The Shortcut Problem – Complexity and Approximation*

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

[1] Faculty of Informatics, Universität Karlsruhe (TH)
{rbauer,delling,wagner}@informatik.uni-karlsruhe.de
[2] Department of Electrical and Information Engineering, University of L'Aquila
gdangelo@ing.univaq.it

Abstract. During the last years, speed-up techniques for DIJKSTRA's algorithm have been developed that make the computation of shortest paths a matter of microseconds even on huge road networks. The most sophisticated methods enhance the graph by inserting *shortcuts*, i.e. additional edges, that represent shortest paths in the graph. Until now, all existing shortcut-insertion strategies are heuristics and no theoretical results on the topic are known. In this work, we formalize the problem of adding shortcuts as a graph augmentation problem, study the algorithmic complexity of the problem, give approximation algorithms and show how to stochastically evaluate a given shortcut assignment on graphs that are too big to evaluate it exactly.

1 Introduction

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

Often, this problem is coped with by dividing the computation of the shortest paths into two stages. In the offline stage, some data is precomputed that is used in the online stage to answer a query heuristically faster than DIJKSTRA's algorithm. Such an algorithm is called a *speed-up technique*. During the last years, speed-up techniques have been developed for road networks (see [14,18] for an overview), that make the shortest path computation a matter of microseconds [4] even on huge road networks consisting of millions of nodes and edges. One core part of many of these speed-up techniques is the insertion of *shortcuts* [3,5,7,8,9,11,13,15,16,17], 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 speed-up technique. Until now, all existing shortcut insertion strategies are heuristics and no theoretical worst-case or average case results are known.

* Partially supported by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. FP6-021235-2 (project ARRIVAL) and the DFG (project WAG54/16-1).

M. Nielsen et al. (Eds.): SOFSEM 2009, LNCS 5404, pp. 105–116, 2009.
© Springer-Verlag Berlin Heidelberg 2009

All speed-up techniques that use shortcuts have one point in common. The shortcuts reduce the search space which corresponds to reducing the number of edges in a shortest path in the graph. Therefore, it seems reasonable to insert shortcuts in a manner that minimizes the average number of edges of a shortest path in the graph but keeps the space occupancy low. In this work we formalize this idea by defining the SHORTCUT PROBLEM (SP) and give a theoretical study of the complexity of the problem. In particular, the arc-flag method [12] can easily be enriched by externally computed shortcuts. Therefore, considering SP independent from a specific speed-up technique can lead to a reversed process: first compute shortcuts according to SP and then apply a speed-up technique on the resulting graph. SHARC-Routing, a sophisticated variant of arc-flags, already uses externally computed shortcuts. Therefore, we consider it as a strong candidate to benefit from that approach (details in [3]). Finally, besides its relevance as a first step at all towards theoretical results on speed-up techniques, we consider the problem to be interesting on its own.

To the best of our knowledge, the problem of finding shortcuts as stated in this work has never been treated before. Speed-up techniques that incorporate the usage of shortcuts are the following. Given a graph $G = (V,E)$ the multi-level overlay graph technique [5,11,15,16,17] uses some centrality measures or separation strategies to choose a set of important nodes V' on 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 the same in (V,E) and (V',E'). Highway hierarchies [13] and reach based pruning [8,9] 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 and for (nearly) each pair of edges (u,v), (v,w) a shortcut (u,w) is inserted. SHARC-Routing [3] and Contraction Hierarchies [7] use a similar strategy.

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. Two approximation algorithms of SP that work on a special graph class and the corresponding worst-case bounds are reported in Section 4. A stochastical approach to evaluate a given solution of SP is introduced in Chapter 5. Our work is concluded by a summary and possible future work in Section 6.

Some proofs in the paper have been omitted due to space restrictions. The full version containing all proofs can be found here [2].

