Computing Constrained Shortest-Paths at Scale

Alberto Vera, Siddhartha Banerjee, Samitha Samaranayake
Cornell University

Motivated by the needs of modern transportation service platforms, we study the problem of computing constrained shortest paths (CSP) at scale via preprocessing techniques. Our work makes two contributions in this regard:
1. We propose a scalable algorithm for CSP queries, and show how its performance can be parametrized in terms of a new network primitive, the *constrained highway dimension*. This development extends recent work which established the *highway dimension* as the appropriate primitive for characterizing the performance of unconstrained shortest-path (SP) algorithms. Our main theoretical contribution is deriving conditions relating the two notions, thereby providing a characterization of networks where CSP and SP queries are of comparable hardness.
2. We develop practical algorithms for scalable CSP computation, augmenting our theory with additional network clustering heuristics. We evaluate these algorithms on real-world datasets to validate our theoretical findings. Our techniques are orders of magnitude faster than existing approaches, while requiring only limited additional storage and preprocessing.

Key words: Constrained Shortest Path; Distance Oracles; Highway Dimension.

1. Introduction

Motivated by the requirements of modern transportation systems, we consider the fast computation of constrained shortest paths (CSP) in large-scale graphs. Though the unconstrained shortest-path (SP) problem has a long history, it has been revolutionized by recent algorithmic advancements that help enable large-scale mapping applications (cf. (Bast et al. 2016, Demetrescu et al. 2009) for surveys). In particular, the use of preprocessing techniques and network augmentation has led to dramatic improvements in the scalability of SP computation for road networks. These techniques, however, do not extend to the CSP, and hence can not fully leverage the rich travel-time distribution data available today.

The SP and CSP problems can be summarized as follows: We are given a graph G, where each edge has an associated *length* and *cost*. The SP problem requires finding an (s,t)-path of minimum length for any given nodes s and t. The CSP problem inputs an additional budget b, and requires finding a minimum length (s,t)-path *with total cost at most* b. The two problems, though similar, have very different runtime complexity: SP admits polynomial-time algorithms (in particular, the famous Dijkstra's algorithm), while CSP is known to be NP-Hard (Festa 2015). That said, a standard dynamic program computes CSPs in pseudo-polynomial time for discrete costs (Clawson et al. 2015), and gives a natural scaling-based FPTAS for continuous costs (as in the knapsack problem).

Though there is a rich literature on CSP (Festa 2015), existing approaches do not scale to support modern applications. To this end, we study *preprocessing and network augmentation for speeding up CSP computation*. Our work contributes to a growing field of algorithms for large-scale problems (non-convex methods, sketching techniques, etc.), with relatively poor worst-case performance, but which are provably efficient for practically relevant settings.

Applications of large-scale CSP computation: Our primary motivation for scaling CSP comes from the requirements of modern transportation platforms (Lyft, Uber, etc.) for accurate routing and travel-time estimates. Modern SP engines like Google Maps do not make full use of available traffic information. In particular, they do not incorporate uncertainties in travel times, leading to inaccurate estimates in settings with high traffic variability. This can be addressed by computing shortest paths based on *probabilistic* travel-time estimates: if ℓ is the (possibly random) edge length function, given vertices s,t, tolerance δ and failure probability p, we want to find an (s,t)-path P minimizing $\mathbb{E}[\ell(P)]$, subject to $\mathbb{P}(\ell(P) > \mathbb{E}[\ell(P)] + \delta) \leq p$. Computing this exactly for general distributions is expensive due to the need for computing convolutions of distributions – indeed, there is no known polynomial time algorithm for doing this even with black-box access to convolution solvers (Nikolova et al. 2006). However, conditioning on public state variables (e.g. weather and traffic in key neighborhoods), we can approximate the distributions with uncorrelated travel-times across different road segments (Woodard et al. 2017). Thus, Chebyshev's inequality gives us that $\mathbb{P}(\sum_e \ell_e > \mathbb{E}[\sum_e \ell_e] + \delta) \leq \sum_e \mathbb{V}\text{ar}(\ell_e)/\delta^2$. Using this, we can reformulate the stochastic path optimization problem as $\min_{P\in\mathcal{P}_{s,t}} \sum_{i\in P} \mu_i$ s.t. $\sum_{i\in P} \sigma_i^2 \leq \delta^2 p$, which is now a CSP problem (see also (Nikolova et al. 2006) for a similar approach for Gaussian travel times). Note that, though we relax the condition $\mathbb{P}(\ell(P) > \mathbb{E}[\ell(P)] + \delta) \leq p$, our solution always respects this constraint – this is often critical for practical applications, e.g., for Lyft/Uber, accuracy of ETA estimates is more important than choosing the optimal route.

Another problem that can be modeled as a CSP is that of finding *reliable shortest paths*. Consider the case where each edge has a probability q_e of triggering a bad event, with resulting penalty p (for example, slowdowns due to accidents). In this case, we want to minimize the travel time as well as the expected penalty. Assuming independence, we have the following natural problem: $\min_{P\in\mathcal{P}_{s,t}} \ell(P) + p\big(1 - \prod_{e\in P}(1-q_e)\big)$. This model is considered in (Correa et al. 2017) for routing with fare evasion, where q_e is the probability of encountering an inspector, and p the penalty; the authors suggest using a CSP formulation, wherein the non-linear objective is replaced by a linear constraint by taking logarithms.

1.1. Our Contributions

We consider the problem of developing oracles that support fast CSP queries in large networks. Specifically, given integer edge-costs, edge-lengths, and a budget upper bound, we use preprocessing to create a data-structure supporting arbitrary source-destination-budget queries; moreover, we obtain formal guarantees on the performance: preprocessing, storage, and query time.

In the case of SP algorithms, the seminal work of Abraham et al. (Abraham et al. 2016, 2010) demonstrated that the preprocessing, storage, and query times of several widely-used heuristics could be parametrized using a graph structural metric called the *Highway Dimension* (HD). Our work extends these notions for the CSP; our contributions are summarized as follows:

Theoretical contributions: We define the *constrained highway dimension* (CHD) for the set of *efficient paths* (i.e., minimal solutions to the CSP; cf. Definition 1). We show how the CHD can be used to parametrize the performance of CSP algorithms. One hurdle, however, is that the CHD can be much bigger than the HD in general. Our main theoretical contribution is in showing that the *HD and CHD can be related under an additional partial witness condition* (Definition 7); therefore, in settings where SP computation is scalable, we can also solve the harder problem of CSP. Our next contribution is to show that, under *average performance metrics*, we can obtain even weaker conditions to relate the *average CHD and HD*, thus certifying the performance of our algorithms. The conditions can be interpreted in terms of having few physical overpasses in road networks.

Practical contributions: We use our theoretical results to develop new practical data-structures for CSP queries, based on *hub labels* (Cohen et al. 2003). We evaluate our algorithm on datasets with detailed travel-time information for San Francisco and Luxembourg. In experiments, our algorithms exhibit query-times 10^4 faster than existing (non-preprocessing) techniques, have small storage requirements, and good preprocessing times even on a single machine.

Paper outline: In Section 2.1, we introduce the SP and CSP problems, and extend the notion of the HD (as defined in (Abraham et al. 2016)) to directed graphs and general path systems; this allows us to define an analogous notion of a *constrained highway dimension* (CHD) for constrained shortest paths in Section 3. We then show that the two can be related under an additional *partial witness condition* (Section 3.3). In Section 3.4, we study average-case performance, and show how a small average CHD can be related to physical overpasses in road networks. Finally, in Section 4, we present our practical hub-label construction and our experiments on SF and Luxembourg data.

1.2. Related work

CSP problems have an extensive literature, surveyed in (Festa 2015). More recently, there has been significant interest in stochastic SP problems, including the related stochastic on-time arrival

(SOTA) problem (Fan et al. 2005); recent works have proposed both optimal and approximate policies (Sabran et al. 2014, Hoy and Nikolova 2015). Existing approaches for these problems, however, are limited in their use of preprocessing and augmentation techniques, and consequently do not support the latencies required for mapping applications.

As we mentioned before, our work is inspired by the recent developments in shortest path algorithms (Abraham et al. 2016, 2011b, 2010, Demetrescu et al. 2009, Geisberger et al. 2008, Kosowski and Viennot 2017); refer (Bast et al. 2016) for an excellent survey of these developments. The pre-processing technique we use for speeding up CSP computations is hub labels (HL), first introduced for SP computations in (Cohen et al. 2003). More recently, HL was proved to have the best query-time bounds for SP computation in low HD graphs (Abraham et al. 2016, 2010) (this was experimentally confirmed in (Abraham et al. 2011b), (Bast et al. 2016, Figure 7)). The notion of HD has been widely applied and therefore there are several variants (definitions) of this concept, we refer to Appendix A and also (Feldmann et al. 2018, Section 9) for a more detailed discussion. Finally, the HD-based bound for hub labels was shown to be tight in (Babenko et al. 2015, White 2015), and it was also shown that finding optimal hub labels is NP hard.

A related class of problems to CSP is that of SP under label constraints (Barrett et al. 2008), where the aim is to find shortest paths that avoid certain labels (e.g. toll roads, ferries, etc.). In this setting, there is work on using preprocessing to improve query-times (Rice and Tsotras 2010). These problems are essentially concatenations of parallel SP problems, involving only local constraints. In contrast, the CSP involves global constraints on paths. Our results do in fact shed light on why preprocessing works well for label-constrained SP queries.

Finally, we note that the notion of Highway Dimension has been successfully applied to parameterize the complexity of other NP hard problems. Graphs with bounded HD admit a probabilistic embedding into graphs with bounded tree-width (Feldmann et al. 2018), this implies that several NP hard problems have quasi-polynomial approximation schemes in the context of transportation networks. Subsequently, (Becker et al. 2018) introduced new ideas to get an embedding that allows for polynomial time approximation schemes (as opposed to quasi-polynomial) for several problems when the HD is bounded. Let us consider an important problem, the k-center problem, arising in logistics and other domains. The k-center problem requires to find k center vertices such that the distance of every node to the closest center is minimized. This problem is known to be hard, but it surprisingly remains hard even if we parameterize the complexity jointly by k, the HD, and the tree-width (Feldmann and Marx 2018). On the other hand, despite its parameterized hardness, there are approximation algorithms when k and the HD are combined (Feldmann 2019). Two fundamental problems are TSP (traveling salesman) and STP (Steiner tree). (Disser et al. 2019) gives improved results for TSP and STP when the HD is 1, obtaining a FPTAS and, on the negative side, they also show a hardness result when the HD is 6 or larger.

