h_G(b,t)$ as the lemma assumes that there is a shortest s-a-b-t-path. This yields

$$\begin{aligned} h_G(s,t) - h_{G[(a,b)]}(s,t) &= h_G(s,t) - h_G(s,a) - 1 - h_G(b,t) \\ &= h_G(a,b) - 1 \underbrace{- h_G(s,a) - h_G(a,b) + h_G(s,b)}_{= -\omega_{sa}(b)} \\ &\quad \underbrace{- h_G(s,b) - h_G(b,t) + h_G(s,t)}_{= -\omega_{sb}(t)} \\ &= g_s(a,b) - \omega_{sb}(t). \end{aligned}$$

Case $[g_s(a,b) - \omega_{sb}(t) \leq 0]$. With Equation (22) we obtain $h_G(s,t) \leq h_G(s,a) + 1 + h_G(b,t)$. Hence, a shortcut (a,b) does not improve the hop-distance from s to t and we have $h_G(s,t) - h_{G[(a,b)]}(s,t) = 0$. □

Lemma 6 implies that vertices t in $P^+(s,b)$ with the same value of $\omega_{sb}(t)$ benefit from a shortcut ending at b to the same extent, if we restrict ourselves to shortest paths starting at s. We divide the vertices in $P^+(s,b)$ in equivalence classes with respect to ω_{sb}. Let

$$\Delta_i(s,b) := |\{t \in P^+(s,b) \mid \ \omega_{sb}(t) = i\}|$$

be the number of vertices in these equivalence classes.

The algorithm we propose makes use of partial (weighted) sums of the $\Delta_i(s,b)$ for fixed s and b in V. For convenience, we introduce two further abbreviations :

$$\begin{aligned} C_r(s,b) &:= \sum_{i=0}^{r} \Delta_i(s,b) \\ D_r(s,b) &:= \sum_{i=0}^{r} i \cdot \Delta_i(s,b). \end{aligned}$$

With these definitions, we can form an alternative equation for $w(a,b)$.

Lemma 7. Let $a,b,s,t \in V$ be arbitrary nodes. Then

$$w(a,b) = \sum_{\substack{s \in P^-(a,b) \\ g_s(a,b)>0}} \Big(g_s(a,b) \cdot C_{g_s(a,b)-1}(s,b) - D_{g_s(a,b)-1}(s,b) \Big).$$

Proof:

$$\begin{aligned} w(a,b) &= \sum_{s,t \in V} \big(h_G(s,t) - h_{G[(a,b)]}(s,t)\big) \\ &= \sum_{(s,t)\in P(a,b)} \big(h_G(s,t) - h_{G[(a,b)]}(s,t)\big) + \sum_{(s,t)\notin P(a,b)} \underbrace{\big(h_G(s,t) - h_{G[(a,b)]}(s,t)\big)}_{=0} \\ &= \sum_{\substack{(s,t)\in P(a,b) \\ \omega_{sb}(t)<g_s(a,b)}} \big(h_G(s,t) - h_{G[(a,b)]}(s,t)\big) + \sum_{\substack{(s,t)\in P(a,b) \\ \omega_{sb}(t)\geq g_s(a,b)}} \underbrace{\big(h_G(s,t) - h_{G[(a,b)]}(s,t)\big)}_{=0 \text{ with Lemma 6}}. \end{aligned}$$

With Lemma 6, we yield

$$w(a,b) = \sum_{\substack{(s,t)\in P(a,b) \\ \omega_{sb}(t)<g_s(a,b)}} g_s(a,b) - \omega_{sb}(t).$$

It is $\omega_{sb}(t) \geq 0$ as $(s,t) \in P(a,b)$, hence with Lemma 5 we have

$$w(a,b) = \sum_{\substack{s\in P^-(a,b) \\ g_s(a,b)>0}} \sum_{i=0}^{g_s(a,b)-1} \sum_{\substack{t\in P^+(s,b) \\ \omega_{sb}(t)=i}} g_s(a,b) - i.$$

As $g_s(a,b)$ is independent of t we can transform the equation as follows

$$\begin{aligned}
w(a,b) &= \sum_{\substack{s\in P^-(a,b)\\ g_s(a,b)>0}} \sum_{i=0}^{g_s(a,b)-1} \Delta_i(s,b)\cdot\big(g_s(a,b)-i\big)\\
&= \sum_{\substack{s\in P^-(a,b)\\ g_s(a,b)>0}} \Big(g_s(a,b)\sum_{i=0}^{g_s(a,b)-1}\Delta_i(s,b) - \sum_{i=0}^{g_s(a,b)-1}\big(i\cdot\Delta_i(s,b)\big)\Big)\\
&= \sum_{\substack{s\in P^-(a,b)\\ g_s(a,b)>0}} \Big(g_s(a,b)\cdot C_{g_s(a,b)-1}(s,b) - D_{g_s(a,b)-1}(s,b)\Big).
\end{aligned}$$

This finishes the proof. □

Lemma 7 is the key to obtain our $O(|V|^3)$-algorithm for performing one GREEDY-step, which is stated as Algorithm 3: First, all distances and hop-distances are precomputed. We then consider, for each $s \in V$, each shortest-paths subgraph with root s separately. It is easy to see that the values of $\Delta_\cdot(s,\cdot)$, $C_\cdot(s,\cdot)$ and $D_\cdot(s,\cdot)$ can be computed in time $O(|V|^2)$.

