Karlsruhe Reports in Informatics 2010,17
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

The Shortcut Problem – Complexity and Algorithms

Reinhard Bauer, Gianlorenzo D'Angelo, Daniel Delling,
Andrea Schumm and Dorothea Wagner

2010

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

The Shortcut Problem – Complexity and Algorithms*

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

[1] Faculty of Informatics, Karlsruhe Institute of Technology (KIT),
{Reinhard.Bauer, Andrea.Schumm, Dorothea.Wagner}@kit.edu
[2] Department of Electrical and Information Engineering, University of L'Aquila,
gianlorenzo.dangelo@univaq.it
[3] Microsoft Research, dadellin@microsoft.com

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 $c \in \mathbb{Z}^+$, the *shortcut problem* asks how to insert c shortcuts in 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. 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.

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 [15, 20] for an overview).

One core part of many of these *speedup techniques* is the insertion of *shortcuts* [6–10, 12, 14, 16–18], 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. Until now, all existing shortcut insertion strategies are heuristics and only few theoretical worst-case or average case results are known [1, 4].

In this context, an interesting new theoretical problem arises: Given a weighted, directed graph G and a number $c \in \mathbb{Z}^+$, 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. We study the algorithmic complexity of the problems and 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 little 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.

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

Related Work. Part of this work has previously been published in [5]. The diploma thesis [19] experimentally examines heuristic algorithms to find shortcut assignments with high quality, including local search strategies and a betweenness-based approach. The GREEDY-step algorithm introduced in Section 5 is taken from 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 multi-level overlay graph technique [7, 12, 16–18] uses some centrality measures or separation strategies to choose a set of important nodes V' in the graph and sets 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 [14] and reach based pruning [9, 10] 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 [6] and Contraction Hierarchies [8] 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. 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 Chapter 7. Our work 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 $len : E \to \mathbb{R}^+$. 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 $(x_i, x_{i+1}) \in E$, $i = 1, \ldots, k-1$. 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 *(rooted) tree with root (node) s* is a graph $T = (V', E')$ 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 *cycle* C is a path of the form $s = x_1, \ldots, x_k = s$ with $k \geq 2$.

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. Consider a path $(x_1, x_2, \ldots, x_k)$. 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 $v = x_j$. We abbreviate that by $u \leq_P v$. An x_1(-x_2)(-x_3)-x_4-path is a path from x_1 to x_4 such that $x_1 (\leq_P x_2)(\leq_P x_3) \leq_P x_4$. A shortest x_1(-x_2)(-x_3)-x_4-path is an x_1(-x_2)(-x_3)-x_4-path that is a shortest path from x_1 to x_4. 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 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.

Considered Graphs. A *simple graph* is a graph such that for any two nodes u and v, there is at most one edge (u,v) and such that no edge of the form (u,u) exists. Throughout this work, we only consider simple graphs. 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.

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, there exists 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 $len' : E \cup E' \to \mathbb{R}^+$ 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.

The SHORTCUT PROBLEM consists of adding a number c of shortcuts to a graph, such that the gain is maximal.

Definition (SHORTCUT PROBLEM). Let $G = (V, E, len)$ be a graph and $c \in \mathbb{Z}^+$ 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.

We shortly 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. Hence, maximizing the gain and minimizing this expected number of edges are equivalent problems.

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 $|\{(u, v) \in V \times V \mid \text{dist}(u, v) < \infty, u \neq v\}|$. 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 $\sum_{s,t \in V} h_G(s,t)$.

Definition (REVERSE SHORTCUT PROBLEM). Let $G = (V, E, len)$ be a graph and $k \in \mathbb{N}$ be a positive integer that is 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 will also consider the SHORTCUT DECISION PROBLEM.

Definition (SHORTCUT DECISION PROBLEM). Let $G = (V, E, len)$ be a graph and $c, k \in \mathbb{N}$ be positive integers. Given an instance (G, c, k), the SHORTCUT DECISION PROBLEM is to decide if there is a shortcut assignment E' for $G = (V, E, len)$ 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 $\sum_{c \in C} |c|$.

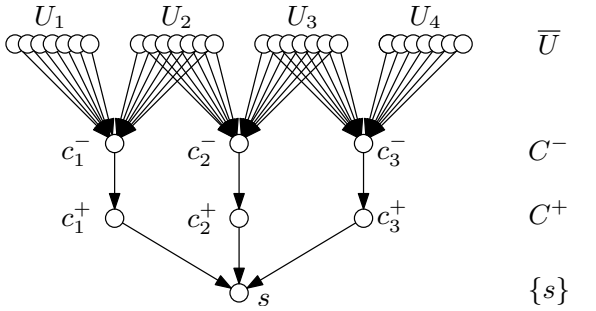


Fig. 1: Graph $G = (V,E)$ constructed from the SET COVER-instance $\{c_1 = \{1,2\}, c_2 = \{2,3\}, c_3 = \{3,4\}\}$.

Notation (Solution). Given an {SHORTCUT PROBLEM, REVERSE SHORTCUT PROBLEM, MIN SET COVER}-instance I, we denote by $\text{opt}_{\{\text{SP,RSP,MSC}\}}(I)$ an arbitrary (optimal) solution of I on 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. Then, there is a graph $G = (V,E,\text{len})$ 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 the graph $G = (V,E,\text{len})$ as follows: 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 $C^+ := \{c^+ | c \in C\}$. See Figure 1 for an illustration.