2. Setting and Overview

2.1. Problem Statement

W e consider a directed graph $G = (V, E)$, where each edge $e \in E$ has an associated *length* $\ell(e) \in \mathbb{N}_+$, and *cost* $c(e) \in \mathbb{N}_+ \cup \{0\}$. The triplet (G, ℓ, c) is called a *network*. For any path P, we define its length $\ell(P)$ and cost $c(P)$ as the sum of edge lengths and edge costs in P, respectively. Our goal is to develop a data structure to answer *Constrained Shortest-Path* (CSP) queries:

Constrained Shortest-Path Queries

Input:	Graph $G = (V, E)$, costs c, lengths ℓ, and maximum budget $B \in \mathbb{N}$.
Preprocessing Task:	Create a data structure $\mathcal{S}$.
Query Task:	For any source-terminal pair $s, t \in V$ and budget $b \leq B$, use $\mathcal{S}$ to return a path solving $\min\{\ell(P) : c(P) \leq b, P \text{ is an } (s,t)\text{-path}\}$. The triplet (s, t, b) defines a *query* and it is arbitrarily specified by the user.
Performance Metrics:	Size of $\mathcal{S}$, preprocessing time (to compute $\mathcal{S}$), and query time (to return a path for given s, t, b).

Note that the two computation times (preprocessing and query) are in different scales. Indeed, it is acceptable to have a preprocessing time of a few minutes, but the query time should be in milliseconds. Observe also that the performance metrics are in conflict. To illustrate this, define $n = |V|$ as the number of nodes and consider two extreme cases. (i) for each (s, t, b) we store in $\mathcal{S}$ the solution to the associated query, which has $O(1)$ query time, but space $|\mathcal{S}| = \Omega(Bn^2)$. (ii) use no data structure, which consumes no space, but requires $\Omega(bn \log n)$ query time using Dijkstra (see Proposition 3).

We stress that either storage $\Omega(n^2)$ or query time $\Omega(n)$ is unacceptable for modern applications. The goal is to find $\mathcal{S}$ in polynomial time with small storage and fast query time. Additionally, note that we require the actual path and not just the distance; prior work calls this a path-reporting oracle, see (Elkin and Pettie 2016) for a discussion.

Summary of our Results. We identify a parameter h_c, called the Constrained Highway Dimension (CHD), which generalizes the concept introduced in (Abraham et al. 2016), and obtain the results below. In each case, the data structure can be computed in polynomial time. D represents the diameter of the graph.

1. Parameterization via h_c: We give a data structure of size $\tilde{O}(nB \cdot Bh_c \log D)$ and query time $\tilde{O}(bh_c \log D)$ for any (s, t, b), see Theorems 1 and 2. We relate h_c to the HD in Theorem 3.
2. Average Case Metrics: We introduce the notion of *Average CHD*, which is strictly weaker than CHD, and show that we can obtain a data structure with the same guarantees, but the query time is in average over (s, t), see Theorem 4.

3. Use Average Case to Solve CSP: we give a condition, interpreted as having few overpasses, such that the query time is $\tilde{O}(bh\alpha \log D)$, average over (s,t), with space requirement $\tilde{O}(nB \cdot Bh\alpha \log D)$, where h is the HD and α the doubling constant, see Theorem 5. We observe that, for road networks, it is conjectured that $h = \text{polylog}(n)$ and $\alpha = O(1)$, hence our result explains the good empirical performance of our algorithm, see Section 4.

2.2. Preliminaries: Hitting Sets and the Highway Dimension

We start by defining notation we will use throughout. For any source-terminal pair $s,t \in V$, we denote by $\mathcal{P}_{s,t}$ the set of all simple (s,t)-paths (without loops or cycles). Throughout this work, we only consider simple paths, which we refer to as paths for brevity. We denote the shortest (s,t)-path (if it exists) as $P(s,t)$, and denote the set of all shortest paths in G as $\mathcal{P}^*$.

For $s,t \in V$, the distance from s to t, denoted $\text{dist}(s,t)$, is the smallest length among all paths $P \in \mathcal{P}_{s,t}$. We define $\text{dist}(s,t|b)$ to be the length of the shortest path with cost at most b, i.e., the length of the path returned by the query (s,t,b). If there is no feasible solution, we define $\text{dist}(s,t|b) = \infty$.

For a node v and a path P, we abuse notation to denote $\text{dist}(v,P)$ as the minimum distance from v to any node $w \in P$; the distance $\text{dist}(P,v)$ from P to v is defined analogously. Note that $\text{dist}(P,v)$ and $\text{dist}(v,P)$ need not be the same as the graph is directed. We define $D := \max_{P \in \mathcal{P}^*} \ell(P)$ to be the diameter of G.

For $r > 0$ and $v \in V$, we define the *forward and reverse balls of radius* r by $B_r^+(v) := \{u \in V : \text{dist}(v,u) \le r\}$ and $B_r^-(v) := \{u \in V : \text{dist}(u,v) \le r\}$, and also define $B_r(v) := B_r^+(v) \cup B_r^-(v)$. Finally, a graph G is said to have a *doubling constant* α if, for any node v and any $r > 0$, the ball $B_{2r}(v)$ can be covered by at most α balls of radius r.

Generalizing Highway Dimension. We give a generalization of the notion of *highway dimension*, introduced by (Abraham et al. 2016, 2010) to parametrize shortest-path computations in undirected graphs. The technical challenges of these extensions may not be clear to a non-expert reader, thus we differ all discussions on the matter to Appendix A. Very broadly speaking, (Abraham et al. 2016) deals only with undirected graphs and shortest paths, whereas our approach covers directed graphs and general sets of paths. In particular, if we restrict the set of paths accordingly, we recover the original notion.

To motivate general sets of paths, note that the CSP problem may have multiple solutions as there could be several paths with the same length and cost lower than b. To limit these solutions to those with minimal cost, we require that the path also be *efficient*.

Definition 1 (Efficient Path). A path $P \in \mathcal{P}_{s,t}$ is called *efficient* if there is no other path $P' \in \mathcal{P}_{s,t}$ such that $\ell(P') \le \ell(P)$ and $c(P') \le c(P)$ with at least one inequality strict. We denote the set of all efficient paths as $\mathcal{P}^E$.

Definition 2 (Path System). Recall that $G = (V, E)$ is fixed throughout. We define a *path system* $\mathcal{Q}$ as any collection of paths in G.

Our goal is to define the HD of any path system, then in Section 2.3 show how to answer queries parametrized it. We will use mainly two path systems, $\mathcal{P}^*$ (all shortest paths) and $\mathcal{P}^E$ (efficient paths), but note that our notion is general and encompasses other restrictions such as label constraints.

We say that a set $C \subseteq V$ *hits* any given path Q if some node in Q belongs to C. Moreover, we say that C is a *hitting set for a path system* $\mathcal{Q}$ if it hits every $Q \in \mathcal{Q}$. *Hitting sets are fundamental and they drive all the analysis and design of our algorithms.* A crucial intuitive concept we use is that of *compression.* As we discussed at the beginning of Section 2.1, if we want query time better than Dijkstra, i.e., $O(n \log n)$, a priori we need space $\Omega(n^2)$ to store all the shortest paths. We say that a data structure $\mathcal{S}$ compresses CSP if it has size $o(n^2)$ and query time $o(n \log n)$. The holy grail is a compression of space $O(n\text{polylog}(n))$ with query time $O(\text{polylog}(n))$. Hitting sets are fundamental because, as we will prove in Proposition 2, they allow to compress path systems.

For any $r > 0$, we say a path Q is r-significant if $\ell(Q) > r$. For a given path system $\mathcal{Q}$, we denote $\mathcal{Q}_r$ as the set of all r-significant paths in $\mathcal{Q}$. In particular, even if the hitting set is large, the extent to which a path system can be compressed depends on the *local sparsity* of hitting sets with respect to *significant paths* of $\mathcal{Q}$.

Definition 3 (Locally-Sparse Hitting Sets). Given a path system $\mathcal{Q}$ and $r > 0$, an (h, r) locally-sparse hitting set (or (h, r)-LSHS) is a set $C \subseteq V$ with two properties:

1. Hitting: C is a hitting set for $\mathcal{Q}_r$.
2. Local sparsity: for every $v \in V$, $|B_{2r}(v) \cap C| \leq h$.

Remark 1. As we discuss in Section 2.3, the existence of (h, r)-LSHS immediately enables the compression of path system $\mathcal{Q}$ via the construction of *hub labels*. However, the existence of LSHS does not guarantee the ability to efficiently compute these objects.

To address this computability issue, we need a stronger notion; the *highway dimension* is a property that ensures both existence and efficient computation of LSHS. To define the highway dimension (HD), we first need two additional definitions: for $v \in V, r > 0$, the *forward path-neighborhood* with respect to a path system $\mathcal{Q}$ is $S_r^+(v, \mathcal{Q}) := \{Q \in \mathcal{Q}_r : \text{dist}(v, Q) \leq 2r\}$ and similarly $S_r^-(v, \mathcal{Q}) := \{Q \in \mathcal{Q}_r : \text{dist}(Q, v) \leq 2r\}$ is the reverse neighborhood. As before, $S_r(v, \mathcal{Q}) := S_r^+(v, \mathcal{Q}) \cup S_r^-(v, \mathcal{Q})$. Now we can define the HD of a path system $\mathcal{Q}$. Essentially, the HD re-orders the sequence of qualifiers in the definition of (h, r)-LSHS: it requires the existence of a small hitting set for each individual neighborhood, rather than a single hitting set which is locally sparse.