Algorithm 3: GREEDY STEP

Input: Strongly connected graph $G = (V, E, \text{len})$
Output: shortcut (a,b) maximizing $w_G(\{(a,b)\})$

compute $\text{dist}(\cdot,\cdot)$, $h(\cdot,\cdot)$
initialize $w(\cdot,\cdot) \equiv 0$
initialize $\Delta_i(\cdot,\cdot) \equiv 0$
for $s \in V$ **do**
  for $b,t \in V$ **do** `/* compute` Δ `*/`
    if *there exists a shortest s-t-path containing b in G* **then**
      $j \leftarrow \omega_{sb}(t)$
      $\Delta_j(s,b) \leftarrow \Delta_j(s,b) + 1$
  for $b \in V$ **do** `/* compute` C `and` D `*/`
    $C_0(s,b) \leftarrow \Delta_0(s,b)$
    $D_0(s,b) \leftarrow 0$
    for $r := 1$ **to** $|V|-1$ **do**
      $C_r(s,b) \leftarrow C_{r-1}(s,b) + \Delta_r(s,b)$
      $D_r(s,b) \leftarrow D_{r-1}(s,b) + r\cdot\Delta_r(s,b)$
  for $a,b \in V$ **do** `/* apply Lemma 7 */`
    if *there exists a shortest s-b-path containing a* **and** $g_s(a,b) > 0$ **then**
      $w(a,b) \leftarrow w(a,b) + g_s(a,b)\cdot C_{g_s(a,b)-1}(s,b) - D_{g_s(a,b)-1}(s,b)$
output arbitrary (a,b) with maximum $w(a,b)$

Prepared with these values we are ready to apply Lemma 7. We initialize the values $w(\cdot,\cdot)$ with 0. For each triple $s,a,b \in V$, we check if there is a shortest s-a-b-path and if $g_s(a,b) > 0$. We increment $w(a,b)$ according to Lemma 7 in case of a positive answer. Finally, we take an arbitrary shortcut (a,b) that maximizes $w(a,b)$. The correctness of the algorithm directly follows from the definitions of $\Delta.(\cdot,\cdot)$, $C.(\cdot,\cdot)$ and $D.(\cdot,\cdot)$ and Lemma 7. To reach the runtime in $O(|V|^3)$ we answer the question if a shortest s-a-b path exists by checking if $\text{dist}(s,a)+\text{dist}(a,b) = \text{dist}(s,b)$.

6 Approximation via Partitioning

The second algorithm works for sp-unique graphs in which the degree of each vertex is bounded by a constant. Given an sp-unique graph $G = (V,E,len)$ in which the degree of each vertex is bounded by a constant B. Algorithm 4 partitions V into small subsets, solves the SHORTCUT PROBLEM restricted to each subset and then chooses the best solution among all subsets as an approximated solution. If the subsets are small enough, we can solve the SHORTCUT PROBLEM restricted to each set in polynomial time.

In detail, our scheme works as follows. First, we partition the set V into sets $\mathscr{P} = \{P_1,\ldots,P_k\}$, where each P_i has size $size = \sqrt[c]{|V|^\varepsilon}/B$ for an arbitrary constant $\varepsilon \in (0,c)$. Then, for each set $P_i \in \mathscr{P}$, we compute the neighborhood $C_i := P_i \cup \{u \in N(v) \mid v \in P_i\}$ of P_i and solve the shortcut problem on G restricted to shortcuts in C_i. That is, we compute

$$\widetilde{S}_i = \text{argmax}\{w(S) \mid S \text{ is shortcut assignment } \subseteq C_i \times C_i \text{ and } |S| \le c\}.$$

Finally, we determine the set C_i, for which the shortcut assignment yields the highest gain. This solution gives an approximation ratio of $O\left(\max\left\{|V|^{1-\frac{\varepsilon}{c}}, \frac{1}{c}\cdot|V|^{1+\frac{\varepsilon}{c}}\right\}\right)$ to the optimal solution (see Theorem 1).