2 Preliminaries

Throughout the work $G = (V,E,len)$ denotes a directed, weighted, graph with n nodes, m edges and positive length function $len : E \to \mathbb{R}^+$. Given a node v, $N(v)$ denotes the set of neighbors of v, that is the set of nodes $u \in V$ such that $(u,v) \in E$ or $(v,u) \in E$. Given a set S of nodes, the *neighborhood* of S is the set $S \cup \bigcup_{u \in S} N(u)$. A path P from x_1 to x_n in G is a finite sequence $x_1, x_2, \ldots, x_n$ of nodes such that $(x_i, x_{i+1}) \in E$, $i = 1, \ldots, n-1$. The *length* of a path P in G is the sum of the length of all edges in P. A shortest path between nodes s and t is a path from s to t with minimum length. By $P(s,t)$ we denote the set of all shortest s-t-paths. The *hop-length* $|P|$ of a path P in G is the number of

edges in P. Given two nodes s,t the distance $dist(s,t)$ from s to t is the length of a shortest path between s and t, while the hop-distance $h(s,t)$ from s to t is the hop-length of a hop-minimal shortest path between s and t. The *diameter* of a graph is the length of the longest distance in G. The *reverse graph* $\overline{G} = (V, \overline{E}, \overline{len})$ is the one obtained from G by substituting each $(u,v) \in E$ by (v,u) and by defining $\overline{len}(v,u) = len(u,v)$. The eccentricity $\varepsilon_G(v)$ of a node v is the maximum distance between v and any other node u of G.

A shortcut is an edge (u,v) such that $len(u,v) = dist(u,v)$. The notation $G' = (V, E \cup E', len')$ indicates a supergraph of G with shortcuts E' whereas $len' : E \cup E' \to \mathbb{R}^+$ is such that $len'(u,v)$ equals $dist(u,v)$ if $(u,v) \in E'$ and equals $len(u,v)$ otherwise. Further, $h'(s,t)$ denotes the hop-distance from s to t in G'.

3 Problem 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.

The SHORTCUT PROBLEM consists of adding a number c of shortcuts to a graph, such that the sum of the hop lengths of hop-minimal shortest paths on the graph becomes minimal.

Definition 1 (SHORTCUT PROBLEM (SP)). *Given a graph $G = (V,E,len)$, a positive integer $c \in \mathbb{N}$, find a graph $G' = (V, E \cup E', len')$ such that $|E'| \leq c$ and*

$$w(E') := \sum_{s,t \in V} h(s,t) - \sum_{s,t \in V} h'(s,t)$$

is maximal, whereas $len' : E \cup E' \to \mathbb{R}^+$ equals $dist(u,v)$ if $(u,v) \in E'$, equals $len(u,v)$ otherwise, $h(s,t)$ denotes the hop distance in (V,E) and $h'(s,t)$ denotes the hop distance in $(V, E \cup E')$.

We call $w(E')$ the *decrease in overall hop length*. The REVERSE SHORTCUT PROBLEM (RSP) is the variant of Definition 1 for which the decrease in overall hop length $w(E')$ must be at least a given value k and the objective is to minimize $|E'|$.

Definition 2 (REVERSE SHORTCUT PROBLEM (RSP)). *Given a graph $G = (V,E,len)$, a positive integer $k \in \mathbb{N}$, find a graph $G' = (V, E \cup E', len')$ such that*

$$w(E') := \sum_{s,t \in V} h(s,t) - \sum_{s,t \in V} h'(s,t) \geq k$$

and $|E'|$ is minimal, whereas $len' : E \cup E' \to \mathbb{R}^+$ equals $dist(u,v)$ if $(u,v) \in E'$, equals $len(u,v)$ otherwise, $h(s,t)$ denotes the hop distance in (V,E) and $h'(s,t)$ denotes the hop distance in $(V, E \cup E')$.