Definition 4 (Highway Dimension). A path system $\mathcal{Q}$ has HD h if, $\forall r > 0, v \in V$, there exists a set $H_{v,r} \subseteq V$ such that $|H_{v,r}| \leq h$ and $H_{v,r}$ is a hitting set for $S_r(v, \mathcal{Q})$.

As shorthand, we refer to the HD of (G, ℓ) as that of $\mathcal{P}^*$. Note that $HD \leq h$ is a more stringent requirement than the existence of an (h,r)-LSHS C, since $C \cap B_{2r}(v)$ need not hit all the paths in $S_r(v, \mathcal{Q})$. However, if G has $HD \leq h$, then this guarantees the existence of a (h,r)-LSHS according to the following result, which can be proven by adapting the proof from (Abraham et al. 2016, Theorem 4.2) to our general case.

Proposition 1. *If the path system $\mathcal{Q}$ has HD h, then, $\forall\, r > 0$, there exists an (h,r)-LSHS.*

More importantly, note that the result is about existence and does not touch on computability. As we discuss in Section 3.2.2, if G has $HD \leq h$, then this permits efficient computation of LSHS.

2.3. Preliminaries: Shortest-Paths via Hub Labels

Two of the most successful data-structures enabling fast shortest path queries at scale are *contraction hierarchies* (CH) (Geisberger et al. 2008) and *hub labels* (HL) (Cohen et al. 2003). These are general techniques which always guarantee correct SP computation, but have no uniform storage/query-time bounds for all graphs. We now explain the construction for HL; for the construction and results of CH refer to Appendix B.

The basic HL technique for SP computations is as follows: Every node v is associated with a hub label $L(v) = \{L^+(v), L^-(v)\}$, comprising of a set of forward hubs $L^+(v) \subseteq V$ and reverse hubs $L^-(v) \subseteq V$. We also store $\text{dist}(v,w)\, \forall\, w \in L^+(v)$ and $\text{dist}(u,v)\, \forall\, u \in L^-(v)$. Hub Labels must satisfy the *cover property* defined as follows: for any $s \neq t \in V$, $L^+(s) \cap L^-(t)$ contains at least one node in $P(s,t)$. In the case that t is not reachable from s, it must be that $L^+(s) \cap L^-(t) = \varnothing$.

With the aid of the cover property, we can obtain $\text{dist}(s,t)$ by searching for the minimum value of $\text{dist}(s,w) + \text{dist}(w,t)$ over all nodes $w \in L^+(s) \cap L^-(t)$. If the hubs are sorted by ID, this can be done in time $O(|L^+(s)| + |L^-(t)|)$ via a single sweep. Moreover, by storing the second node in $P(s,w)$ for each $w \in L^+(s)$, and the penultimate node in $P(w,t)$ for each $w \in L^-(t)$, we can also recover the shortest path recursively, as each HL query returns at least one new node $w \in P(s,t)$. Note that we need to store this extra information, otherwise we could have $L^+(s) \cap L^-(t) = \{s\}$. Let $L_{\max} := \max_v |L^+(v)| + \max_v |L^-(v)|$ be the size of the maximum HL. The per-node storage requirement is $O(L_{\max})$, while the query time is $O(L_{\max} \ell(P(s,t)))$.

Although hub labels always exist (in particular, we can always choose $L^+(s)$ to be the set of nodes reachable from v, and $L^-(s)$ the set of nodes that can reach v), finding *optimal* hub-labels (in terms of storage/query-time bounds) is known to be NP-hard (Babenko et al. 2015). To construct hub labels with guarantees on preprocessing time and $L_{\max}$, we need the additional notion of a *multi-scale LSHS*. We assume that (G,ℓ) admits a collection of sets $\{C_i : i=1,\dots,\log D\}$, such that each C_i is an $(h,2^{i-1})$-LSHS. Given such a collection, we can now obtain small HL. We outline this construction for directed graphs, closely following the construction in (Abraham et al. 2016, Theorem 5.1) for the undirected case.

Proposition 2. *For (G,ℓ), given a multi-scale LSHS collection $\{C_i : i=0,\dots,\log D\}$, where each C_i is an $(h,2^{i-1})$-LSHS, we can construct hub labels of size at most $h(1+\log D)$.*

Proof. For each node v, we define the hub label $L(v)$ as

$$L^+(v) := \bigcup_{i=0}^{\log D} C_i \cap B^+_{2^i}(v) \quad \text{and} \quad L^-(v) := \bigcup_{i=0}^{\log D} C_i \cap B^-_{2^i}(v).$$

Since each C_i is an $(h,2^{i-1})$-LSHS which we intersect with balls of radius $2\cdot 2^{i-1}$, every set in the union contributes at most h elements and the maximum size is as claimed.

To prove the cover property, we note that, if t is not reachable from s, by definition $L^+(s)\cap L^-(t)=\varnothing$. This is because the elements in $L^+(s)$ are reachable from s and the elements in $L^-(t)$ reach t. On the other hand, when $P(s,t)$ exists, we do a case analysis on $\ell(P)$ to prove the cover property. Let i be such that $2^{i-1} < \ell(P(s,t)) \le 2^i$. Finally, any point in the path belongs to both $B^+_{2^i}(s)$ and $B^-_{2^i}(t)$, and hence $C_i \cap P(s,t)$ is in both hubs (which is not empty since C_i hits all SPs of length $\ge 2^{i-1}$). $\square$

Finally, we need to compute the desired multi-scale LSHS in polynomial time. In Section 3.2.2 we show that, if the HD is h, in polynomial time we can obtain sparsity $h' = O(h\log(h))$. In other words, the HL have size $h'(1+\log D)$ instead of $h(1+\log D)$ if we are not given the multi-scale LSHS and have to compute them in polynomial time. A more subtle point is that the resulting algorithm, even though polynomial, is impractical for large networks. In Section 4, we discuss heuristics that work better in practice.

3. Scalable CSP Algorithms: Theoretical Guarantees

We develop a data-structure that supports fast queries for *efficient paths*. Specifically, given a graph $G=(V,E)$ and a maximum budget B, we construct a data-structure such that, for any $s,t\in V$ and $b\le B$, we return the length of the shortest (s,t)-path with cost at most b, denoted $\mathrm{dist}(s,t|b)$. The actual path can easily be recovered as discussed in Section 2.3, hence we focus on querying $\mathrm{dist}(s,t|b)$ only.

In Section 2.3 we discussed that, if a graph G has HD h, we can simultaneously bound, as functions of h, the preprocessing time, storage requirements, and query time for hub labels. This suggests that, for the construction of provably efficient hub labels for the CSP problem, we need an analogous property for the set of *efficient paths*.

Definition 5 (Constrained Highway Dimension). The constrained highway dimension (CHD) of (G, ℓ, c), denoted h_c, is the HD of the efficient-path system $\mathcal{P}^E$.

Note that, since every shortest path is efficient, $h_c \geq h$.

We now have two main issues with this definition: first, it is unclear how this can be used to get hub labels, and second, it is unclear how the corresponding hub labels compare with those for shortest-path computations. To address this, we first convert efficient paths in G to shortest paths in a larger *augmented graph*. In Section 3.2, we use this to construct hub labels for CSP queries whose storage and query complexity can be bounded as Bh_c (which can be strengthened further to $g(b)h_c$, where $g(b)$ measures the size of the Pareto frontier, cf. Section 3.2.3.). Finally, in Section 3.3, we show that the hub labels for CSP queries can in fact be related to the hub labels for SP queries under an additional natural condition on the efficient paths.

3.1. Augmented Graph

In order to link the constrained highway dimension to hub labels, we first convert the original graph G (with length and cost functions) into an *augmented graph* G^B with only edge lengths, such that the *efficient paths of* G *are in bijection with the shortest paths of* G^B. We achieve this as follows: Each node in G^B is of the form $\langle v, b\rangle$, which encodes the information of the remaining budget $b \geq 0$ and location $v \in V$. A node is connected to neighbors (according to E) as long as the remaining budget of that transition is non-negative. Finally, we create n sink nodes, denoted v^-, and connect node $\langle v, b\rangle$ to v^- with length $1/(b+1)$. An illustration of the construction is presented in Figure 1. The following definition formalizes this.

Definition 6 (Augmented Graph). Given (G, ℓ, c) and $B \in \mathbb{N}$, the augmented version G^B has vertex set $V^B := \{\langle v, b\rangle : v \in V, b = 0, 1, \ldots, B\} \cup \{v^- : v \in V\}$, the edge set E^B is $\{\langle v, b\rangle\langle w, x\rangle : vw \in E, x = b - c_{vw}, x \geq 0\} \cup \{\langle v, b\rangle v^- : v \in V, 0 \leq b \leq B\}$. The length function in G^B is $\ell(\langle v, b\rangle, v^-) := \frac{1}{b+1}$ and $\ell(\langle v, b\rangle, \langle w, x\rangle) := \ell(vw)$.

Paths in G^B are mapped to paths in G in the intuitive way, by removing the budget labels and sink nodes. We call this mapping the *projection* of a path. The next result, proved in Appendix D, shows that efficient paths are in correspondence with projections.

Proposition 3. *A shortest path from source* $\langle s, b\rangle$ *to sink node* t^- *projects to an efficient path in* G *solving* $dist(s, t|b)$.

Later we give a construction that depends on LSHS for the system $\mathcal{P}^E$, we call this object an efficient path hitting set (EPHS). The next result allows us to relate EPHS of the original graph to LSHS in the augmented graph. Note that, in G^B, we are interested only in shortest paths ending in

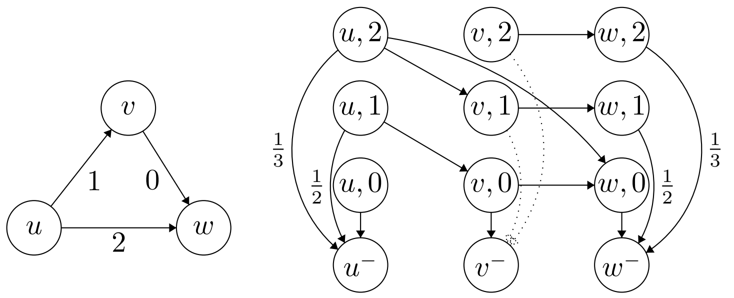


Figure 1 Example of a graph augmentation: The original graph G has all paths of unit length, and costs as indicated on the edges. In the augmented graph G^B, the labels represent the edge lengths (unlabeled edges have length 1). Note the additional edges from (w, b) to the sink node w^- (and similarly for u and v).

sink nodes (since these map to efficient paths). Let $\mathcal{P}^B$ be the path system comprising all shortest paths in G^B ending in a sink node. A hitting set for $\mathcal{P}^E$ (in G) can be used to obtain a hitting set for $\mathcal{P}^B$ (in G^B), but, since the augmented graph has more information, the sparsity increases.