Algorithm 4: PARTITION

input : graph $G = (V,E,len)$, number of shortcuts c, parameter $\varepsilon \in (0,c)$
output: shortcut assignment S'

1 Partition the set V into sets $\mathscr{P} = \{P_1,\ldots,P_k\}$ each of size $size = \sqrt[c]{|V|^\varepsilon}/B$.
2 **for** $P_i \in \mathscr{P}$ **do**
3 $\quad C_i := P_i \cup \{u \in N(v) \mid v \in P_i\}$
4 $\quad \widetilde{S}_i := \text{argmax}\{w_G(S) \mid S \subseteq C_i \times C_i \text{ and } |S| \le c\}$
5 output $S' := \text{argmax}\{w_G(\widetilde{S}_i) \mid i = 1,2,\ldots,k\}$

Since $size = \sqrt[c]{|V|^\varepsilon}/B$ and G has bounded degree B, $|C_i| \le \sqrt[c]{|V|^\varepsilon}$ holds. Hence, each solution $\widetilde{S}_i$ can be computed by performing at most $(\sqrt[c]{|V|^\varepsilon})^{2c} = |V|^{2\varepsilon}$ all pairs shortest paths computations in G. As there are $\lceil |V|/size \rceil = \lceil |V|B/\sqrt[c]{|V|^\varepsilon} \rceil$ sets and $B = O(1)$, the overall computation time is $O(f(|V|)\cdot|V|^{2\varepsilon}\cdot|V|/\sqrt[c]{|V|^\varepsilon})$, where $f(|V|)$ is the time needed for computing all pairs shortest paths in G.

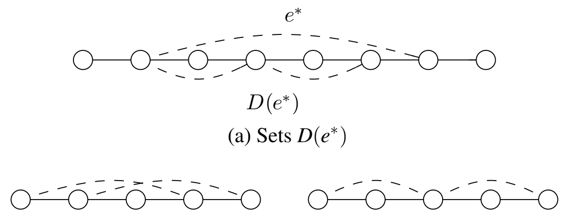


Figure 4: Illustrations to proof of Theorem 1

The following theorem shows the approximation ratio for PARTITION. A detailed discussion on the ratio is given after the proof.

Theorem 1. Given a weighted, directed, SP-unique graph $G = (V, E, len)$ in which the degree of each vertex is bounded by a constant, and a positive integer $c \in \mathbb{N}$. Then, the solution computed by PARTITION is an $O\left(\max\left\{|V|^{1-\frac{\varepsilon}{c}}, \frac{1}{c} \cdot |V|^{1+\frac{\varepsilon}{c}}\right\}\right)$ approximation for the optimal solution of the SHORTCUT PROBLEM instance (G, c).

Proof: The proof is outlined as follows: We break up the shortcuts in an optimal shortcut assignment into shortcuts whose endpoints are contained in the same neighborhood. We then show that at least one of the neighborhoods contains a subset of these shortcuts that fulfills the approximation guarantee stated in the theorem.

Let E^* denote an optimal solution to (G, c), $\mathscr{P}$ the partition and $\mathscr{C}$ the set of neighborhoods C_i used by PARTITION. Now each shortcut $e^* = (a, b) \in E^*$ is subdivided as follows: Let $p = (v_1, \ldots, v_r)$ be the unique shortest path from a to b in G and $D(e^*)$ be the set of shortcuts containing (v_i, v_{i+2}) for all odd i with $1 \leq i \leq i-2$, see Figure 4a. It is easy to see that for each of these shortcuts there is at least one neighborhood containing both endpoints, as these are connected by a path of length 2 in the original graph. Let $E' = \bigcup_{e^* \in E^*} D(e^*)$ and E_i be the set of $e = \{a, b\} \in E'$ such that both a and b are contained in C_i. Due to the construction of the E_i, this is a cover of E', i. e. $\bigcup_{i=1}^{\lceil |V|/\text{size} \rceil} E_i = E'$.

Claim. $w_G(E^*) \leq 2 \cdot \sum_{i=1}^{\lceil |V|/\text{size} \rceil} w_G(E_i)$

Let s, t be a pair of vertices such that $h_G(s,t) > h_{G[E^*]}(s,t)$ and $E^*(s,t)$ be the set of shortcuts on an arbitrary hop-minimal shortest s-t-path in $G[E^*]$. As shortest paths in G are unique, $h_G(s,t) - h_{G[E^*]}(s,t)$ equals the sum of the hop-distances between the endpoints of the shortcuts in $E^*(s,t)$ minus $|E^*(s,t)|$. Furthermore, for each e^* in E^*, $h_G(e^*) - 1 \leq 2 \cdot |D(e^*)|$ and thus,

$$h_G(s,t) - h_{G[E^*]}(s,t) = \sum_{e^* \in E^*(s,t)} (h_G(e^*) - 1) \leq 2 \cdot \sum_{e^* \in E^*(s,t)} |D(e^*)|$$

For $e^* \in E^*(s,t)$, the sets $D(e^*)$ are disjoint, as the respective shortcuts e^* lie on different parts of the chosen s-t-path. Therefore,

$$\begin{aligned}2 \cdot \sum_{e^* \in E^*(s,t)} |D(e^*)| &= 2 \cdot |\{e \in E' \mid \exists e^* \in E^*(s,t) \text{ with } e \in D(e^*)\}| \\ &\leq 2 \cdot \sum_{i=1}^{\lceil |V|/\text{size} \rceil} |\{e \in E_i \mid \exists e^* \in E^*(s,t) \text{ with } e \in D(e^*)\}|\end{aligned}$$