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 $w(E') \geq \Delta|U|$.

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 $|E'| > |C|$. In this 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 first 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-hard.

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 called NP-hard if for every decision problem $P' \in$ NP, the problem P' can be solved in polynomial time by an algorithm which uses an oracle that, for any instance of P, returns 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

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

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 [3]:

As described in Lemma 1, we first construct the graph G, then compute $E' = \text{apx}(G,(2|C|+1)|U|)$ and finally transform E' to a set cover instance C' of (C,U) of size at most $|E'|$. With Lemma 1 we have that $|\text{opt}_{\text{MSC}}(C,U)| = |\text{opt}_{\text{RSP}}(G,(2|C|+1)|U|)|$. Hence it is $|C'|/|\text{opt}_{\text{MSC}}| \leq |E'|/|\text{opt}_{\text{RSP}}(G,(2|C|+1)|U|)| \leq \alpha$ which shows the theorem. ■

Using a stronger result on the inapproximability of the MIN SET COVER-problem, we get a slightly tighter lower bound on the approximation factor of the REVERSE SHORTCUT PROBLEM. This is stated in the following proposition.

Proposition 1. Unless P = NP, there is a constant η such that there exists no polynomial time algorithm that approximates the REVERSE SHORTCUT PROBLEM to a factor $\eta \cdot \ln\big(W(|V|)-2\big)$, where W denotes the Lambert W function.

Proof. By [2], 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 $|\operatorname{apx}(G,k)|/|\operatorname{opt}_{\mathrm{RSP}}(G,k)| \leq \eta \cdot \ln\big(W(|V|)-2\big)$ 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 $|C'|/|\operatorname{opt}_{\mathrm{MSC}}(C,U)| \leq \eta \cdot \ln\big(W(|V|)-2\big)$. As $|C| \leq 2^{|U|}$, it is

$$|V| \leq (2^{|U|+1}+1)|U|+2^{|U|+1}+1 = 2^{|U|+1}(|U|+1)+|U|+1 \leq 2^{|U|+2}(|U|+1) \leq e^{|U|+2}(|U|+2)$$

Hence, it is $|U|+2 \geq W(|V|)$ and thus $|C'|/|\operatorname{opt}_{\mathrm{MSC}}(C,U)| \leq \eta \cdot \ln|U|$, which shows the proposition. ■

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

- $\operatorname{apx}(G,k)$ is a shortcut assignment for G of cardinality at most k
- the runtime of $\operatorname{apx}(G=(V,E,\mathrm{len}),k)$ is polynomial in the size of (G,k)
- $w_G(\operatorname{opt}_{\mathrm{SP}}(G,k)) - w_G(\operatorname{apx}(G,k)) \leq \alpha$ for all instances (G,k) 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 $(G=(V,E,len),c,k)$ be a SHORTCUT DECISION PROBLEM-instance. To assure $\alpha \in \mathbb{Z}^+$, we set $\alpha := \lceil\alpha\rceil$. We construct an instance $(\overline{G}=(\overline{V},\overline{E},\overline{len}),c)$ 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 $\overline{len}(v_i,v)=1$ 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 $(G=(V,E,len),c,k)$ in polynomial time. We can insert at most $c_{\max} := |\{(u,v) \in V\times V \setminus E \mid \operatorname{dist}(u,v) < \infty, u \neq v\}|$ 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* $(G=(V,E,len),c,k)$ *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. $\operatorname{opt}_{\mathrm{SP}}(\overline{G},c) \subseteq V\times V$. 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. ■

Trivial approximation bounds. Consider an arbitrary non-empty shortcut assignment S. It is $n(n-1) \leq \sum_{s \in V} \sum_{t \in V} h_G(s,t) \leq n^3$ for any graph $G = (V,E)$ and hence $w_G(S) \leq n^3 - n^2 + n$. As each shortcut in S decreases the hop-distance from its start to its end-node to 1 we have that S is a trivial factor $(n^3 - n^2 + n)/|S|$-approximation of the SHORTCUT PROBLEM.

Further, any shortcut assignment achieving the desired gain is a trivial factor n^2-approximation for the REVERSE SHORTCUT PROBLEM.

Bounded number of shortcuts. If the number of shortcuts c 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 a shortcut s, $s \notin S$, the following two inequalities do not hold in general:

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

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.

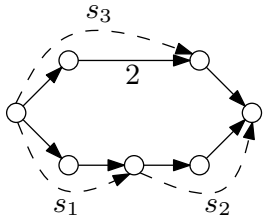


Fig. 2: Example Graph G with shortcuts s_1, s_2, s_3, all edges for which no weight is given in the picture have weight 1.

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 $S = \{s_1, s_2, \ldots, s_k\}$. Then, $w_G(S) \leq \sum_{i=1}^{k} w_G(s_i)$ and $w_G(S) \leq w_G(\{s_1, \ldots s_{k-1}\}) + w_G(s_k)$.