PROPOSITION 4. *Given a (h_c, r)-EPHS for the path system $\mathcal{P}^E$, in polynomial time we can construct a (h_cB, r)-LSHS for $\mathcal{P}^B$.*

PROOF. Given C, an (h_c, r)-EPHS for $\mathcal{P}^E$, define

$$C^B := \{\langle v, b\rangle : v \in C, v \text{ hits } \bar{P} \in \mathcal{P}^B_r, c(\bar{P}) = b \le B\}. \tag{1}$$

We prove that C^B hits $\mathcal{P}^B_r$ and is locally sparse. By Proposition 3, we know that shortest paths are efficient, hence C^B hits all the desired paths. Finally, we prove local sparsity. Take any node $\langle s, b\rangle$ and observe that

$$B^+_{2r}(\langle s, b\rangle) = \{\langle t, x\rangle : \exists P \in \mathcal{P}_{s,t}, \ell(P) \le 2r, c(P) = b - x\} \subseteq \{\langle t, x\rangle : t \in B^+_{2r}(s), x \le b\}. \tag{2}$$

We know that $|B^+_{2r}(s) \cap C| \le h_c$, therefore $|B^+_{2r}(\langle s, b\rangle) \cap C^B| \le h_c b \le h_c B$. A similar argument shows the sparsity for the reverse ball. □

The proof above shows a stronger result: In Eq. (2) we see that the sparsity around the node $\langle u, b\rangle$ is $h_c b$. This is key for our subsequent query time guarantees.

Surprisingly, in this case we can also relate the HDs of the path systems $\mathcal{P}^E$ and $\mathcal{P}^B$. Note that this does not follow from Proposition 4, since the HD is a stronger notion than existence of locally-sparse hitting sets.

PROPOSITION 5. *If the HD of the system $\mathcal{P}^E$ is h_c, then the HD of the system $\mathcal{P}^B$ is Bh_c.*

PROOF. Fix $r > 0$ and $\langle v, b\rangle \in V^B$. Let $H_{v,r} \subseteq V$ be the set hitting $S_r(v, \mathcal{P}^E)$ and define $H := H_{v,r} \times \{0, 1, \dots, B\}$. We show that H hits $S^+_r(\langle v, b\rangle, \mathcal{P}^B)$.

Take $P \in S^+_r(\langle v, b\rangle, \mathcal{P}^B)$. Since $\text{dist}(\langle v, b\rangle, P) \le 2r$, $\text{dist}(v, \bar{P}) \le 2r$, therefore $\bar{P} \in S^+_r(v, \mathcal{P}^E)$. Finally, $H_{v,r}$ hits $\bar{P}$, thus H hits P. A similar argument shows that H hits $S^-_r(\langle v, b\rangle, \mathcal{P}^B)$. □

3.2. Solving CSP via Hub Labels

We present a construction similar to HL for shortest-paths (cf. Section 2.3). A subtle difference is that we are only interested in paths ending in a sink node. Each node $\langle v, b\rangle$ has a forward hub label $L^+(\langle v, b\rangle) \subseteq V^B$, and *only sink nodes* u^- have a reverse hub $L^-(u^-) \subseteq V^B$. The cover property must be satisfied for every $\langle s, b\rangle$ and t^-. Finally, if we want to reconstruct the path, we can proceed similarly as in Section 2.3; we can augment the hub labels with the next-hop node, and compute the entire path recursively. Putting things together, we can construct hub labels for answering CSP queries, with preprocessing time and storage parameterized by the CHD h_c.

3.2.1. Query Time and Data Requirements

Theorem 1. *For a network (G, ℓ, c), given a multi-scale EPHS $\{C_i : i = 0, 1, \dots, \log D\}$, where C_i is an $(h_c, 2^{i-1})$-EPHS, we can construct hub labels to answer queries for s, t, b in time $O((B+1)h_c \log D)$. The total space requirement is $O(nB \cdot Bh_c \log D)$.*

Proof. Create C_i^B as in Eq. (1). Define $L(\langle v, b\rangle)^+ := \bigcup_{i=1}^{\log D} C_i^B \cap B^+_{2^i}(\langle v, b\rangle)$ and $L(u^-)^- := \bigcup_{i=1}^{\log D} C_i^B \cap B^-_{2^i}(u^-)$. The cover property is proved similarly as in Proposition 2; we are left to bound the hub size. For a reverse hub we use that $B^-_{2^i}(t^-) = \{\langle s, x\rangle : \exists P \in \mathcal{P}_{s,t}, c(P) = x, \ell(P) \leq 2^i\} \subseteq B^-_{2^i}(t) \times \{0, 1, \dots, B\}$. Thus, $B^-_{2^i}(t^-) \cap C_i^B \leq (B+1)h_c$. For forward hubs, the size follows from observing that $|C_i^B \cap B^+_{2^i}(\langle v, b\rangle)| \leq (b+1)h_c$. □

3.2.2. Preprocessing Computing hitting sets is difficult in general, but it becomes tractable when the underlying set has small VC-dimension (Even et al. 2005). The key idea is that we can perturb the edge lengths slightly so that we can guarantee uniqueness, i.e., every shortest path is unique. It was recently proved that the set system of unique directed shortest paths has VC-dimension 3 (Funke et al. 2014, Theorem 1). We note that perturbing edge lengths is a common in the literature and it does not have practical drawbacks (Abraham et al. 2010, 2016).

Finally, polynomial-time preprocessing now follows the main result in (Even et al. 2005). The desired result is stated in Proposition 6, we present the proof in Appendix D for completeness.

Proposition 6. *If a path system $\mathcal{Q}$ has HD h, then, for any $r > 0$, we can obtain in polynomial time a (h', r)-LSHS, where $h' = O(h \log(h))$.*

3.2.3. Using the size of the Pareto Frontier The linear dependence on B in the bound on HL sizes (cf. Theorem 1) is somewhat weak. Essentially, this corresponds to a worst-case setting where the efficient paths between any pair of nodes is different for each budget level. In most practical settings, changing the budget does not change the paths too much, and ideally the hub label sizes should reflect this fact. This is achieved via a more careful construction of hub labels, resulting in the following bound.

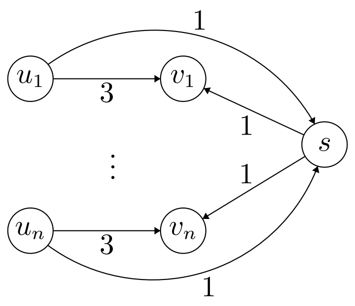


Figure 2 Graph with small HD but large CHD: The graph comprises $2n+1$ nodes, with the edge labels representing the lengths. Note that the shortest paths in the graph are of the form sv_i, u_is and u_isv_j (for all combinations i,j). Thus, the HD is 1 as $H_{v,r}=\{s\}$ is a hitting set for all these paths. On the other hand, if we have costs such that $c(u_iv_i)=0\,\forall\, i$, while all other edges have cost 1, then we have n parallel efficient paths u_iv_i, which must all be hit by any EPHS.

THEOREM 2. *Let (G,ℓ,c) as in Theorem 1 and $g:\mathbb{N}\to\mathbb{N}$ be such that, for every $s,t\in V$, $b\in\mathbb{N}$, $|\{P\in\mathcal{P}^E_{s,t}:c(P)\le b\}|\le g(b)$. Then, we can construct hub labels of size $O(g(B)h_c\log D)$, and answer queries with budget b in time $O(g(b)h_c\log D)$.*

Note that there always exists such a function g and the worst case is $g(b)=b$. The proof depends on a different technique for constructing HL (refer to Algorithms 1 and 2 in Appendix D). The main idea is to sort the efficient paths for each source node s by cost, and then carefully mark nodes when they are added to the forward HL; these marked nodes are then used to construct the reverse HL. For brevity, the complete algorithmic details and proof are deferred to Appendix D.

3.3. Comparing HD and CHD

The previous sections show that we can construct hub labels for solving CSP whose preprocessing time, storage and query time can all be parameterized in terms of the constrained highway dimension h_c. This, however, still does not give a sense of how much worse the hub labels for the CSP problem can be in comparison with those for finding shortest paths. We now try to understand this question. Comparing the size of the *optimal* hub labels for SP and CSP is infeasible as even finding the optimal hub labels for SP is NP-hard (Babenko et al. 2015). However, since we can parametrize the complexity of HL construction for SP in terms of the HD, a natural question is whether graphs with small HD also have a small CHD. Note that the answer to this depends on both the graph and the costs. We now show that the CHD and, moreover, the sparsity of any EPHS, can be *arbitrarily worse* than the HD.

PROPOSITION 7. *There are networks with HD 1 where the CHD is n. Furthermore, the sparsity of an EPHS is also n.*

Proof. Consider the directed graph G defined in Figure 2. It is easy to see that $H_{v,r} = \{s\}$ is a shortest-path hitting set for every $r > 0$ and $v \in V(G)$; hence the HD is 1. On the other hand, suppose the costs are such that $c(u_iv_i) = 0$ for every i, while all other costs are set to 1. Note that the 1-significant efficient paths intersecting the ball $B_s(2)$ are u_iv_i, which are all disjoint. Therefore, the hitting set $H_{s,1}$ must contain at least n elements. This concludes the separation between HD and CHD. Finally, the same argument shows that there is LSHS for $\mathcal{P}^*$ with sparsity 1, whereas the sparsity of *any* EPHS is also lower bounded by n. Thus the sparsities of LSHS and EPHS are also separated. □

Remark 2. One criticism of the graph in Fig. 2 is that it has a maximum degree of n. However, the result holds even for bounded degree graphs. In Appendix A, where we discuss alternative notions of HD, we give a more involved example with bounded degrees that exhibits the same separation between LSHS and EPHS. Another criticism is that the edge u_iv_i is not the shortest (u_i, v_i)-path, but this is for exposition only and the same holds with the following modification: add a node w_i between u_i and v_i, then set $\ell(u_iw_i) = 2$, $\ell(w_iv_i) = 1$, and $c(u_iw_i) = c(w_iv_i) = 0$.