We say that two shortcuts (v_{i_1}, v_{i_2}) and (v_{j_1}, v_{j_2}) *overlap* on a path $v_1, \ldots, v_r$ if neither $i_2 \leq j_1$ nor $j_2 \leq i_1$, see Figure 4b. The shortcuts in E_i that are the result of dividing non-overlapping shortcuts on the shortest s-t-path are pairwise non-overlapping, thus

$$2 \cdot \sum_{i=1}^{\lceil |V|/\text{size} \rceil} |\{e \in E_i \mid \exists e^* \in E^*(s,t) \text{ with } e \in D(e^*)\}| \leq 2 \cdot \sum_{i=1}^{\lceil |V|/\text{size} \rceil} h_G(s,t) - h_{G[E_i]}(s,t)$$

Hence, it is

$$\begin{aligned}w_G(E^*) &= \sum_{s,t \in V} h_G(s,t) - h_{G[E^*]}(s,t) \\ &\leq 2 \cdot \sum_{s,t \in V} \sum_{i=1}^{\lceil |V|/\text{size} \rceil} h_G(s,t) - h_{G[E_i]}(s,t) = 2 \cdot \sum_{i=1}^{\lceil |V|/\text{size} \rceil} w_G(E_i)\end{aligned}$$

Let B be the maximum degree of a node in G and S' be the solution computed by PARTITION. For each i, two cases may occur:

- if $|E_i| \leq c$, since $S' = \text{argmax}\{w_G(\widetilde{S_i}) \mid i = 1, 2, \ldots, \lceil |V|/size \rceil\}$, then $w_G(E_i) \leq w_G(S')$.
- If $|E_i| > c$, then we can group the shortcuts in S_i into sets of size c. Since $S' = \text{argmax}\{w_G(\widetilde{S_i}) \mid i = 1, 2, \ldots, \lceil |V|/size \rceil\}$, each set of shortcuts of size c gives a decrease in overall hop length on shortest paths that is smaller than $w_G(S')$ and hence $w_G(E_i) \leq w_G(S') \frac{|E_i|}{c} \leq w_G(S') \frac{size^2 B^2}{c}$.

It follows that,

$$w(S^*) \leq 2 \sum_{i=1}^{\lceil |V|/size \rceil} w(S') \max\left\{1, \frac{size^2 \cdot B^2}{c}\right\} \leq 2 \left\lceil \frac{|V|}{size} \right\rceil w(S') \max\left\{1, \frac{size^2 \cdot B^2}{c}\right\}$$

Hence, the approximation ratio can be bound as follows.

$$\begin{aligned}\frac{w(S^*)}{w(S')} &\leq 2 \left\lceil \frac{|V|}{size} \right\rceil \max\left\{1, \frac{size^2 \cdot B^2}{c}\right\} = O\left(\frac{|V|}{size} \max\left\{1, \frac{size^2}{c}\right\}\right) \\ &= O\left(\frac{|V|}{\sqrt[c]{|V|^{\varepsilon}}} \max\left\{1, \frac{\sqrt[c]{|V|^{2\varepsilon}}}{c}\right\}\right) = O\left(\max\left\{|V|^{1-\frac{\varepsilon}{c}}, \frac{1}{c} \cdot |V|^{1+\frac{\varepsilon}{c}}\right\}\right) .\end{aligned}$$

□

c	GREEDY	PARTITION	Trivial bound
$O(1)$	$O(1)$	$\Omega(\|V\|)$	$\Theta(\|V\|^3)$
$\Theta(\log\|V\|)$	$\Theta(\log\|V\|)$	$\Theta(\|V\|)$	$\Theta\left(\frac{\|V\|^3}{\log\|V\|}\right)$
$\Theta(\|V\|)$	$\Theta(\|V\|)$	$\Theta(\|V\|)$	$\Theta(\|V\|^2)$
$\Theta(\|V\|\cdot\log\|V\|)$	$\Theta(\|V\|\cdot\log\|V\|)$	$\Theta(\|V\|)$	$\Theta(\frac{\|V\|^2}{\log\|V\|})$

Table 3: Comparison of approximation bounds for sample configurations

Discussion of Approximation Bounds. In this section, we compare PARTITION and GREEDY against each other and against trivial guarantees on some example configurations. Let $G = (V, E, len)$ be a weighted, directed, SP-unique graph in which the degree of each vertex is bounded by a constant. The latter assumption is not necessary for the statements concerning the greedy algorithm, its approximation guarantee of c holds for arbitrary degree distributions. We distinguish examplarily four settings for the parameter c:

- $c \in O(1)$: In this case, as stated in Section 3, the problem is polynomially solvable by a brute-force algorithm. However, the runtime of this approach is exponential in c, while the runtime of the greedy algorithm has only a linear dependence on c and its approximation guarantee is constant. The guarantee given by the partitioning approach is in $\omega(|V|)$.
- $c \in \Theta(\log(|V|))$: Let $x > 0$ be a constant, then $|V|^{1-\frac{\varepsilon}{x\cdot\log(|V|)}} = |V| \cdot 2^{-\frac{\varepsilon}{x}}$ and $\frac{1}{\log(|V|)} \cdot |V|^{1+\frac{\varepsilon}{x\cdot\log|V|}} = \frac{1}{\log(|V|)} \cdot |V| \cdot 2^{\frac{\varepsilon}{x}}$. Hence, the guarantee given for PARTITION is in $\Theta(|V|)$, which is worse than the bound of the greedy algorithm.
- $c \in \Theta(|V|)$: It is $|V| \geq |V|^{1-\frac{\varepsilon}{x\cdot|V|}} \geq |V|^{1-\frac{\varepsilon}{x\cdot\log(|V|)}}$. Thus, analoguous to the last case, the approximation guarantee of PARTITION is in $\Theta(|V|)$, which matches the bound given for GREEDY.
- $c \in \Theta(|V| \cdot \log(|V|))$: The guarantee of the partitioning algorithm stays in $\Theta(|V|)$, which is better than the guarantee of the greedy algorithm. Note that this guarantee is much tighter than the trivial bound of $\frac{|V|^2}{\log(|V|)}$ given in Section 3.

A summary of these configurations is given in Table 3.

7 Evaluation of the Measure Function

To evaluate the gain of a given shortcut assignment, a straightforward algorithm requires computing all-pairs shortest-paths. Since this computation is expensive, we provide a probabilistic method to quickly assess the quality of a shortcut assignment E'. This approach can be used for networks where the computation of all-pairs shortest-paths is prohibitive, such as big road networks. For the sake of simplicity we state the approach for the evaluation of $\mu(E') := \sum_{s,t \in V} h_{G[E']}(s,t)$, the adaption to the SHORTCUT PROBLEM is straightforward. More concrete, we apply the sampling technique to obtain an unbiased estimate for $\mu(E')$ and apply *Hoeffding's Bound* [18] to get a confidence intervall for the outcome. As an auxiliary result we propose algorithms that approximate the maximum hop-distance in a graph.

Theorem 6 (Hoeffding's Bound). If $X_1, X_2, \ldots, X_K$ are real valued independent random variables with $a_i \le X_i \le b_i$ and expected mean $\mu = \mathbb{E}[\sum X_i / K]$, then

$$\mathbb{P}\left\{\left|\frac{\sum_{i=1}^{K} X_i}{K} - \mu\right| \ge \xi\right\} \le 2e^{-2K^2\xi^2/\sum_{i=1}^{K}(b_i - a_i)^2}$$

for each $\xi > 0$.

We now model the assessment of a shortcut assignment E' of a graph G in terms of Hoeffding's Bound. Let $X_1, \ldots, X_K$ be the family of random variables such that X_i is defined by

$$X_i := |V| \sum_{t \in V} h_{G[E']}(s_i, t)$$

where $s_i \in V$ is a vertex chosen uniformly at random. We estimate $\mu(E')$ by

$$\hat{\mu} := \sum_{i=1}^{K} X_i / K \ .$$

Because of

$$\mathbb{E}(\hat{\mu}) = \mathbb{E}\left(\sum_{i=1}^{K} \frac{X_i}{K}\right) = \sum_{i=1}^{K} \frac{\mathbb{E}(X_i)}{K} = \mathbb{E}(X_1) = \frac{1}{|V|} \sum_{s \in V} |V| \sum_{t \in V} h_{G[E']}(s,t) = \mu(E')$$

we can apply Hoeffding's Bound if we know lower and upper bounds for the variables X_i. The values 0 and $|V|^3$ are trivial such bounds. We introduce the notion of *shortest-paths diameter* to obtain stronger upper bounds.

Definition. The *shortest path diameter* $\mathrm{spDiam}(G)$ of a graph G is the maximum hop-distance from any node to any other node in G.

Applying Hoeffding's Bound with $0 \le X_i \le |V|^2 \, \mathrm{spDiam}(G)$ yields

$$\mathbb{P}\left\{\left|\hat{\mu} - \mu(E')\right| \ge \xi\right\} \le 2e^{-2K\xi^2/(|V|^4 \cdot \mathrm{spDiam}(G)^2)}$$

and with $l_{rel} := \xi / \hat{\mu}$ we have

$$\mathbb{P}\left\{\left|\frac{\hat{\mu} - \mu(E')}{\hat{\mu}}\right| \ge l_{rel}\right\} \le 2e^{-2K(\hat{\mu} \cdot l_{rel})^2/(|V|^4 \cdot \mathrm{spDiam}(G)^2)}$$

where the parameter l_{rel} states the relative size of the confidence intervall. The values of the variables X_i are chosen by randomly choosing values from the *finite population* $c_1, \ldots, c_{|V|}$ *with replacement* where $c_i := |V| \sum_{t \in V} h_{G[E']}(v_i, t)$ and $V = \{v_1, \ldots v_{|V|}\}$. In [18] it is reported that Hoeffding's Bound stays correct if, when sampling from a finite population, the samples are being chosen without replacement. Algorithm 5 is an approximation algorithm that exploits the above inequality and that samples without replacement.