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

Definition 3 (MIN SET COVER). *Given a collection C of subsets of a finite set U find a minimum cardinality set cover of U, i.e. a subset C' of C such that every element in U belongs to at least one member of C' and that $|C'|$ is minimal.*

Given an instance $I = (C,U)$ of MIN SET COVER we construct an instance $I' = (G,k)$ of RSP the following way (see Figure 1 for a visualization): we denote by Δ the value $2|C|+1$. We introduce a node s to G. For each $u_i \in U$, we introduce a set of nodes $U_i = \{u_1^i, \ldots, u_\Delta^i\}$ to G. For each C_i in C, we introduce nodes C_i^-, C_i^+ and edges (C_i^-, C_i^+), (C_i^+, s) to G. The graph furthermore contains, for each $u_i \in U$ and each $C_j \in C$ with $u_i \in C_j$, the edges $(u_r^i, C_j^-), r = 1, \ldots, \Delta$. All edges are directed and have length 1. Finally we set k to be $\Delta \cdot |U|$. The transformation is polynomial.

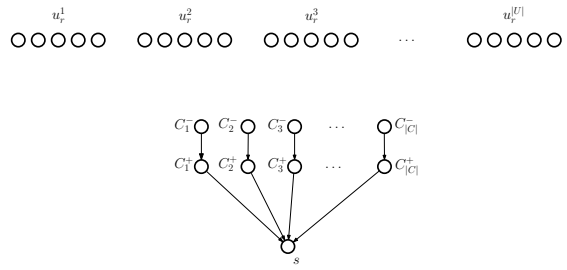

Fig. 1. Instance I' after the transformation from MIN SET COVER (edges of the form (u_r^ℓ, C_j^-) are not drawn as they depend on the instance I)

Lemma 1. *Let C' be a solution of I, then there exists a solution E' of I' with $|E'| = |C'|$.*

It directly follows that an optimal solution E' of the instance I' satisfies $|E'| \leq |C|$.

Lemma 2. *There is an optimal solution E' of I' that only consists of shortcuts of the form (C_i^-, s) for some $i \in \mathbb{N}$.*

Proof. Let $\widetilde{E}$ be an optimal solution of I'. We decompose $\widetilde{E} = \widetilde{E}_A \dot\cup \widetilde{E}_B$ such that $\widetilde{E}_A$ only contains shortcuts of the form (C_i^-, s) for some i and $\widetilde{E}_B$ contains all other shortcuts. If $\widetilde{E}_B$ is not empty there exists an $\ell \in \mathbb{N}$ such that for every $j, r \in \mathbb{N}$ for which $(u_r^\ell, C_j^-) \in E$ the shortcut (C_j^-, s) is not contained in $\widetilde{E}_A$. Otherwise $\widetilde{E}_A$ would be a feasible solution and $|\widetilde{E}_A| < |\widetilde{E}|$ in contradiction to the optimality of $\widetilde{E}$.

We fix such an ℓ. Let $i \in \mathbb{N}$ be such that there is an edge $(u_1^\ell, C_i^-) \in E$. Further, let p_ℓ denote the number of nodes in $U_\ell = \{u_r^\ell \in V \mid r = 1 \ldots \Delta\}$ such that a shortcut (u_r^ℓ, s) or (u_r^ℓ, C_j^+) is in $\widetilde{E}_B$ for some $j \in \mathbb{N}$. Assume that $p_\ell > 1$. Then, we could delete all shortcuts outgoing from a node in U_ℓ from $\widetilde{E}_B$ (this increases the overall hop length by at maximum $2|C|$) and introduce the shortcut (C_i^-, s) in $\widetilde{E}_A$ (this decreases the overall hop length by at least $\Delta + 1 = 2|C| + 2$). This solution would be better than the old one in contradiction to the optimality of $\widetilde{E}_A \dot\cup \widetilde{E}_B$. Hence, p_ℓ is at most 1.