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

$$w^{ab}_{G[s_1,\dots,s_{k-1}]}(\{(x,y)\}) \leq h_{G[s_1,\dots,s_{k-1}]}(\{(x,y)\}) - 1 \leq h_G(\{(x,y)\}) - 1 = w^{ab}_G(s_k).$$

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

$$w^{ab}_G(S) \leq w^{ab}_G(\{s_1,\dots s_{k-1}\}) + w^{ab}_G(s_k)$$

which shows the lemma. ■

Later, we use these results to present approximation algorithms 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 $s \in V$, we denote by P_s the set of all vertices $u \in V$ for which an s-u-path exists. Remember that we denote by $P^+(s,u)$ the set of all vertices $v \in V$ for which a shortest s-v path containing u 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 all distances in the graph to be precomputed and hence the sets P_s, $P^{\bowtie}(x,y)$ and $P^+(s,u)$ to be known for all $s,u \in V$.

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

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

The variable $c(u,v)$ equals 1 if the applied shortcut assignment contains (u,v). Instead of maximizing the gain, our aim is to minimize the expected hop-distance in the augmented graph.

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

such that

$$\sum_{v \in P^{\bowtie}(s,t), v \neq t} k^s_t(v,t) = 1 \qquad s \in V; t \in P_s \tag{4}$$

$$\sum_{u \in P^{\bowtie}(s,v), u \neq v} k^s_t(u,v) = \sum_{w \in P^{\bowtie}(v,t), w \neq v} k^s_t(v,w) \qquad s \in V; t \in P_s; v \in P^{\bowtie}(s,t), v \neq s,t \tag{5}$$

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

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

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

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

Constraint 4 ensures that a shortest path for every s-t-pair is considered while Constraint 5 guarantees that s and t are connected by a path. The 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 $\{(u, v) \in V \times V \mid c(u, v) = 1\}$ 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 shortcut assignment found contains (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. More detailed, 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).

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.

$$\text{(LSP)} \quad \text{minimize} \quad \text{obj}(f, c) := \sum_{s \in V,\ u \in P_s,\ v \in P^+(s,u),\ u \neq v} f^s(u, v) \tag{10}$$

such that

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

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

$$f^s(u, v) \leq |P^+(s, v)| \cdot c(u, v) \qquad s \in V,\ u \in P_s,\ v \in P^+(s, u)$$

$$(u, v) \notin E,\ u \neq v \tag{13}$$

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

$$f^s(u, v) \in \mathbb{Z}_{\geq 0} \qquad s \in V,\ u \in P_s,\ v \in P^+(s, u),\ u \neq v \tag{15}$$

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

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 path 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 topological 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. Hence, 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 a solution E' of the SHORTCUT PROBLEM the variable assignment

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

can be extended to a solution of (LSP). Further, given a solution (f,c) of (LSP), it is

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

a solution for the SHORTCUT PROBLEM.

Proof. Given is a SHORTCUT PROBLEM-instance $(G = (V,E,\text{len}),c)$. When working on $(G = (V,E,\text{len}),c)$ 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 path 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 $(\text{edge length}, \text{hop distance})$ and applying lexicographical ordering. Let

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

and

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

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 $|P_T(s,t)|$ the number of edges on this path. It is

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

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

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, it is

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

a feasible solution of the SHORTCUT PROBLEM. Finally, we show that $\text{obj}(E'') \leq \text{obj}(f,c)$. 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

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

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 $(\cdot, c)$ can be extended to a solution 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. ■

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

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

An additional heuristic improvement works as follows: It is $\sum_{s,t \in V} h_G(s,t)$ a sharp lower bound for the objective function of model (LSP) in case no shortcuts are allowed. The value $h_G(a,b) \cdot |P(a,b)|$ 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 $h_G(a,b) \cdot |P(a,b)|$. Then we can add the constraint

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

to additionally tighten the model.

Experimental Results. 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 quick 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 graph classes. The graph *disk* is a unit disk graph and generated by randomly assigning 100 nodes to a point in the unit square of the Euclidean plain. Two nodes are connected by an edge in case their Euclidean distance is below a given radius. This radius is adjusted such that the resulting graph has approximately 1000 edges. The graph *ka* represents a part of the road network of Karlsruhe. It contains 102 nodes and 241 edges. The graph *grid* is based on a two-dimensional 10x10 square grid. The nodes of the graph correspond to the crossings in the grid. There is an edge between two nodes if these are neighbors on the grid. Finally, the graph *path* is a path consisting of 30 nodes. Edge weights are always 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 k-variables of the simple model and the f-variables 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 Equation 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 simple and flow formulation we obtain that the flow formulation is the winner. 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 clearly was computation time. However, to enable a vague comparison of the memory consumption, we report the number of nonzeros reported by CPLEX after the presolve routine in Table 2. Note that this number turned out to be almost independent from the number of shortcuts to be inserted.