Algorithm 5: STOCHASTICALLY ASSESS SHORTCUT ASSIGNMENT

```
input : graph G = (V, E ∪ E′, len),
        size of confidence intervall l_rel, significance level α
output: approximation μ̂ for μ = ∑_{s,t∈V} h_G(s,t)
compute random order v_1, v_2, ..., v_n of V
compute upper bound spDiam‾(G) for shortest-paths diameter
i ← 0; sum ← 0; μ̂ ← 0
while not (i = |V| or 2·exp(−2i(μ̂·l_rel)^2/(|V|^4 spDiam‾(G)^2)) ≤ α) do
    i ← i+1
    T ← grow shortest-paths tree rooted at v_i (favor edge-minimal shortest
    paths)
    sum ← sum + |V| · ∑_{t∈V} h′_G(v_i, t)
    μ̂ ← sum/i
output μ̂
```

Approximating the Shortest-Paths Diameter. A straightforward approach to compute the exact shortest path diameter requires computing all-pairs shortest-paths. This is reasonable when working with mid-size graphs that allow the computation of all-pairs shortest-paths at least once and for which a large number of shortcut assignments is to be evaluated. In case the computation of all-pairs shortest-paths is prohibitive one can also use upper bounds for the shortest path diameter. We obtain an upper bound the following way:

First we compute an upper bound $\overline{\mathrm{diam}}(G)$ for the diameter of G. To do so we choose a set of nodes $s_1, s_2, \ldots, s_l$ uniformly at random. We denote the *eccentricity* of node v in graph $G = (V, E, \mathrm{len})$ by $\varepsilon_G(v) = \max\{\mathrm{dist}_G(v,t) \mid t \in V\}$. For each node s_i, the value $\varepsilon_{\overleftarrow{G}}(s_i) + \varepsilon_G(s_i)$ is an upper bound for the diameter of G: Let $u, v \in V$ be such that $\mathrm{dist}(u,v) = \mathrm{diam}(G)$. Then

$$\mathrm{diam}(G) = \mathrm{dist}(u,v) \leq \mathrm{dist}(u,s_i) + \mathrm{dist}(s_i,v) \leq \varepsilon_{\overleftarrow{G}}(s_i) + \varepsilon_G(s_i) \ .$$

We set $\overline{\mathrm{diam}}(G)$ to be the minimum of these values over all s_i. The bound $\overline{\mathrm{diam}}(G)$ is a 2-approximation for the exact diameter $\mathrm{diam}(G)$ of G (already for $l = 1$) as there are $u, v \in V$ and $s_i \in V$ such that

$$\overline{\mathrm{diam}}(G) = \mathrm{dist}(u,s_i) + \mathrm{dist}(s_i,v) \leq \mathrm{diam}(G) + \mathrm{diam}(G) = 2 \cdot \mathrm{diam}(G).$$

Let $\mathrm{len}_{\max}$ and $\mathrm{len}_{\min}$ denote the lengths of a longest and a shortest edge in G, respectively. The value $\overline{\mathrm{diam}}(G)/\mathrm{len}_{\min}$ is an upper bound for $\mathrm{spDiam}(G)$: Let P be an edge-minimal shortest path in G with $|P| = \mathrm{spDiam}(G)$ edges. Then

$$\mathrm{spDiam}(G) = |P| \leq \frac{\mathrm{len}(P)}{\mathrm{len}_{\min}} \leq \frac{\mathrm{diam}(G)}{\mathrm{len}_{\min}} \leq \frac{\overline{\mathrm{diam}}(G)}{\mathrm{len}_{\min}} \ .$$

Further, $\overline{\mathrm{diam}}(G)/\mathrm{len}_{\min}$ is a $2 \cdot \mathrm{len}_{\max}/\mathrm{len}_{\min}$-approximation for $\mathrm{spDiam}(G)$ as with

$\mathrm{spDiam}(G) \geq \mathrm{diam}(G)/\,\mathrm{len}_{\max}$ follows that

$$\frac{\overline{\mathrm{diam}}(G)}{\mathrm{len}_{\min}} \leq \frac{2\,\mathrm{diam}(G)}{\mathrm{len}_{\min}} \leq \frac{2\,\mathrm{len}_{\max} \cdot \mathrm{spDiam}(G)}{\mathrm{len}_{\min}} \ .$$

A more expensive approach works as follows, pseudocode is given as Algorithm 6: After computing $\overline{\mathrm{diam}}(G)$, we choose a tuning parameter η. Then we grow, for each node s in G, a shortest-paths tree whose construction is stopped directly before one vertex with distance greater than $\overline{\mathrm{diam}}(G)/\eta$ is settled. When breaking ties between different shortest paths we favor edge-minimal shortest paths. We denote by $\tau_{\max}$ the maximum number of edges of the shortest paths on any of the trees grown *plus one*. Then $\overline{\mathrm{spDiam}}(G) := \tau_{\max} \cdot \eta$ is an upper bound for the shortest path diameter of G: Let $P = (v_1, \ldots, v_n)$ be an arbitrary edge-minimal shortest path in G. We can split P in sub-paths

$$P_1 = (v_1, \ldots, v_{k_1}),\ P_2 = (v_{k_1}, \ldots, v_{k_2}),\ \ \ldots\ \ ,\ P_\ell = (v_{k_{\ell-1}}, \ldots, v_{k_\ell})$$

such that

$$\mathrm{dist}(v_{k_i}, v_{k_{i+1}}) > \overline{\mathrm{diam}}(G)/\eta \ \text{ and } \ \mathrm{dist}(v_{k_i}, v_{k_{i+1}-1}) \leq \overline{\mathrm{diam}}(G)/\eta \ .$$