Intuitively, the separation between HD and CHD occurs due to the fact that, for arbitrary graphs and cost functions, the shortest and efficient paths may be completely unrelated. For real-world networks, however, this appears unlikely. In particular, intuition suggests that efficient paths largely comprise of segments which are in fact shortest-paths in their own right. This notion can be formalized via the following definition of a *partial witness*

Definition 7 (Partial Witness). Let $\beta \geq 0$. We say that a path system $\mathcal{Q}$ is β-witnessed by the path system $\mathcal{Q}'$ if, for every $Q \in \mathcal{Q}$, $\exists Q' \in \mathcal{Q}'$ such that $Q' \subseteq Q$ and $\ell(Q') \geq 2^{-\beta}\ell(Q)$.

We use the partial-witness property to provide a link between the CHD and HD, essentially showing that scaling CSP queries is of similar complexity to scaling SP queries as long as any efficient path contains large shortest-path segments, this is formally stated in Theorem 3 below. We illustrate the definition and the intuition behind it in the following example.

Example 1 (Multimodal Network). Consider a *multimodal mapping service* which gives transit routes combining different modes of transportation, e.g., walking, bus, subway, trolley, etc. As a desirable feature, we ensure that routes have at most k transfers. We claim that, in this network, the partial witness property is satisfied with $\beta = \log_2(k+1)$. Indeed, with at most k transfers there are at most $k+1$ segments, hence one of them must be larger than a fraction $\frac{1}{k+1}$. Finally, we remark that, if each individual network has small HD, then the CHD is also small (cf. Proposition 14).

We can now ask if the hub labels for computing SPs and CSPs can be related in settings where the shortest-path system $\mathcal{P}^*$ is a partial witness for the efficient path system $\mathcal{P}^E$. At an intuitive level, the partial witness property says that efficient and shortest paths are not completely different, i.e., if Q is efficient, a fraction $2^{-\beta}$ of Q is a shortest path. As a consequence, a node hitting numerous paths in $\mathcal{P}^*$, should also hit many paths in $\mathcal{P}^E$. Note that asking for the witness property to hold for all lengths is too extreme, as this essentially requires that all single-hop paths with 0 costs are shortest paths. Thus, we want this property only for 'long-enough' paths.

We now show that if, for some β, the network indeed has the partial witness property for paths longer than some r_β, then we can relate the HL sizes for the two problems in terms of β and the doubling constant α. Note that the doubling constant depends on G and ℓ; the partial witness property depends on the interplay between G, c and ℓ. Observe also that, if α is a constant, then the requirement in Theorem 3 is for paths longer than $r_\beta \sim h\alpha^{\beta-2}$.

THEOREM 3. *Assume G is α-doubling and $\mathcal{P}^E_r$ is β-witnessed by $\mathcal{P}^*$ for every $r \geq r_\beta$, where $r_\beta = 2^{\log_\alpha(h\alpha^{\beta-2})}$. Then, for any $r > 0$, given an (h,r)-LSHS, we can construct, in polynomial time, an $(h\alpha^\beta, r)$-EPHS for (G,ℓ,c).*

PROOF. For any r, we need to construct a hitting set C^E for $\mathcal{P}^E_r$. Assume first $r \geq r_\beta$. Let C be the hitting set for $\mathcal{P}^*_{2^{-\beta}r}$ which is guaranteed to be sparse with respect to balls of radius $2^{-\beta+1}r$. Define the desired set by $C^E := \{v \in C : v \text{ is in some } r\text{-efficient path}\}$.

Since $\mathcal{P}^*$ is a $2^{-\beta}$-witness for $\mathcal{P}^E_r$, C^E is indeed a hitting set for $\mathcal{P}^E_r$. We are only left to prove the sparsity. Take some $u \in V$, by doubling constant we can cover $B^+_{2r}(u)$ by at most α^β balls of radius $2^{-\beta+1}r$. Each of these balls contains at most h elements of C, therefore the sparsity is as claimed. The argument for reverse balls is identical.

Now we analyse the case $r < r_\beta$. It is no longer true that efficient paths are witnessed, but now the neighborhoods are small. We first claim that, for any $v \in V$ and $r > 0$, $|B_r(v)| \leq \alpha^{\log_2 r+1}$. Indeed, using the doubling property $\log_2 r + 1$ times, we can cover $B_r(v)$ with balls of radii $1/2$. Since the minimum edge length is 1, all of these balls must be singletons and the claim follows. Now we can take $C = V$ as the EPHS. Clearly C hits all the paths and the local sparsity is at most the size of the ball. Using our assumption on r_β, we verify that $|B_{2r}(v)| \leq \alpha^{\log_2 r_\beta + 2} \leq h\alpha^\beta$. $\square$

3.4. Average-Case Performance Guarantees

Converting the partial-witness condition to a more interpretable condition is difficult in general, as the structure of $\mathcal{P}^*$ and $\mathcal{P}^E$ may be complex. One way to get such a condition, however, is by considering *average-case* performance metrics. For this, we relax the definition of HD in two ways: (i) we require LSHS to be locally sparse "on average" over all nodes, and (ii) we only require the existence of LSHS (as opposed to a hitting set for $S_r(v,\mathcal{Q})$).

Definition 8 (Average LSHS). Given $r > 0$ and a system $\mathcal{Q}$, a set $C \subseteq V$ is an average (h, r)-LSHS if it hits $\mathcal{Q}_r$ and is locally sparse in average, i.e., $\frac{1}{n}\sum_{v\in V}|B_{2r}(v) \cap C| \leq h$.

Definition 9 (Average HD). The system $\mathcal{Q}$ has average HD h if, for every $r > 0$, there exists an average (h, r)-LSHS.

Recall that all we need to construct HL is the ability to find LSHS, hence our definition is transparent in the sense that we ask only for the existence of LSHS. We note that (Abraham et al. 2011a) also introduces a notion of average HD, but they use a more restrictive concept. Indeed, their stronger condition implies both (i) existence and (ii) ability to find LSHS. Therefore, we need to prove that existence is enough to find these sets (approximately) in polynomial time, we do this in Theorem 4. Finally, we remark that our weaker notion of average HD allows us to get stronger results (because it is easier to satisfy and verify), hence we believe it can be useful in other problems too.

Theorem 4. *If $\mathcal{P}^*$ has average HD h, then we can obtain, in polynomial time, HL with average size $\frac{1}{n}\sum_{v\in V}|L^+(v)| \leq h' \log D$ and $\frac{1}{n}\sum_{v\in V}|L^-(v)| \leq h' \log D$, where $h' = O(h \log(hn))$.*

Note that since query time depends linearly on the hub size, the above result implies both storage and performance bounds.

Proof. We only show how to compute average LSHS, since the construction of HL is the same as in Proposition 2 and the bound for the size easily follows. The objective is to obtain a set C_i that is an average $(h', 2^i)$-LSHS. This turns out to be a minimum-cost hitting set problem. Indeed, we want to solve

$$C_i = \text{argmin}\left\{\sum_{v\in V}|B_{2^{i+1}}(v) \cap C| : C \subseteq V, C \text{ hits } \mathcal{P}^*_{2^i}\right\}.$$

This follows from a symmetry argument, assigning to each node u the cost $c(u) = |\{v \in V : u \in B_{2^{i+1}}(v)\}|$. On the other hand, given a minimum cost hitting set problem with optimum value τ, if the set system has VC-dimension d, the algorithm in (Even et al. 2005) finds a solution, in polynomial time, with cost at most $O(d\tau \log(d\tau))$.

By assumption, the minimum of the problem is at most hn. We apply the algorithm in (Even et al. 2005) and obtain a solution C_i with cost at most $O(hn \log(hn))$; this gives the promised average $(h', 2^i)$-LSHS. □

3.4.1. Relaxed Witness We describe a realistic setting where we can obtain small, in average, HL for the CSP.

First, we assume that individual edges are shortest paths. This assumption is mainly for exposition purposes and can be further relaxed using the same analysis. Second, we add an additional constraint wherein we insist our efficient paths have bounded *stretch* compared to the shortest path; note that this is natural in applications as users do not want to be presented solutions which are far away from the optimum, even if it saves them budget. Formally, we define:

Definition 10 (Stretch). An algorithm for CSP has stretch $\mathrm{St} \geq 1$ if, $\forall s,t \in V$ and $b \leq B$, it outputs $\mathrm{dist}(s,t|b)$ whenever $\mathrm{dist}(s,t|b) \leq \mathrm{St} \cdot \mathrm{dist}(s,t)$ and outputs "infeasible" when $\mathrm{dist}(s,t|b) > \mathrm{St} \cdot \mathrm{dist}(s,t)$.

We henceforth treat St as an extra constraint given by the application. Let $E_c := \{e \in E : c_e > 0\}$ denote the set of 'costly' edges. We define the following notion of an *overpass*:

Definition 11 (Overpass). For $r > 0$, the edge $e = (u,v)$ is an r-overpass if:
(1) e belongs to a path $Q \in \mathcal{P}^E_{2r} \setminus \mathcal{P}^*$
(2) both u and v are endpoints of paths in $\mathcal{P}^*_{r(2/\mathrm{St}-1)}$ and
(3) $\min(\mathrm{dist}(e,E_c), \mathrm{dist}(E_c,e)) \leq 3r/2$

Essentially, overpasses are edges connecting long shortest-paths in a costly zone; Fig. 3 shows an example. In case costs are contiguous (for example, tolls on highways or traffic jams), then the definition corresponds to the intuitive notion of an overpass. In more detail, our notion of overpass identifies those edges that allow to circumvent or zigzag costly edges; these are the "bad edges" because they make shortest and efficient paths too dissimilar, hence we will require that they are not too dense. Our definition restricts as much as possible which edges are considered to be overpasses: (1) is the resolution, i.e., we consider an edge only if it lies in a sufficiently long path that is efficient but not shortest, (2) requires the edge to merge long shortest paths, but the concatenation of the two paths is not shortest, hence the edge is overpassing a costly zone, (3) is mostly technical, but it restricts even more which edges are considered bad, i.e., dropping this condition does not change our results.