5 Approximation using the GREEDY-Strategy

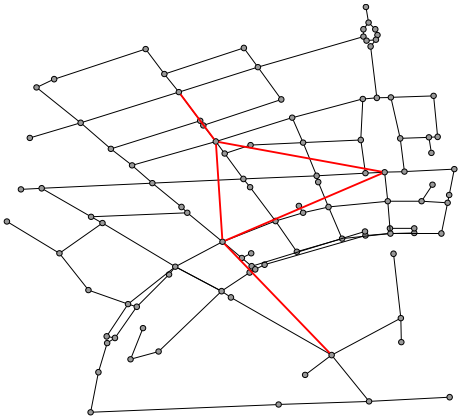
In this section, we propose a polynomial time algorithm that approximatively solves the SHORTCUT PROBLEM in a greedy fashion. Given the number c of shortcuts to insert, the approach finds a c-approximation of the optimal solution if the underlying graph is sp-unique. While the algorithm works on arbitrary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

shortcuts	model	Eq19	Eq18	grid			ka			path			disk		
				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

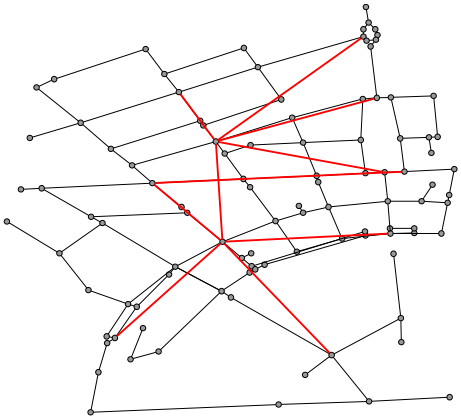
Table 1: Experimental results of the ILP-approaches.

model	Eq19	Eq18	grid	ka	path	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

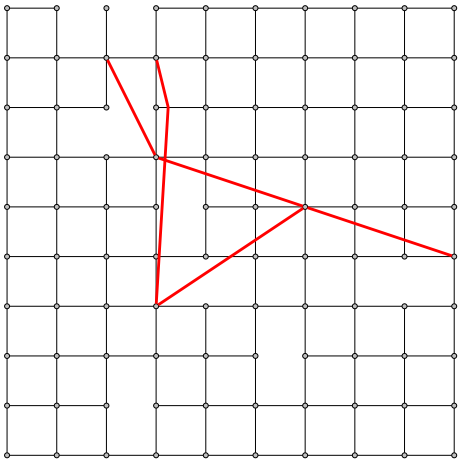
Table 2: Number of nonzeros reported by CPLEX after the presolve routine for each model and graph.



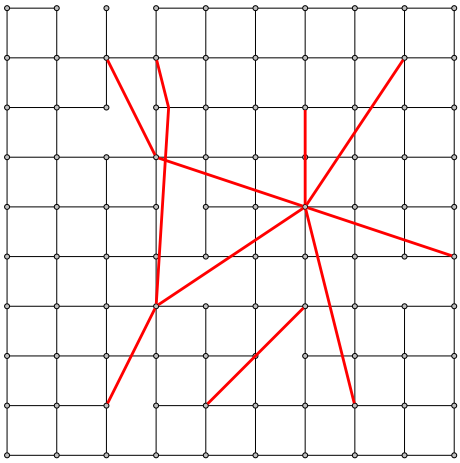
graph *ka* with 5 optimal shortcuts



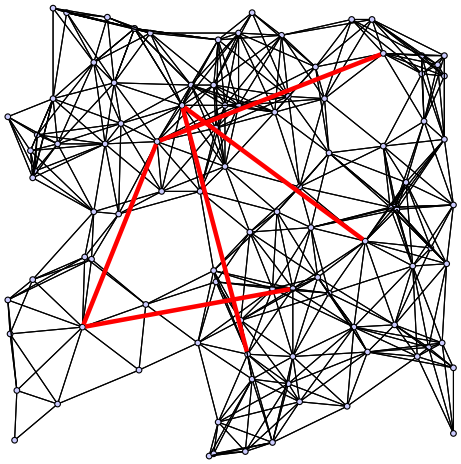
graph *ka* with 10 optimal shortcuts



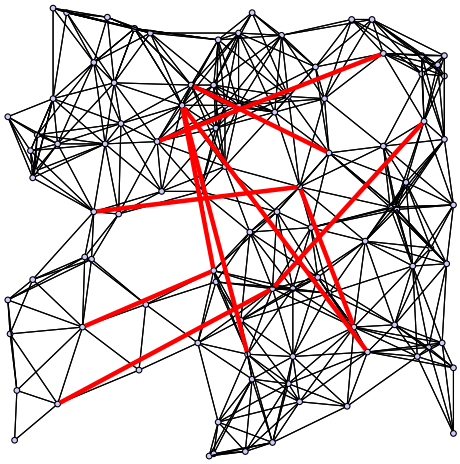
graph *grid* with 5 optimal shortcuts



graph *grid* with 10 optimal shortcuts



graph *disk* with 5 optimal shortcuts



graph *disk* with 10 optimal shortcuts

Fig. 3: Optimal shortcut assignments for some example graphs.

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'

$E' \leftarrow \emptyset$;
for $i = 1,2,\ldots,c$ **do**
  $(x,y) \leftarrow \operatorname{argmax}\{w_{G[E']}(\{(x,y)\}) \mid (x,y) \in V \times V \setminus (E \cup E'),\ \operatorname{dist}(x,y) < \infty\}$
  $E' \leftarrow E' \cup \{(x,y)\}$
output E'.

Theorem 5. Consider an sp-unique graph $G = (V,E,len)$ together with a positive integer $c \in \mathbb{Z}^+$. 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 $\operatorname{dist}(s,t)$ for each pair $s,t \in V$ as well as the shortest-path 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 path subgraphs G_s can be adapted by adding (a,b) to each subgraph G_s with $\operatorname{dist}(s,a) + \operatorname{dist}(a,b) = \operatorname{dist}(s,b)$. 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-path subgraphs can be neglected.