We now state a polynomial time algorithm that computes a desired solution out of the given solution $\widetilde{E}$. We repeatedly proceed as follows until $\widetilde{E}_B$ is empty. First, we find an ℓ as defined above and a j such that $(u_1^\ell, C_j^-) \in E$. If no such ℓ exists, $\widetilde{E}_A$ is the desired solution. In case $p_\ell = 1$ we delete the shortcut with source node in U_ℓ. In case $p_\ell = 0$ we delete an arbitrary shortcut with source node in an $U_i, i = 1, \ldots, |U|$. If $w(\widetilde{E})$ still is high enough we do nothing. Otherwise we insert the shortcut (C_j^-, s). Obviously, the algorithm runs in polynomial time and computes a desired solution. ■

Lemma 3. *Let E' be an optimal solution of I'. Then, there exists a solution C' of I with $|E'| = |C'|$.*

Proof. Let E' be an optimal solution of I'. By Lemma 2, we know that there exists an optimal solution E'' with $|E''| = |E'|$ and shortcuts of the form (C_i, s) for some $i \in \mathbb{N}$.

We denote by U'' the collection of sets U_i for which there is a shortcut (C_j^-, s) in E'' and edges (u_r^i, C_j^-) in E. As E'' is a feasible solution, we know that $w(E'') = |E''| + \Delta|U''| \geq k = \Delta|U|$. Because of $|E''| \leq |C| < \Delta$ we know that $|U''| = |U|$ which means that for every node in u_r^i, there is a shortcut on a path to s. Therefore, the set $\{C_i \mid (C_i^-, s) \in E''\}$ is a solution of I. ■

Theorem 1. *Unless $P = NP$, no polynomial-time constant factor approximation algorithm exists for* RSP.

The SHORTCUT DECISION PROBLEM (SDP) is the variant of Definition 1 where the aim is to decide for a given $c \in \mathbb{N}$ and a given $k \in \mathbb{N}$ whether there exists a shortcut assignment E' with $|E'| = c$ and $w(E') \geq k$. The proof of the non-approximability of RSP directly transfers to a proof for the *NP*-completeness of SDP.

Corollary 1. SP *and* RSP *are NP-hard,* SDP *is NP-complete.*

Theorem 2. *Unless $P = NP$, no polynomial-time algorithm exists that approximates* SP *up to an additive constant.*

Proof. Assume there exists a polynomial-time algorithm $\mathscr{A}$ that approximates SP within a fixed bound of Δ. Then we can solve SDP in polynomial time as follows.

Let $(G = (V, E, len), c, k)$ be an instance of SDP. We create a new instance $(\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{len}), c)$ of SP by adding, for each node v in G, $\chi := \Delta + 1 + n^2$ nodes $v_1, \ldots, v_\chi$ and directed edges (v_i, v) such that $\widetilde{len}(v_i, v) = 1$, $i = 1 \ldots \chi$. This can be done in polynomial time.

We will first prove, that the endpoints of all shortcuts inserted in $\widetilde{G}$ by $\mathscr{A}$ will be in V: Let E' be the set of all shortcuts inserted by $\mathscr{A}$ in $\widetilde{G}$. Assume there is a shortcut $(\widetilde{u}, v) \in (\widetilde{V} \setminus V) \times V$ in E'. This shortcut only decreases the overall hop length on shortest paths by at most n^2. W.l.o.g we assume that it is possible to insert c shortcuts into G. Therefore there must be a shortcut $(x, y) \in V \times V$ that is not contained in E'. This shortcut will result in a decrease of overall hop length of at least χ. Therefore, deleting $(\widetilde{u}, v)$ and inserting (x, y) would decrease the overall hop length on shortest path by more than Δ which is a contradiction to the approximation bound of $\mathscr{A}$.