Our main requirement is the following *bounded growth* condition, controlling the number of r-overpasses for every scale $r > 0$.

Definition 12 (Bounded Growth). (G,c,ℓ) satisfies the bounded growth condition if, $\forall r > 0$, $|\{u \in V : \exists v, uv \text{ is an } r\text{-overpass}\}| \leq \phi(2r)$, where $\phi(r) := nh\alpha^{\beta-2-\log_2 r}$

Observe that ϕ is a slowly decreasing function of r and, even when $r = D$, we allow for overpasses. Now, with these conditions, we get our main result of this section:

Theorem 5. *Let (G,ℓ,c) be a network with HD h and doubling constant α. If the bounded growth is satisfied, we can obtain, in polynomial time, hub labels for CSP queries that guarantee average query-time $O((b+1)h'\alpha \log D)$ and total storage $O(nB \cdot Bh'\alpha \log D)$, where $h' = O(h \log(hn))$.*

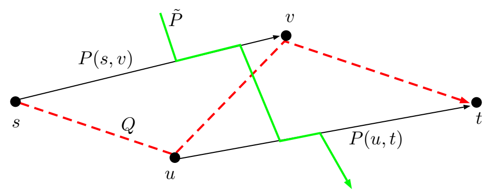


Figure 3 Edge uv here is an overpass, lying on the efficient path Q (dashed red), connecting a pair of long shortest paths $P(s,v)$ and $P(u,t)$, and close to costly edges $\tilde{P}$ (solid green).

Roadmap of the proof. The remainder of this section is devoted to the proof of Theorem 5. We henceforth refer to the average HD of $\mathcal{P}^E$ as average CHD. We prove that, under the bounded growth condition, the average CHD is $h_c \leq 2h\alpha$, which intuitively allows for the construction of HL based on our previous result (Theorem 4). To bound the average CHD, we need to operate under even less restrictive assumptions than the partial-witness. We first weaken the partial-witness concept, as defined below, by allowing for an additional *supplementary witness set* D such that *every efficient path is either witnessed by a shortest-path or hit by* D. This corresponds to the intuition that a few bad efficient paths should not completely ruin the algorithm. This weakened concept drives our analysis and allows us to bound the average CHD.

Definition 13 (Weak Partial Witness). Given $\beta \geq 0$, we say a path system $\mathcal{Q}$ is weakly β-witnessed by the path system $\mathcal{Q}'$ if, for every $r > 0$, $\exists D_r \subseteq V$ such that, $\forall Q \in \mathcal{Q}_r$ either (1) Q is β-witnessed by $\mathcal{Q}'$ or (2) Q is hit by D_r. Additionally we require $|D_r| \leq \phi(r)$.

Intuitively, the supplementary witness set D_r takes care of all the corner cases where $\mathcal{Q}$ and $\mathcal{Q}'$ differ too much. We show that our requirement on $|D_r|$ guarantees a bound on the average HD.

Proposition 8. *Assume that G is α-doubling and let $\mathcal{Q}'$ be weakly β-witnessed by $\mathcal{Q}$. If the HD of $\mathcal{Q}$ is h, then the average HD of $\mathcal{Q}'$ is $h' \leq 2h\alpha^{\beta}$*

Proof. Let $r > 0$, and C be an $(h, 2^{-\beta}r)$-LSHS for $\mathcal{Q}$. We show that $C \cup D_r$ is an average (h,r)-LSHS for $\mathcal{Q}'$. Clearly it is a hitting set for $\mathcal{Q}'_r$ and we can compute

$$\begin{aligned}
h' &\leq \frac{1}{n}\sum_{v\in V}|B_{2r}(v)\cap(C\cup D_r)| \\
&\leq \frac{1}{n}\left(\sum_{v\in V}|B_{2r}(v)\cap C| + \sum_{v\in V}|B_{2r}(v)\cap D_r|\right) \\
&\leq \frac{1}{n}\left(\sum_{v\in V}h\alpha^{\beta} + \sum_{v\in D_r}|B_{2r}(v)|\right) \leq h\alpha^{\beta} + \frac{|D_r|\alpha^{\log_2 r+2}}{n}.
\end{aligned}$$

In the third inequality we used that C is sparse with respect to balls of radius $2^{-\beta+1}r$ and that $\sum_{v\in V}|B_{2r}(v)\cap D_r| = \sum_{v\in D_r}|B_{2r}(v)|$ by symmetry of the bi-directional balls. In the last inequality we used that, by doubling constant, balls of radius r have at most $\alpha^{\log_2 r+1}$ elements. Since $|D_r| \leq \phi(r)$, the result follows. $\square$

This now allows us to link $\mathcal{P}^E$ and $\mathcal{P}^*$ as follows:

Proposition 9. *Under the bounded growth condition, $\mathcal{P}^*$ is a weak 1-witness for $\mathcal{P}^E$.*

Proof. We first need some additional notation: For a path Q and two vertices $u, v \in Q$ we denote $Q[u,v] \subseteq Q$ as the sub (u,v)-path; for two paths P, Q with a common endpoint, we denote $P|Q$ as their concatenation.

Consider $Q \in \mathcal{P}^E$ with endpoints s,t and set $\ell(Q) = 2r$. We will show how to obtain a vertex for the supplementary witness set in case Q is not witnessed and then we bound the size of the set. Assume that $Q \neq P(s,t)$, otherwise the path is trivially witnessed. Let $(u,v) \in Q$ be such that $\ell(Q[s,v]), \ell(Q[u,t]) \geq r$ (see Fig. 3). If either $Q[s,v]$ or $Q[u,t]$ is a shortest path or $\ell_{uv} \geq r$, then Q is witnessed. Thus, we henceforth assume that all of the above conditions fail.

We claim that uv is a r-overpass (in fact, this is the exact scenario depicted in Fig. 3). Condition 1 is clearly satisfied. Condition 2 also holds because both $Q[s,v]$ or $Q[u,t]$ have stretch at most $\frac{2-\mathrm{St}}{\mathrm{St}}$. To see this, note that both $P(s,v)|Q[v,t]$ and $Q[s,u]|P(u,t)$ are no shorter than $P(s,t)$ and $\mathrm{St}\,\ell(P(s,t)) \geq \ell(Q)$, hence $\frac{2r}{\mathrm{St}} \leq \ell(P(s,t)) \leq \ell(P(s,v)) + \ell(Q[v,t])$ and $\frac{2r}{\mathrm{St}} \leq \ell(P(s,t)) \leq \ell(Q[s,u]) + \ell(P(u,t))$. Since each path $Q[s,u], Q[v,t]$ has length less than r, it follows that both $P(s,v), P(u,t)$ have length strictly greater than $\frac{2-\mathrm{St}}{\mathrm{St}}r$. Finally, to show condition 3, we have $\ell(Q[s,v]) + \ell(Q[u,t]) = 2r + \ell_{uv}$ and since $\ell_{uv} < r$, one of $Q[s,v]$ or $Q[u,t]$ has length at most $3r/2$. Since neither of these paths is shortest, it must be that both $P(s,v), P(u,t)$ have costly edges and thus one of u,v is closer than $3r/2$ to E_1 and the condition is satisfied.

For every path of length $2r$, we can thus either exhibit a witness or show that it contains an overpass and add use the tail of the edge as a supplementary witness. For a fixed $r > 0$, we need to add at most $\phi(r)$ nodes to D_r to cover all the efficient paths. The result follows. $\square$

We are almost ready to prove that bounded growth allows to solve the CSP. The last piece is Proposition 10, the proof of which follows form similar arguments as those in Theorems 1 and 4.

Proposition 10. *If the average HD of $\mathcal{P}^E$ is h_c, then we can construct, in polynomial time, hub labels for CSP, which guarantee average query-time $O((b+1)h_c' \log D)$ for queries with budget b, and total storage requirements $O(nB \cdot Bh_c' \log D)$.*

Proof of Theorem 5. We argue that the average CHD is $h_c \leq 2h\alpha$. By Proposition 9, $\mathcal{P}^E$ is weakly 1-witnessed by $\mathcal{P}^*$. It follows by Proposition 8 that the average CHD is at most $2h\alpha$ as needed. Applying Proposition 10 yields the result. $\square$

4. Scalable CSP Algorithms: Implementations and Experiments

Our theoretical results in the preceding sections suggest that using hub labels for CSP queries should perform well in road networks, as these are known to have low highway dimension, and potentially also satisfy the (average) partial witness property. We use our theoretical findings to guide the construction of practical algorithms; the modifications of the theoretical algorithms are simply to replace some of the black box routines. We now describe how our techniques can be adapted to give practical hub label constructions, and discuss experimental results for two real world networks using these methods.

4.1. Practical CSP Algorithms

We start by defining a more scalable construction of G^B. The augmented graph G^B defined in Section 3.1 is not a minimal representation as it may contain a lot of redundant information. For example, the same efficient path uv can be repeated many times in the form $\langle u,1\rangle\langle v,0\rangle$, $\langle u,2\rangle\langle v,1\rangle$ and so on. By encoding this information more efficiently, we get considerable improvements both in query time and in data storage.

We construct our *pruned* augmented graph $\tilde{G}^B$ as follows: As before, nodes are pairs $\langle v,b\rangle$, but now we add an edge $\langle v,b\rangle\langle v',b'\rangle$ only if it is essential for some efficient path, i.e., removing said edge impacts correctness. If we let $\mathcal{P}^E_{s,t}$ be the set of all efficient paths from s to t, we take every $P\in\mathcal{P}^E_{s,t}$ with cost $b\le B$ and trace it in the augmented graph such that it terminates at $\langle t,0\rangle$.