In the remainder of this section, we show how to perform this step in time $O(|V|^3)$ using dynamic programming. Consequently, the GREEDY-strategy can be implemented to work in time $O(c \cdot |V|^3)$. For better readability we assume that the underlying graph is strongly connected. The generalization to arbitrary graphs is straightforward.

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

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

Exploiting this 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 $\text{dist}(s,b)+\text{dist}(b,v) = dist(s,v)$.

Algorithm 2: GREEDY STEP ON SP-UNIQUE GRAPHS

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

```
1 initialize w(·,·) ≡ 0
2 compute h_G(·,·)
3 for s ∈ V do
4 |   for b ∈ V do
5 |   |   compute |P+(s,b)|
6 |   for a ∈ V do
7 |   |   if dist(s,a) + dist(a,b) = dist(s,b) then
8 |   |   |   w(a,b) ← w(a,b) + (h_g(a,b) − 1)|P+(s,b)|
9 output arbitrary (a,b) with maximum w(a,b)
```

The problem with 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*

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

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 5. 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

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

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

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

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 5 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. So, for convenience, we introduce two further abbreviations :

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

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

Lemma 6. 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.

$$\begin{aligned} w(a,b) &= \sum_{s,t \in V} \big(h_G(s,t) - h_{G[(a,b)]}(s,t) \big) \\ &= \sum_{(s,t) \in P(a,b)} \big(h_G(s,t) - h_{G[(a,b)]}(s,t) \big) + \sum_{(s,t) \notin P(a,b)} \underbrace{\big(h_G(s,t) - h_{G[(a,b)]}(s,t) \big)}_{=0} \\ &= \sum_{\substack{(s,t) \in P(a,b) \\ \omega_{sb}(t) < g_s(a,b)}} \big(h_G(s,t) - h_{G[(a,b)]}(s,t) \big) + \sum_{\substack{(s,t) \in P(a,b) \\ \omega_{sb}(t) \geq g_s(a,b)}} \underbrace{\big(h_G(s,t) - h_{G[(a,b)]}(s,t) \big)}_{=0 \text{ with Lemma 5}}. \end{aligned}$$

With Lemma 5, 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 $(s,t) \in P(a,b)$ and hence 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 6 is the key for obtaining 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 $O(|V|^2)$.

Prepared with these values we are ready to apply Lemma 6. 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 6 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 $\Delta_\cdot(\cdot,\cdot)$, $C_\cdot(\cdot,\cdot)$ and $D_\cdot(\cdot,\cdot)$ and Lemma 6. To reach the runtime in $O(|V|^3)$ we answer the question if a shortest s-a-b path exists by checking if $\mathrm{dist}(s,a)+\mathrm{dist}(a,b)=\mathrm{dist}(s,b)$.

Algorithm 3: GREEDY STEP

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

```
compute dist(·,·) and h(·,·)
initialize w(·,·) ≡ 0
initialize Δ_i(·,·) ≡ 0
for s ∈ V do
    for b,t ∈ V do
        if there exists a shortest s-t-path containing b in G then
            j ← compute ω_b(s,t)
            Δ_j(s,b) ← Δ_j(s,b) + 1
    for b ∈ V do
        C_0(s,b) ← Δ_0(s,b)
        D_0(s,b) ← 0
        for r := 1 to n − 1 do
            C_r(s,b) ← C_{r−1}(s,b) + Δ_r(s,b)
            D_r(s,b) ← D_{r−1}(s,b) + r · Δ_r(s,b)
    for a,b ∈ V do
        if there exists a shortest s-b-path containing a and g_s(a,b) > 0 then
            w(a,b) ← w(a,b) + g_s(a,b) · 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)
```

6 Approximation via Partitioning

The second algorithm works for sp-unique graphs with bounded degree and is based on a partition of the nodes. It finds an $O\left(\lambda \cdot \max\left\{1, n^2/(\lambda^2 c)\right\}\right)$ approximation of the optimal solution, where λ is the number of subsets of the underlying partition.

Given an sp-unique graph $G = (V, E, len)$ with bounded degree B, our approximation scheme works as follows. It 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. Furthermore, the approximation ratio depends on the number of subsets. In fact, if each optimal shortcut has both of its endpoints contained in one of the subsets, then the worst case approximation ratio is given by the number of subsets. Otherwise, the following proposition gives us the idea on how to bound the gain of the shortcuts which cross more than one subset.

Proposition 2. Let $G = (V, E, len)$ be an sp-unique graph and $s = (v_1, v_\ell)$ be a shortcut in G. Let $p = (v_1, v_2, \ldots, v_\ell)$ be the shortest v_1-v_l-path. If we divide s into a set of shortcuts $s_1, s_2, \ldots, s_k$ such that $s_1 = (v_{j_0} = v_1, v_{j_1})$, $s_2 = (v_{j_1}, v_{j_2})$, $\ldots$,$s_k = (v_{j_{k-1}}, v_{j_k})$, $j_i - j_{i-1} \geq 2$, for each $i = 1, 2, \ldots, k$ and $\ell - 1 \leq j_k \leq \ell$ then, $w_G(s) \leq 2\sum_{i=1}^{k} w_G(s_i)$.