With $\Delta := 0$ it directly follows that an optimal solution of I' only consists of shortcuts in $V \times V$. Given a set of shortcuts $E' \subseteq V \times V$ we denote the overall decrease of hop

length in G with $w(E')$ and in $\widetilde{G}$ with $\widetilde{w}(E')$. It is $\widetilde{w}(E') = (1+\chi)w(E')$. Given an optimal solution E^* for I and I', it follows $(1+\chi)\,(w(E^*) - w(E)) = \widetilde{w}(E^*) - \widetilde{w}(E) \leq \Delta$. Hence, $w(E^*) - w(E) \leq \frac{\Delta}{1+\chi} < 1$ which implies $w(E^*) = w(E')$ as both $w(E^*)$ and $w(E')$ are integer values. Therefore, we have a polynomial time, exact algorithm for solving SP. We can use this algorithm to decide SDP in polynomial time. ■

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 SP is polynomial in the size of the graph:

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

Evaluating a given solution means solving the APSP, hence this can be done in time $O(n(n\log n + m))$. For this reason, the whole problem 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.

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 prove that in the following lemma and corollary.

Lemma 4. *Given a graph $G = (V,E)$ having unique shortest paths, a set of shortcuts S and a shortcut s. Then, $w(S \cup \{s\}) \leq w(S) + w(s)$.*

Corollary 2. *Given a graph $G = (V,E)$ having unique shortest paths and a set of shortcuts $S = \{s_1, s_2, \ldots, s_k\}$. Then, $w(S) \leq \sum_{i=1}^{k} w(s_i)$.*

In the next section we use these results to present approximation algorithms which work in the case of graphs where shortest paths are unique for each pair of nodes.

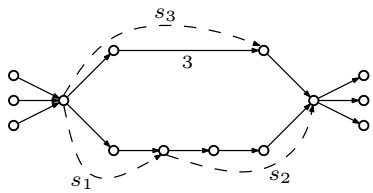

Fig. 2. Example Graph G with shortcuts s_1, s_2, s_3, all edges for which no weight is given in the picture have weight 1

4 Approximation Algorithms

In this section, we propose two polynomial time algorithms for approximatively solving SP in the special case that, for each pair s,t of nodes on the underlying graph, the shortest s-t-path is unique. 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. The first algorithm is a greedy strategy that consists of iteratively solving the problem where the number of shortcuts allowed is one. This algorithm finds a c-approximation of the optimal solution. The second algorithm works for 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.

4.1 The Greedy Strategy

Given a weighted directed graph $G = (V,E,len)$, the GREEDY approximation scheme consists of iteratively constructing a sequence $G = G_0, G_1, \ldots, G_c$ of graphs where G_{i+1} is the graph that results from solving SP on G_i for which the number of shortcuts allowed to insert is one. GREEDY is a polynomial time c-approximation of SP. The approximation bound follows directly from Lemma 4. In detail, our scheme works as follows.

The value of $w(s)$ in G_i can be computed by performing an all pairs shortest paths computation in G_i. Hence, each iteration step of GREEDY can be solved by evaluating every of the $O(n^2)$ possible shortcuts. This gives an overall time complexity of $O(cn^2 \cdot n(n\log n + m))$. The following theorem shows the approximation ratio for GREEDY.

Theorem 3. *Given is a weighted directed graph $G = (V,E,len)$ with unique shortest paths and a positive integer $c \in \mathbb{N}$. If S^* is the set of shortcuts in an optimal solution for* SP *instance (G,c), then the solution $G' = (V, E \cup E', len')$ computed by* GREEDY *is a factor c-approximation.*

Proof. Let us consider s_1, that is the shortcut computed by the first iteration of GREEDY. Then, $w(s) \leq w(s_1)$ for each $s \in V \times V$. Moreover, for any $S \subseteq V \times V$ and for each $s \in S$, $w(s) \leq w(S)$ and, by Corollary 2, $w(S) \leq \sum_{s \in S} w(s)$. If we write $S^* = \{s_1^*, s_2^*, \ldots s_c^*\}$, it follows that

$$w(S^*) \leq \sum_{i=1}^{c} w(s_i^*) \leq \sum_{i=1}^{c} w(s_1) = cw(s_1) \leq cw(E').$$

Hence, the inequality $w(S^*)/w(E') \leq c$ holds.