The number ℓ of these subpaths is at most η, as $\ell > \eta$ would imply that

$$\mathrm{len}(P) > \frac{\overline{\mathrm{diam}}(G)}{\eta}(\ell - 1) \geq \overline{\mathrm{diam}}(G).$$

It is $|P_i| \leq \tau_{\max}$ which yields $|P| \leq \tau_{\max} \cdot \eta$. As P was arbitrary we have that

$$\mathrm{spDiam}(G) \leq \tau_{\max} \cdot \eta \ .$$

Further $\tau_{\max} \cdot \eta$ is a 2η-approximation and an $\eta(1 + 1/(\tau_{\max} - 1))$-approximation of $\mathrm{spDiam}(G)$: With $\tau_{\max} - 1 \leq \mathrm{spDiam}(G)$ follows that

$$\frac{\tau_{\max} \cdot \eta}{\mathrm{spDiam}(G)} \leq \frac{(\mathrm{spDiam}(G)+1)\eta}{\mathrm{spDiam}(G)} = \eta(1 + \frac{1}{\mathrm{spDiam}(G)}) \leq \eta(1 + \frac{1}{\tau_{\max} - 1}) \leq 2 \cdot \eta \ .$$

Obviously, the whole proceeding only makes sense for graphs for which the shortest path diameter is much smaller than the number of nodes. This holds for a wide range of real-world graphs, in particular for road networks. For example, the road network of Luxembourg provided by the PTV AG [22] consists of 30733 nodes and has a shortest path diameter of only 429. The road network of the Netherlands consists of 946.632 nodes and has a shortest-paths diameter of 1503.

8 Conclusion

Summary. In this work we studied two problems. The SHORTCUT PROBLEM is the problem of how to insert c shortcuts in G such that the expected number of edges that

Algorithm 6: COMPUTE UPPER BOUND FOR SHORTEST-PATHS DIAMETER

input : graph $G = (V, E, \text{len})$, tuning parameter l, tuning parameter η
output: upper bound $\overline{\text{spDiam}}(G)$ for the shortest-paths diameter of G

$\overline{\text{diam}}(G) \leftarrow \infty$; $\tau \leftarrow 0$;

for $i = 1, \ldots, l$ **do** `/* compute` $\overline{\text{diam}}(G)$ `*/`
  $s \leftarrow$ choose node uniformly at random
  grow shortest-paths tree rooted at s
  grow shortest-paths tree rooted at s on the reverse graph $\overleftarrow{G}$
  $\overline{\text{diam}}(G) \leftarrow \min\{\overline{\text{diam}}(G), \max_{v \in V}\{\text{dist}(s,v)\} + \max_{v \in V}\{\text{dist}(v,s)\}\}$

for $s \in V$ **do** `/* compute` $\overline{\text{spDiam}}(G)$ `*/`
  $T \leftarrow$ grow partial shortest-paths tree rooted at s (favoring edge-minimal shortest paths). Stop growing the tree directly before the first node with $\text{dist}(s,v) > \overline{\text{diam}}(G)/\eta$ is settled.
  $\tau_{\max} \leftarrow \max\{\tau_{\max}, 1 + \text{maximal number of edges of a path in } T\}$

output $\overline{\text{spDiam}}(G) := \tau_{\max} \cdot \eta$

are contained in an edge-minimal shortest path from a random node s to a random node t is minimal. The REVERSE SHORTCUT PROBLEM is the variant of the SHORTCUT PROBLEM where one has to insert a minimal number of shortcuts to reach a desired expected number of edges on edge-minimal shortest paths.

We proved that both problems are NP-hard and that there is no polynomial-time constant-factor approximation algorithm for the REVERSE SHORTCUT PROBLEM, unless $P = NP$. Furthermore, no polynomial-time algorithm exists that approximates the SHORTCUT PROBLEM up to an additive constant unless $P = NP$.

The algorithmic contribution focused on the SHORTCUT PROBLEM. We proposed two ILP-based approaches to exactly solve the SHORTCUT PROBLEM: A straightforward formulation that incorporates $O(|V|^4)$ variables and constraints and a more sophisticated flow-like formulation that requires $O(|V|^3)$ variables and constraints.