Proof. Let us denote as $a := h(v_1, v_\ell) = \ell - 2$ the number of edges shortcutted by s and let us denote as b the number of shortest paths shortcutted by s, $b := |P(v_1, v_\ell)|$, that is b is the number of hop-minimal shortest paths in G and it is equal to the number of hop-minimal shortest paths that uses the shortcut s in the graph augmented by adding s. By definition,

$$w(s) = a \cdot b.$$

Similarly, for each $i = 1, 2, \ldots, k$, we can denote as $a_i = j_i - j_{i-1} - 1$ the number of edges shortcutted by s_i and as $b_i := |P(v_j, v_{j-1})|$. Then,

$$w(s_i) = a_i \cdot b_i.$$

As the shortest paths are unique, for each $i = 1, 2, \ldots, k$, all the shortest paths in G that have p as a subpath have also p_i as a subpath. Thus, $b \leq b_i$, for each $i = 1, 2, \ldots, k$. Moreover, $a \leq \sum_{i=1}^{k}(a_i + 1)$ because of $a_i = j_i - j_{i-1} - 1$ and $\sum_{i=1}^{k}(a_i + 1) = j_1 - 1 + j_2 - j_1 + \ldots + j_k - j_{k-1} = j_k - 1 \geq \ell - 2 = a$. Further, for each $i = 1, 2, \ldots, k$ is $j_i - j_{i-1} \geq 2$. Hence $a_i \geq 1$ and $a_i + 1 \leq 2a_i$. It follows that

$$w(s) = a \cdot b \leq \sum_{i=1}^{k}(a_i + 1) \cdot b \leq 2\sum_{i=1}^{k} a_i \cdot b \leq 2\sum_{i=1}^{k} a_i b_i \leq 2\sum_{i=1}^{k} w(s_i).$$

■

A proof of the above proposition which work with a set of shortcut instead of only one is given in the proof of Theorem 6.

In detail, our scheme works as follows. First, we partition the set V into sets $\mathscr{P} = \{P_1, \ldots, P_\lambda\}$, where each P_i has size $size = \sqrt[c]{n^\varepsilon}/B$ (i.e. $\lambda = \lceil n/size \rceil$) for an arbitrary $\varepsilon > 0$. 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

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

Finally, we determine the set C_i, for which the shortcut assignment yields the most gain.

Algorithm 4: PARTITION

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

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

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

The following theorem shows the approximation ratio for PARTITION.

Theorem 6. Given a weighted, directed, SP-unique graph $G = (V, E, len)$ with bounded degree and a positive integer $c \in \mathbb{N}$. Then, the solution computed by PARTITION is an $O\left(\lambda \cdot \max\left\{1, \frac{n^2}{\lambda^2 c}\right\}\right)$ approximation for the optimal solution of the SHORTCUT PROBLEM instance (G, c) where λ denotes the number of sets used by PARTITION.

Proof. Let S^* denote an optimal solution to (G, c), $\mathscr{P}$ the partition and $\mathscr{C}$ the set of neighbourhoods C_i used by PARTITION. Let R be the set of (a, b) in $V \times V$ such that $\operatorname{dist}(a, b) \neq \infty$.

For each $(a, b) \in R$, we choose an arbitrary hop-minimal shortest a-b-path in $G[S^*]$. Let $S^*(a, b) \subseteq S^*$ be the set of shortcuts on this path. Note that the shortest paths represented by the shortcuts in $S^*(a, b)$ are edge-disjoint.

Now each $s \in S^*$ is divided into a set of shortcuts as follows: Let $p = (v_1, \ldots, v_r)$ be the path represented by s, $D(s) := \emptyset$ and $i := 1$. As long as $i < r - 1$, proceed as follows: Let P be the cell of v_{i+1} in $\mathscr{P}$. Let j be the first index greater than $i + 1$ such that $v_j \notin P$, or r if such an index does not exist. Set $D(s) = D(s) \cup \{(v_i, v_j)\}$ and $i := j$.

For a shortcut $s = (v_1, v_2)$, let $h_G(s)$ denote the hop-distance between v_1 and v_2 in G. It is easy to see that for each $s' \in D(s)$, $h_G(s') \geq 2$ and the shortest path represented by s' is contained completely in at least one set in $\mathscr{C}$. Furthermore, for each s in S^*, $\sum_{s' \in D(s)} h_G(s') \geq h_G(s) - 1$, hence $2 \cdot \sum_{s' \in D(s)} (h_G(s') - 1) \geq h_G(s) - 1$.

Let $S = \bigcup_{s \in S^*} D(s)$ and for each $s \in S$, $U(s)$ be the set of $s^* \in S^*$ with $s \in D(s)$. We now divide the shortcuts in S according to the sets C_i in the following way. Let S_i be the set of $s \in S$ such that the shortest path represented by s is completely contained in C_i. Due to the construction of S, this is a cover of S, i. e. $\bigcup_{i=1}^{\lceil n/\text{size} \rceil} S_i = S$.