■

4.2 Approximation via Partitioning

Given a weighted, directed graph $G = (V,E,len)$ with bounded degree B, our approximation scheme works as follows. It partitions V into small subsets, solves SP restricted to each subset and then chooses the best solution among all subsets as an approximated solution. If the subsets are small enough, then SP restricted to each set can be solved

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, we use the following lemma to bound the decrease in overall hop length on shortest paths of the shortcuts which cross more than one subset.

Lemma 5. *Let $G = (V,E,len)$ be a weighted directed graph with unique shortest paths and $s = (v_1, v_\ell)$ be a shortcut in G. Let $p = (v_1, v_2, \ldots, v_\ell)$ be the shortest path shortcut by s. 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, $w(s) \leq 2\sum_{i=1}^{k} w(s_i)$.*

Corollary 3. *Let $G = (V,E,len)$ be a weighted directed graph where the shortest paths are unique and let $S = \{s_1, s_2, \ldots, s_k\}$ be a set of shortcuts in G. For each $s_i \in S$, let S_i be a set of shortcuts that fullfills the condition of Lemma 5 with respect to s_i. Then, $w(S) \leq 2\sum_{i=1}^{k} w(S_i)$.*

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 cell $P_i \in \mathscr{P}$, 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 $\widetilde{S}_i = \text{argmax}\{w(S) \mid S \subseteq C_i \times C_i \text{ and } |S| \leq c\}$. Finally, we determine the set C_i, for which the shortcuts gain the most overall decrease in hop length and set the solution to be the according shortcuts. More formally, we compute $G' = (V, E \cup \widetilde{S}, len')$ where $\widetilde{S} = \text{argmax}\{w(\widetilde{S}_i) \mid i = 1,2,\ldots,\lambda\}$ and $len' : E \cup \widetilde{S} \to \mathbb{R}^+$ is defined such that $len'(u,v)$ equals $dist(u,v)$ if $(u,v) \in \widetilde{S}$ and equals $len(u,v)$ otherwise.

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 $\lambda = \lceil n/size \rceil = \lceil nB/\sqrt[c]{n^\varepsilon} \rceil$ partitions, 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 4. *Given a weighted directed graph $G = (V,E,len)$ with bounded degree and unique shortest paths 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* SP *instance (G,c) where λ denotes the number of cells used by* PARTITION.

5 Approximative Evaluation of the Measure Function

To evaluate the overall decrease in hop length for a given shortcut assignment, we require computing all pairs shortest paths in a graph. Since this computation requires $O(n(n\log n + m))$ time, we provide a stochastical method to quickly evaluate the overall decrease in hop length in this section. This approach can be used for big networks, where APSP is prohibitive. Such networks often arise in the context of timetabling or

Algorithm 1. PARTITION

input : graph $G = (V, E, len)$, number of shortcuts c, parameter $\varepsilon > 0$
output: graph $G' = (V, E \cup \widetilde{S}, len')$ with shortcuts added

1 Partition the set V into sets $\mathscr{P} = \{P_1, \ldots, P_\lambda\}$ each of size $size = \sqrt[3]{n^\varepsilon}/B$.
2 **forall** $P_i \in \mathscr{P}$ **do**
3 $C_i := P_i \cup \{u \in N(v) \mid v \in P_i\}$
4 $\widetilde{S}_i := \operatorname{argmax}\{w(S) \mid S \subseteq C_i \times C_i \text{ and } |S| \leq c\}$
5 $\widetilde{S} := \operatorname{argmax}\{w(\widetilde{S}_i) \mid i = 1, 2, \ldots, \lambda\}$
6 $len' : E \cup \widetilde{S} \to \mathbb{R}^+$ is such that as $len'(u,v) = dist(u,v)$ if $(u,v) \in \widetilde{S}$, otherwise $len'(u,v) = len(u,v)$.
7 Output $G' = (V, E \cup \widetilde{S}, len')$

shortest-paths computation on road networks (see [6] for a prominent example). For the sake of simplicity we state the approach for the evaluation of $\mu := \sum_{s \in V} \sum_{t \in V} h'(s,t)$, the adaption to SP is straightforward.