Definition 14. The pruned augmented graph is defined by $\tilde{G}^B=(\tilde{V}^B,\tilde{E}^B)$, where

$$\tilde{V}^B := \{\langle v,b\rangle : v\in V, b=0,1,\dots,B\},$$
$$\tilde{E}^B := \{\langle v,b\rangle\langle u,x\rangle : \exists s,t\in V, P\in\mathcal{P}^E_{s,t}, c(P)\le B, vu\in P, b=c(P[v,t]), x=c(P[u,t])\}.$$

In $\tilde{G}^B$ all the lengths are preserved.

Note that in $\tilde{G}^B$ there are no sink nodes, hence it has at least n nodes and nB arcs fewer compared to G^B. In the worst case, those nB arcs are the only gain by doing this process. Nevertheless, in our experiments $\tilde{G}^B$ is up to 60% smaller than G^B. Observe that, by running Dijkstra in G^B, $\tilde{G}^B$ can be computed in time $O(n^2B\log(nB))$.

4.1.1. HD of the pruned augmented graph A shortest path in $\tilde{G}$ does not necessarily project to an efficient path, even if the path ends in a node of the form $\langle t,0\rangle$. In contrast, if P projects to an efficient path, then necessarily P is shortest. To bound the HD, the correct system to study is

$$\tilde{\mathcal{P}^B} := \{P : P \text{ ends in a node } \langle t,0\rangle, \bar{P}\in\mathcal{P}^E, c(\bar{P})\le B\}.$$

The following result shows how the HD of this system relates to that of $\mathcal{P}^E$. We omit the proof since it is identical as the one in Proposition 5.

Proposition 11. *Given CHD h_c, the HD of $\tilde{\mathcal{P}}^B$ is Bh_c.*

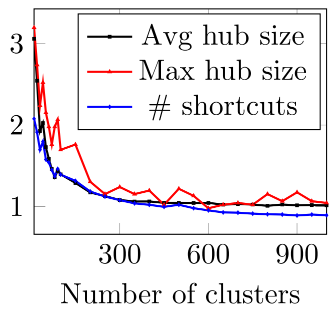
4.1.2. Types of queries We test our algorithms with two different tasks. Recall that our preprocessing is done for some fixed maximum budget B. The first task we consider is a *frontier query*, wherein given s and t, we return the lengths of all efficient paths with costs $b = 0, 1, \ldots, B$. The second we call a *specific query*, we return $\text{dist}(s,t|b)$ for given s, t, b (i.e., a single efficient path).

Note that the pruned augmented graph $\tilde{G}^B$ is designed for frontier queries. To see this, fix the terminal $\langle t, 0\rangle$. As we ask for the shortest path from $\langle s, B\rangle, \langle s, B-1\rangle, \ldots, \langle s, 0\rangle$ we are guaranteed to recover the entire frontier. On the other hand, it may be that the shortest path between $\langle s, b\rangle$ and $\langle t, 0\rangle$ does not correspond to $\text{dist}(s,t|b)$. This occurs when b is not a tight budget and the efficient path requires less.

To answer specific queries, we modify $\tilde{G}^B$ by adding extra edges. For every $v \in V(G)$ and $b = 1, 2, \ldots, B$, we include the edge $\langle v, b\rangle\langle v, b-1\rangle$ with length 0. A simple argument shows that with the added edges, the shortest path between $\langle s, b\rangle$ and $\langle t, 0\rangle$ has length $\text{dist}(s,t|b)$.

4.1.3. HL construction via Contraction Hierarchies We use some techniques described in Abraham et al. (2011b) combined with an approach tailored for augmented graphs. The CH algorithm takes as input any ranking (i.e., permutation) of the nodes, and proceeds by removing nodes from the lowest rank first. Whenever a node is removed, we add new edges, called shortcuts, if needed to preserve the shortest paths. Once we have a graph with shortcuts, a CH search is a special variant of Dijkstra where only higher rank nodes are explored, i.e., we never take an edge uv if $\text{rank}(u) > \text{rank}(v)$. The main idea in our construction is to choose an appropriate ranking, and then define the forward hubs of v as the nodes visited during a contraction-based forward search starting at v. The reverse hubs are defined analogously. These are valid hubs, since the highest rank node in a path is guaranteed to be in both hubs.

The choice of the ranking function is crucial. For our experiments, we ranked nodes in G by running a greedy approximate SP cover, selecting the highest rank node as the one covering most uncovered paths in $\mathcal{P}^*$ and continuing greedily. Specifically, start with a cover $C = \varnothing$ and compute the set of all shortest paths $\mathcal{P}^*$. Take a node $v \notin C$ hitting most paths in $\mathcal{P}^*$, then remove all those paths from $\mathcal{P}^*$, add v to C and iterate. The rank is defined as n for the first node added to C, $n-1$ for the second and so on. To implement the SP cover we follow the algorithm in Abraham et al. (2011b). A practical hurdle in such an approach is that to compute a shortest path cover, a direct approach requires storing all the shortest paths in memory, which, in most mapping applications, is infeasible. To circumvent this, we approximate the shortest-path cover by computing only $k << n$ shortest-path trees and covering these greedily. We use an off-the-shelf clustering method to obtain k cluster centers, from which we compute the shortest paths. As shown in Figure 4, the clustering



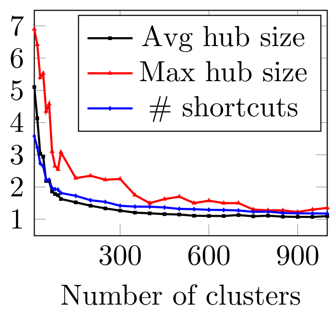


Figure 4 Performance of clustering for San Francisco (left) and Luxembourg (right). In the y-axis the quantities are normalized by the cover using all the shortest paths ($k = n$ clusters). For example, the curve for average hub size, say $y_1(k)$, represents that using k clusters the average hub is $y_1(k)$ times bigger than the hubs using all the shortest paths; in the plot we see that, with a few clusters, the size at most doubles and it can be improved by augmenting k. Note that, while the number of short-cuts is an indicator of performance, it is not perfectly correlated with the hub size.

approach provides a very good approximation of the hubs with even a small k. We stress that this is just an easy way to get a ranking; more sophisticated heuristics that decide on-line the next node to contract usually work well in practice Bast et al. (2016), Rice and Tsotras (2010). Using another contraction scheme may expedite our algorithms and reduce the hub size.

Depending on the size of our instance, and the specific queries, we work with either the augmented graph G^B or the pruned augmented graph $\tilde{G}^B$. Even though $\tilde{G}^B$ takes time to compute, it can speed up the overall process and yield considerably better hubs. Given a ranking for nodes in G, we contract $\tilde{G}^B$ as follows. Say that V is ordered according to the ranking, so node 1 is the least important and n the most important. In $\tilde{G}^B$, we first contract the nodes $\langle 1, b\rangle$ for $b = B, \dots, 0$, then the nodes $\langle 2, b\rangle$ and so on till the nodes $\langle n, b\rangle$ are the last to contract. Finally, when contracting a node v, if u is a predecessor and w a successor of v, we add the short-cut uw only if, by removing v, the distance from u to w is altered, and the new shortest path from u to w is efficient. We can go even further; the short-cut uw is unnecessary if the shortest path is not efficient, even if the distance changes.

To obtain better hubs we prune the results obtained by CH searches. If w is in the forward search of v with distance d, it might be that $\text{dist}(v, w) < d$, this occurs because the search goes only to higher rank nodes and the discovered path is missing some node. When $\text{dist}(v, w) < d$, we can safely remove w from the hub of v, since the highest ranked node in a shortest path will have the correct distance. For frontier queries, we can also prune away a node w if the (v, w)-path has a surplus of budget. The entire process can be summarized in the following steps.

1. Compute the shortest paths in G and use a greedy approach to obtain a cover C
2. Compute the pruned augmented graph $\tilde{G}^B$

3. Contract $\tilde{G}^B$ using the rank induced by C
4. Create hubs $L^+(v), L^-(v)$ using CH
5. Prune the hubs by running HL queries between v and nodes in $L^+(v)$. Run a similar process for $L^-(v)$.

Recall that, for some instances, we skip step 2 and contract G^B instead. Note that in the last step we bootstrap HL to improve it. This works because the fact that some nodes have incorrect distance labels does not impact the correctness of a HL query; a node minimizing the distance is returned and such node must have a correct label.

4.2. Experiments

All our experiments were performed on a 64-bit desktop computer with a 3.40GHz Intel Core i7-6700 processor and 16GB RAM running Ubuntu 16.04. Our code is written in Python 2.7. We use the library Networkx for the graph representation and Dijkstra's algorithm. Although all the steps can be parallelized, we did not implement this. Our complete code (implementation and artificial dataset) is publicly available at `github.com/albvera/HHL_CSP`.

We evaluated the performance of our algorithms with real-world test networks: downtown San Francisco with 2139 nodes and 5697 edges for which real-world travel-time data was available as a Gaussian mixture model Hunter et al. (2013), and Luxembourg City with 4026 nodes and 9282 edges for which travel-time distributions were synthesized from speed limits Niknami and Samaranayake (2016), as real-world data was unavailable.

In our experiments, we use the mean travel times for our length function and the following cost structure; the top 10% of edges with the highest variance are assigned cost 1 and the rest cost 0. This is a measure of risk, since edges with high variance are prone to cause a delay in the travel time.

4.2.1. Query-time performance Table 1 presents the CSP computation times for different maximum budgets B. For frontier queries, labeled as 'f', the query times are measured as the average of 1000 random s,t. For specific queries, labeled as 's', the times are measured as 1000 random triplets s,t,b. The column for $B=0$ represents the original graph (without augmentation). As can be seen in the experimental results, our method finds the constrained shortest path solution on average four orders of magnitude faster than running Dijkstra's algorithm on the augmented graph. Preprocessing for frontier queries results in a more compact set of hub labels, since a node $\langle s,b\rangle$ needs to store information for paths with budget exactly equal to b (in case the path is efficient, otherwise it is not stored). On the other hand, for specific queries, $\langle s,b\rangle$ needs to store information for all budgets up to b. The preprocessing time does not include the cover computation, since this is a flat cost of at most the time to pre-process the instance $B=0$.