Claim. $w(S^*) \leq 2 \cdot \sum_{i=1}^{\lceil n/size \rceil} w(S_i)$

As shortest paths in G are unique, the insertion of S^* decreases the hop-distance between two nodes a and b with $\operatorname{dist}(a, b) \neq \infty$ by the sum of the hop-lengths of the shortest paths represented by the shortcuts in $S^*(a, b)$ minus $|S^*(a, b)|$. Furthermore, as shown above, for each s^* in S^*, $h_G(s^*) - 1 \leq 2 \cdot \sum_{s \in D(s^*)} (h_G(s) - 1)$ and thus,

$$\begin{aligned} w(S^*) &= \sum_{(a,b) \in R} \sum_{s^* \in S^*(a,b)} (h_G(s^*) - 1) \leq 2 \cdot \sum_{(a,b) \in R} \sum_{s^* \in S^*(a,b)} \sum_{s \in D(s^*)} (h_G(s) - 1) \\ &= 2 \cdot \sum_{(a,b) \in R} \sum_{s \in S} \sum_{s^* \in S^*(a,b) \cap U(s)} (h_G(s) - 1) \end{aligned}$$

Assume that $S^*(a,b) \cap U(s)$ contains two shortcuts s_1 and s_2. As $s_1, s_2 \in S^*(a,b)$, the shortest paths represented by s_1 and s_2 are edge-disjoint, contradicting the fact that they both contain the shortest path represented by s. Hence, $|S^*(a,b) \cap U(s)| \leq 1$ and

$$w(S^*) \leq 2 \cdot \sum_{(a,b)\in R} \sum_{s\in S} \sum_{s^* \in S^*(a,b)\cap U(s)} (h_G(s)-1) = 2 \cdot \sum_{(a,b)\in R} \sum_{\substack{s\in S:\\ S^*(a,b)\cap U(s)\neq\emptyset}} (h_G(s)-1)$$

$$\leq 2 \cdot \sum_{(a,b)\in R} \sum_{i=1}^{\lceil n/\text{size}\rceil} \sum_{\substack{s\in S_i:\\ S^*(a,b)\cap U(s)\neq\emptyset}} (h_G(s)-1)$$

For fixed $(a,b) \in R$, let s_1 and $s_2 \in S_i$ be such that there exist $s_1^* \in S^*(a,b) \cap U(s_1)$ and $s_2^* \in S^*(a,b) \cap U(s_2)$. Due to the definition of $S^*(a,b)$, the shortest paths represented by s_1^* and s_2^* are edge-disjoint, implying that the same holds for s_1 and s_2. Hence,

$$w(S^*) \leq 2 \cdot \sum_{(a,b)\in R} \sum_{i=1}^{\lceil n/\text{size}\rceil} \sum_{\substack{s\in S_i:\\ S^*(a,b)\cap U(s)\neq\emptyset}} (h_G(s)-1) \leq 2 \cdot \sum_{(a,b)\in R} \sum_{i=1}^{\lceil n/\text{size}\rceil} (h_G(a,b) - h_{G[S_i]}(a,b))$$

$$= 2 \cdot \sum_{i=1}^{\lceil n/\text{size}\rceil} w(S_i)$$

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

It follows that,

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

Hence, the approximation ratio can be bound as follows.

$$\frac{w(S^*)}{w(S')} \leq 2\left\lceil \frac{n}{size}\right\rceil \max\left\{1, \frac{size^2 \cdot B^2}{c}\right\} = O\left(\frac{n}{size} \max\left\{1, \frac{size^2}{c}\right\}\right)$$

$$= O\left(\frac{n}{\sqrt[c]{n^\varepsilon}} \max\left\{1, \frac{\sqrt[c]{n^{2\varepsilon}}}{c}\right\}\right) = O\left(\lambda \cdot \max\left\{1, \frac{n^2}{\lambda^2 c}\right\}\right) .$$

■

7 Approximative 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. 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 $\mu(S) := \sum_{s,t \in V} h_{G[S]}(s,t)$, the adaption to the SHORTCUT PROBLEM is straightforward. More concrete, we apply the sampling technique to obtain an unbiased estimate for $\mu(S)$ and apply *Hoeffding's Bound* [11] to get a confidence intervall for the outcome.

Theorem 7 (Hoeffding's Bound). If $X_1, X_2, \ldots, X_K$ are real valued independent random variables with $a_i \leq X_i \leq b_i$ and expected mean $\mu = \mathbb{E}[\sum X_i/K]$, then for $\xi > 0$

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

Given is a shortcut assignment S. Let $X_1, \ldots, X_k$ be a family of random variables such that for $i = 1, \ldots, K$, the variable X_i equals $|V| \sum_{t \in V} h_{G[S]}(s_i, t)$ where $s_i \in V$ is a vertex chosen uniformly at random. We estimate $\mu(S)$ by $\hat{\mu} := \sum_{i=1}^{K} X_i/K$.

Because of $\mathbb{E}(\hat{\mu}) = \mu(S)$ we can apply Hoeffding's Bound if we know lower and upper bounds for the X_i. The values 0 and $|V|^3$ are trivial such bounds. We introduce the notion of *shortest path 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 $0 \leq X_i \leq |V|^2 \mathrm{spDiam}(G)$ yields

$$\mathbb{P}\{|\hat{\mu} - \mu| \geq \xi\} \leq 2e^{-2K\xi^2/(|V|^4 \cdot \mathrm{spDiam}(G)^2)}$$

and

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

for a parameter l_{rel} stating the relative size of the confidence intervall.