More precisely, we apply the sampling technique to evaluate the measure function μ in an approximative way. We exploit *Hoeffding's Bound* [10] to get a confidence intervall of the following unbiased estimation: 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 $\xi > 0$

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

Let $X_1, X_2, \ldots, X_K$ be a family of random variables. For $i = 1, 2, \ldots, K$, X_i equals $|V| \cdot \sum_{t \in V} h'(s_i, t)$ where s_i is a node which is chosen uniformly at random. We estimate μ by $\hat{\mu} := \sum_{i=1}^K X_i / K$. Because of $\mathbb{E}(\hat{\mu}) = \mu$ we can apply Hoeffding's Bound if we know an upper bound for the X_i. The value $|V|^3$ is a trivial upper bound.

Definition 4. *The* shortest path diameter $\mathrm{spDiam}(G)$ *of a graph* $G = (V, E, len)$ *is the maximal hop length of a shortest path (shortest with respect to len) on* G.

If we know the shortest path diameter of a graph we obtain $|V|^2 \mathrm{spDiam}(G)$ as upper bound for X_i. If we insert this into Hoeffdings Bound, we gain

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

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}. In [10] it is stated that Hoeffdings Bound stays correct if, when sampling from a finite population, the samples are being chosen without replacement. Algorithm 2 is an approximation algorithm that exploits the above inequality and that samples without replacement.

To compute the exact shortest path diameter of a graph we have to compute APSP. We obtain an upper bound for the shortest path diameter the following way: first we

Algorithm 2. STOCH. EVALUATE OVERALL HOP LENGTH

input : graph $G = (V, E, len)$, size of confidence intervall l_{rel}, significance level α
output: approximation for the overall hop length on shortest paths

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

compute an upper bound $\overline{\text{diam}}(G)$ for the diameter of G. To do that we choose a set of nodes $s_1, s_2, \ldots, s_l$ uniformly at random. For each node s_i the value $\varepsilon_G(s_i) + \varepsilon_{\overline{G}}(s_i)$ 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. Afterwards, 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 where η is a tuning parameter. We denote by τ the maximum hop-length of the shortest paths on any of the trees grown. Then $\overline{\text{spDiam}} = \tau \cdot \eta$ is an upper bound for the shortest path diameter of G. The pseudocode of that algorithm is given in Algorithm 3.

Algorithm 3. COMPUTE UPPER BOUND FOR SP-DIAMETER

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

$\overline{diam}(G) := \infty$; $\tau := 0$;
forall $i = 1, \ldots, l$ **do**
    $s :=$ choose node uniformly at random
    grow shortest paths tree rooted at s
    grow shortest paths tree rooted at s on the reverted graph
    $\overline{diam}(G) := \min\{\overline{diam}(G), \max_{v \in V}\{dist(s, v)\} + max_{v \in V}\{dist(v, s)\}\}$
forall $s \in V$ **do**
    $T :=$ grow partial shortest paths tree rooted at s.
        stop growing when first node with $dist(s, v) \geq \overline{diam}(G)/\eta$ is visited.
    $\tau :=$ max$\{\tau,$ maximal number of edges of a path in $T\}$
output $\tau \cdot \eta$

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 street network of Luxembourg provided by the PTV AG [1] consists of 30733 nodes and has a shortest path diameter of only 429.

6 Conclusion

In this work we studied two problems. The SHORTCUT PROBLEM (SP) is the problem of how to add a given number of edges to a weighted graph, such that distances do not change and the average number of hops on hop minimal shortest paths in the graph becomes minimal. The REVERSE SHORTCUT PROBLEM (RSP) is the variant of SP where the desired decrease in the average number of hops is fixed and the number of inserted edges has to be minimized. We want to stress out, that this is the first approach towards a theoretical foundation for inserting shortcuts, which is heuristically used by many speed-up techniques for DIJKSTRA's algorithm.