We considered two approximation strategies. A straightforward greedy approach computes a c-approximation of the optimal solution if the input graph is such that shortest paths are unique. We further presented a dynamic program that performs a greedy step in time $O(|V|^3)$ which yields an overall runtime in $O(c \cdot |V|^3)$. The main idea of the second approach is to partition the set of nodes. It computes an $O\left(\max\left\{|V|^{1-\frac{\varepsilon}{c}}, \frac{1}{c} \cdot |V|^{1+\frac{\varepsilon}{c}}\right\}\right)$ approximation of the optimal solution if shortest paths in the input graph are unique and the maximum degree is bounded by a constant. If ε is a constant, this algorithm is polynomial.

Finally, we proposed a probabilistic method to quickly evaluate the measure function of the SHORTCUT PROBLEM. This can be used for large input networks where an exact evaluation is prohibitive.

Future Work. A wide range of possible future work exists for the SHORTCUT PROBLEM. From a theoretical point of view the probably most interesting open field is the approximability of the SHORTCUT PROBLEM. It is still unknown if it is in APX. Furthermore, it would be helpful to identify graph classes for which the SHORTCUT PROBLEM or the REVERSE SHORTCUT PROBLEM becomes tractable. FPT-algorithms are also desirable. From an experimental point of view it would be interesting to develop heuristics that find good shortcuts for large real-world inputs. In particular, evolutionary algorithms and local search algorithms seem promising.

Finally, we pose the question if the given ILP-approaches can be used for the design of approximation algorithms. We do not see good chances for rounding-based methods. However, other techniques like primal-dual arguments might work.

References

[1] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In M. Charikar, editor, *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 782–793. SIAM, 2010.

[2] *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX'06)*. SIAM, 2006.

[3] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.

[4] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, and A. Marchetti-Spaccamela. *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 2nd edition, 2002.

[5] 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)*, volume 6078 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 2010.

[6] 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.

[7] R. Bauer and D. Delling. SHARC: Fast and Robust Unidirectional Routing. In I. Munro and D. Wagner, editors, *Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX'08)*, pages 13–26. SIAM, April 2008.

[8] R. Bauer and D. Delling. SHARC: Fast and Robust Unidirectional Routing. *ACM Journal of Experimental Algorithmics*, 14(2.4):1–29, August 2009. Special Section on Selected Papers from ALENEX 2008.

[9] F. Bruera, S. Cicerone, G. D'Angelo, G. D. Stefano, and D. Frigioni. Dynamic Multi-level Overlay Graphs for Shortest Paths. *Mathematics in Computer Science*, 1(4):709–736, April 2008.

[10] E. Brunel, D. Delling, A. Gemsa, and D. Wagner. Space-Efficient SHARC-Routing. In P. Festa, editor, *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, volume 6049 of *Lecture Notes in Computer Science*, pages 47–58. Springer, May 2010.

[11] D. Delling. *Engineering and Augmenting Route Planning Algorithms*. PhD thesis, Universität Karlsruhe (TH), Fakultät für Informatik, 2009.

[12] C. Demetrescu, editor. *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*. Springer, June 2007.

[13] 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.

[14] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In ALENEX'06 [2], pages 129–143.

[15] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Better Landmarks Within Reach. In Demetrescu [12], pages 38–51.

[16] A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A*: Shortest Path Algorithms with Preprocessing. 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 93–139. American Mathematical Society, 2009.

[17] 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.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):713–721, 1963.

[19] M. Holzer. *Engineering Planar-Separator and Shortest-Path Algorithms*. PhD thesis, Karlsruhe Institute of Technology (KIT) - Department of Informatics, 2008.

[20] M. Holzer, F. Schulz, and D. Wagner. Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. In ALENEX'06 [2], pages 156–170.

[21] M. Holzer, F. Schulz, and D. Wagner. Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. *ACM Journal of Experimental Algorithmics*, 13(2.5):1–26, December 2008.

[22] PTV AG - Planung Transport Verkehr. `http://www.ptv.de`, 2008.

[23] 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.

[24] 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.

[25] P. Sanders and D. Schultes. Engineering Fast Route Planning Algorithms. In Demetrescu [12], pages 23–36.

[26] D. Schultes and P. Sanders. Dynamic Highway-Node Routing. In Demetrescu [12], pages 66–79.

[27] F. Schulz. *Timetable Information and Shortest Paths*. PhD thesis, Universität Karlsruhe (TH), Fakultät für Informatik, 2005.

[28] F. Schulz, D. Wagner, and K. Weihe. Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. *ACM Journal of Experimental Algorithmics*, 5(12):1–23, 2000.

[29] F. Schulz, D. Wagner, and C. Zaroliagis. Using Multi-Level Graphs for Timetable Information in Railway Systems. In *Proceedings of the 4th Workshop on Algorithm Engineering and Experiments (ALENEX'02)*, volume 2409 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2002.

[30] A. Schumm. Heuristic Algorithms for the Shortcut Problem. Master's thesis, Karlsruhe Institute of Technology (KIT), July 2009.

[31] D. Wagner and T. Willhalm. Speed-Up Techniques for Shortest-Path Computations. In *Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 of *Lecture Notes in Computer Science*, pages 23–36. Springer, 2007.