The codomain of the variables X_i is finite as V is finite. In [11] 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 $G = (V, E \cup E', len)$, size of confidence intervall l_{rel}, significance level α
output: approximation $\hat{\mu}$ for $\mu = \sum_{s,t \in V} h_G(s,t)$

compute random order $v_1, v_2, \ldots, v_n$ of V
compute upper bound $\overline{\mathrm{spDiam}}$ for shortest path diameter
$i \leftarrow 1$; sum $\leftarrow 0$; $\hat{\mu} \leftarrow -\infty$
while *not* ($i = |V| + 1$ *or* $2 \cdot \exp(-2i(\hat{\mu} \cdot l_{rel})^2/(|V|^4 \overline{\mathrm{spDiam}}(G)^2)) \leq \alpha$) **do**
    $T \leftarrow$ grow shortest-paths tree rooted at v_i (favoring edge-minimal shortest paths)
    sum $\leftarrow$ sum $+ |V| \cdot \sum_{t \in V} h'(v_i, t)$
    $\hat{\mu} \leftarrow$ sum $/ i$
    $i \leftarrow i + 1$
output $\hat{\mu}$

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 and for which a large number of shortcut assignments shall 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 $\overline{\text{diam}}(G)$ for the diameter of G. To do so we choose a set of nodes $s_1, s_2, \ldots, s_l$ uniformly at random. 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. We set $\overline{\text{diam}}(G)$ to be the minimum of these values over all s_i. Note that this approach does not directly transfer to the shortest paths diameter as, in general, $\max_{v \in V} h_G(v, s_i) + \max_{v \in V} h_G(s_i, v)$ is not an upper bound for spDiam(G). The bound $\overline{\text{diam}}(G)$ is a 2-approximation for the exact diameter diam(G) of G already for $l = 1$ as there is are $u, v \in V$ such that

$$\overline{\text{diam}}(G) = \text{dist}(u, s_1) + \text{dist}(s_1, v) \leq \text{diam}(G) + \text{diam}(G) = 2 \cdot \text{diam}(G).$$

Let $\text{len}_{\max}$ and $\text{len}_{\min}$ denote the lengths of a longest and a shortest edge in G, respectively. The value $\overline{\text{diam}}(G)/\text{len}_{\min}$ is a $2 \cdot \text{len}_{\max}/\text{len}_{\min}$ approximation for the shortest path diameter as $\text{spDiam}(G) \geq \text{diam}(G)/\text{len}_{\max}$.

A more expensive approach works as follows: After computing $\overline{\text{diam}}(G)$, we choose a tuning parameter η. Then we grow, for every node s on G, a shortest paths tree whose construction is stopped as soon as one vertex with distance of more than $\overline{\text{diam}}(G)/\eta$ is visited. 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. Then $\overline{\text{spDiam}} = \tau_{\max} \cdot \eta$ is an upper bound for the shortest path diameter of G (and an η-approximation as $\text{spDiam}(G) \geq \tau_{\max}$). The pseudocode of that algorithm is given as Algorithm 6.

Algorithm 6: COMPUTE UPPER BOUND FOR SHORTEST-PATHS DIAMETER

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

```
1  diam(G) ← ∞; τ ← 0;
2  for i = 1,...,l do
3      s ← choose node uniformly at random
4      grow shortest paths tree rooted at s
5      grow shortest paths tree rooted at s on the reverse graph ←G
6      diam(G) ← min{diam(G), max_{v∈V}{dist(s,v)} + max_{v∈V}{dist(v,s)}}
7  for s ∈ V do
       T     ← grow partial shortest paths tree rooted at s (favoring edge-minimal shortest paths).
8              stop growing the tree when the first node with dist(s,v) > diam(G)/η is visited.
       τ_max ← max{τ_max, maximal number of edges of a path in T }
9  output spDiam := τ_max · η
```

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 [13] 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 path 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 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 a more sophisticated flow-like formulation that requires $O(|V|^3)$ variables.

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 $O\left(\lambda \cdot \max\left\{1, |V|^2/(\lambda^2 c)\right\}\right)$ approximation of the optimal solution where λ is the number of subsets of the underlying partition. Let $f(|V|)$ be the time needed for computing all pairs shortest paths. The runtime of the second approach lies in $O(f(|V|) \cdot (|V|/\lambda)^{2c} \cdot \lambda)$ if the input graph has bounded degree.

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. There exists a wide range of possible future work on the problem. From a theoretical point of view the probably most interesting open question is that of the approximability of the SHORTCUT PROBLEM. It is still not known if it is in APX. Furthermore, it would be helpful to identify graph classes for which the SHORTCUT PROBLEM or the REVERSE SHORTCUT PROBLEM become tractable.

From an experimental point of view, it would be interesting to develop heuristics that find good shortcuts for large real-world input. In particular, evolutionary algorithms and local search algorithms seem to be promising.

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. N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):713–721, 1963.
12. 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.
13. PTV AG - Planung Transport Verkehr. `http://www.ptv.de`, 2008.
14. 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.
15. P. Sanders and D. Schultes. Engineering Fast Route Planning Algorithms. In C. Demetrescu, editor, *Proceedings of the 6th Workshop on Experimental Algorithms (WEA'07)*, volume 4525 of *Lecture Notes in Computer Science*, pages 23–36. Springer, June 2007.
16. 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.
17. 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.
18. 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.
19. A. Schumm. Heuristic Algorithms for the Shortcut Problem. Master's thesis, July 2009.
20. 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.