We proved that both problems are *NP*-hard and that there is no polynomial time constant factor approximation algorithm for RSP, unless $P = NP$. Furthermore, no polynomial time algorithm exists that approximates SP up to an additive constant, unless $P = NP$ and that problem is solvable in polynomial time if the number of shortcuts to insert is bounded.

Moreover, we gave two polynomial time approximation algorithms for SP that work for the case that shortest paths on the underlying graph are unique. Finally, we proposed a stochastical method to evaluate the measure function of SP very fast. This can be used for large input networks where an exact evaluation is prohibitive.

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

From the practical point of view, it is important to develop heuristics that find good shortcuts for real-world input. In particular, evolutionary algorithms and local search algorithms (similar to the greedy strategy) seem to be promising. The output of these algorithms should be experimentally tested on their benefit for different speed-up techniques. Further, it is interesting to evaluate the output of the currently used shortcut insertion strategies in the problem's measure function.

References

1. PTV AG - Planung Transport Verkehr (1979), `http://www.ptv.de`
2. Bauer, R., D'Angelo, G., Delling, D., Wagner, D.: The Shortcut Problem – Complexity and Approximation (Full Version), `http://i11www.iti.uni-karlsruhe.de/members/rbauer/pdf/bddw-SOFSEM09_FULL.pdf`
3. Bauer, R., Delling, D.: SHARC: Fast and Robust Unidirectional Routing. In: Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX 2008), pp. 13–26. SIAM, Philadelphia (2008)
4. Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., Wagner, D.: Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 303–318. Springer, Heidelberg (2008)
5. Bruera, F., Cicerone, S., D'Angelo, G., Stefano, G.D., Frigioni, D.: Dynamic Multi-level Overlay Graphs for Shortest Paths. Mathematics in Computer Science 1(4) (2008)
6. Demetrescu, C., Goldberg, A.V., Johnson, D.S. (eds.): 9th DIMACS Implementation Challenge - Shortest Paths (November 2006)

7. Geisberger, R., Sanders, P., Schultes, D., Delling, D.: Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 319–333. Springer, Heidelberg (2008)
8. Goldberg, A.V., Kaplan, H., Werneck, R.F.: Reach for A*: Efficient Point-to-Point Shortest Path Algorithms. In: Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX 2006), pp. 129–143. SIAM, Philadelphia (2006)
9. Goldberg, A.V., Kaplan, H., Werneck, R.F.: Better Landmarks Within Reach. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 38–51. Springer, Heidelberg (2007)
10. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 713–721 (1963)
11. Holzer, M., Schulz, F., Wagner, D.: Engineering Multi-Level Overlay Graphs for Shortest-Path Queries. In: Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX 2006). SIAM, Philadelphia (2006)
12. Lauther, U.: An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background. In: Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung, vol. 22, pp. 219–230. IfGI prints (2004)
13. Sanders, P., Schultes, D.: Engineering Highway Hierarchies. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 804–816. Springer, Heidelberg (2006)
14. Sanders, P., Schultes, D.: Engineering Fast Route Planning Algorithms. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 23–36. Springer, Heidelberg (2007)
15. Schultes, D., Sanders, P.: Dynamic Highway-Node Routing. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 66–79. Springer, Heidelberg (2007)
16. Schulz, F., Wagner, D., Weihe, K.: Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. ACM Journal of Experimental Algorithmics 5 (2000)
17. Schulz, F., Wagner, D., Zaroliagis, C.: Using Multi-Level Graphs for Timetable Information in Railway Systems. In: Mount, D.M., Stein, C. (eds.) ALENEX 2002. LNCS, vol. 2409, pp. 43–59. Springer, Heidelberg (2002)
18. Wagner, D., Willhalm, T.: Speed-Up Techniques for Shortest-Path Computations. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 23–36. Springer, Heidelberg (2007)