B	Prepro [m]	Avg F Size	Avg B Size	Query Dij [ms]	Query HL [ms]
0	1	23	22	10.71	0.005
5-f	5	16	28	80.10	0.02
5-s	5	57	28	31.75	0.01
10-f	9	9	28	168.11	0.03
10-s	10	68	28	56.19	0.01
15-f	12	6	28	237.64	0.03
15-s	16	73	28	77.59	0.01
20-f	17	5	28	342.47	0.03
20-s	20	77	28	100.26	0.01
25-f	22	4	28	460.95	0.03
25-s	25	80	28	126.52	0.01
30-f	26	3	28	569.13	0.03
30-s	31	84	28	152.75	0.01

B	Prepro [m]	Avg F Size	Avg B Size	Query Dij [ms]	Query HL [ms]
0	6	18	18	16.35	0.004
5-f	1	18.0	18.0	175.04	0.02
5-s	21	42.9	18.7	72.03	0.01
10-f	2	18.0	18.0	361.15	0.04
10-s	35	49.3	18.7	102.52	0.01
15-f	3	18.0	18.0	577.89	0.06
15-s	46	53.3	18.7	140.80	0.01
20-f	4	18.0	18.0	821.94	0.07
20-s	60	56.5	18.7	183.11	0.01
25-f	5	18.0	18.0	974.84	0.09
25-s	77	59.5	18.7	227.17	0.01
30-f	7	18	18	1247.72	0.10
30-s	93	62.3	18.7	272.41	0.01

Table 1 Experimental results for San Francisco (left) and Luxembourg City (right). Query times are measured with 1000 random s, t pairs for each network and multiple maximum budget levels B. Results on rows $B - f$ correspond to computing the solution frontier for all budgets $b \leq B$ while rows $B - s$ correspond to computing the solution for budget level b.

Note that preprocessing frontier queries in Luxembourg is faster, despite the network being bigger, this can be explained by the structural properties. For example, in Luxembourg there are more highways and fast roads.

Observe that the *average hub size decreases* in San Francisco for frontier queries, this is because in this instance we use $\tilde{G}$, which prunes away most of the nodes, thus many nodes $\langle v, b\rangle$ are isolated and have empty hubs. The longer preprocessing time for frontier queries can be explained as follows. There are many cases when two nodes are not reachable, to detect this requires Dijkstra to explore the entire graph. In contrast, for specific queries we add extra edges $\langle s, b\rangle\langle s, b-1\rangle$, hence a reachability test ends, in average, earlier. In the contraction step, we want to remove a node without altering the shortest path, a process that requires many reachability tests.

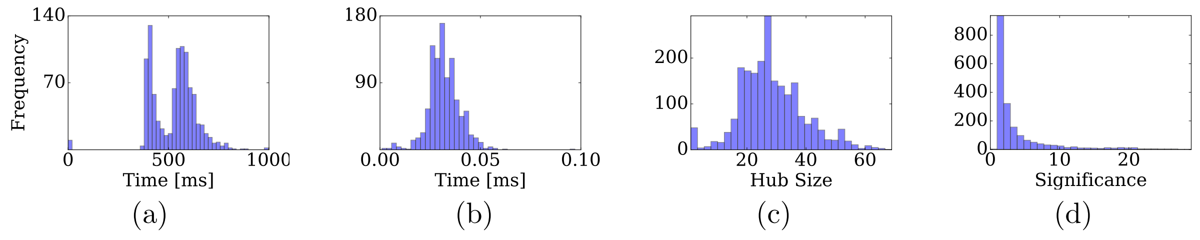
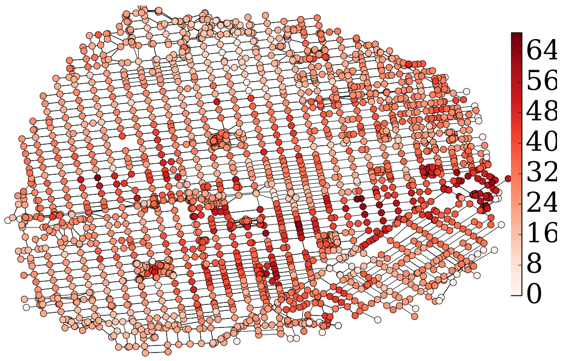
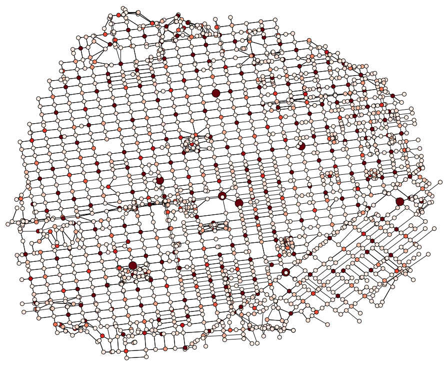


Figure 5 Histogram frontier queries for Dijkstra (a) and HL (b). Times are 1000 random pairs in San Francisco augmented with $B = 25$. Size of reverse hubs (c) and significance (d) for frontier queries in San Francisco augmented with $B = 25$.

4.2.2. Hub sizes and node significance We focus the analysis on two meaningful quantities. The first is hub size, which is well captured by $|L^{-}(\langle t,0\rangle)|$ for $t \in V$. Indeed, for frontier queries the reverse hub is bounding the space requirements; for specific queries the same is true up to a constant factor. For the second quantity, we define the significance of $s \in V$ as the number of hubs containing s, i.e., $\sum_t \sum_b \mathbb{1}_{\{\langle s,b\rangle \in L^{-}(\langle t,0\rangle)\}}$. Intuitively, a node is highly significant if it belongs to many efficient paths. Figure 5 shows a histogram of these metrics.



(a) Heatmap of hub-sizes for SF



(b) Heatmap of significance for SF

Figure 6 Heat maps for frontier queries in San Francisco augmented with $B = 25$. On the left we see the hub size. Note that the size is not homogeneous, but rather we can observe clusters and neighborhoods tend to be similar. On the right the significance. The most significant nodes have been drawn bigger. The top 3 most significant correspond to Geary Blvd & Gough St, Franklin St & O'Farrel St and Market St & Polk St.

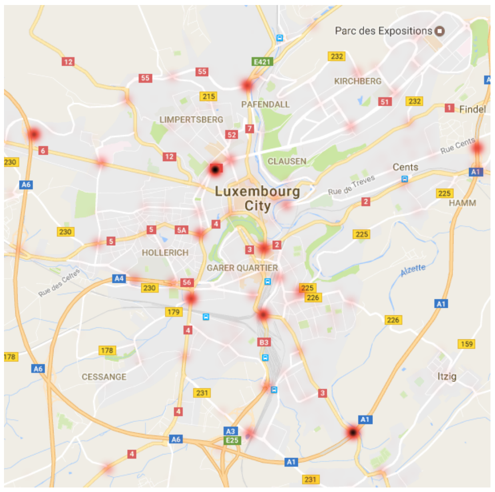


Figure 7 Heat map of significance for frontier queries in Luxembourg City augmented with $B = 25$. Notice how the highly significant nodes are in main road crossings.

Fig. 6 presents the spatial relationships between the hub size and significance in the San Francisco network. Fig. 7 shows the spatial distribution of significance in the Luxembourg network. We observe that highly significant nodes tend to have small hub size. The intuition is simple, if most hubs contain s, then is easier for s to satisfy the cover property with a small hub. Note also that the hub size resembles an harmonic function; nodes are mostly similar to their neighbors.

5. Conclusions

We introduced a new network primitive, the Constrained Highway Dimension, and used it to parametrize the storage and running time of data structures that support fast CSP queries. Our aim was to study when efficient SP computation yields a similar performance for the harder CSP problem. We derived conditions under which this holds and we can compare the HD and CHD. For the worst-case setting, the conditions are given by the partial-witness and for average-case by the milder bounded growth. Both conditions have intuitive interpretations in terms of the physical structure of the network.

On the practical side, we validated our findings by developing algorithms that performed four orders of magnitude better on real-world networks, compared to standard techniques. Our work is a first step in bridging the gap between SP and CSP algorithms and we believe our findings are promising for real-world applications.

Acknowledgments

The authors would like to thank Moritz Kobitzsch for sharing sanitized versions of the San Francisco and Luxembourg road networks.

The authors gratefully acknowledge funding from the US Army Research Laboratory under grant W911NF-17-1-0094, and the National Science Foundation under grants CNS-1952011, CNS-1955997, DMS-1839346 and ECCS-1847393.

References

Abraham I, Delling D, Fiat A, Goldberg AV, Werneck RF (2011a) Vc-dimension and shortest path algorithms. *International Colloquium on Automata, Languages, and Programming*, 690–699 (Springer).

Abraham I, Delling D, Fiat A, Goldberg AV, Werneck RF (2016) Highway dimension and provably efficient shortest path algorithms. *Journal of the ACM (JACM)* 63(5):41.

Abraham I, Delling D, Goldberg AV, Werneck RF (2011b) A hub-based labeling algorithm for shortest paths in road networks. *International Symposium on Experimental Algorithms.*

Abraham I, Fiat A, Goldberg AV, Werneck RF (2010) Highway dimension, shortest paths, and provably efficient algorithms. *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms.*

Babenko M, Goldberg AV, Kaplan H, Savchenko R, Weller M (2015) On the complexity of hub labeling. *International Symposium on Mathematical Foundations of Computer Science.*

Barrett C, Bisset K, Holzer M, Konjevod G, Marathe M, Wagner D (2008) Engineering label-constrained shortest-path algorithms. *International Conference on Algorithmic Applications in Management.*

Bast H, Delling D, Goldberg A, Müller-Hannemann M, Pajor T, Sanders P, Wagner D, Werneck RF (2016) Route planning in transportation networks. *Algorithm engineering*, 19–80 (Springer).

Becker A, Klein PN, Saulpic D (2018) Polynomial-time approximation schemes for k-center, k-median, and capacitated vehicle routing in bounded highway dimension. *26th Annual European Symposium on Algorithms (ESA 2018)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik).

Clawson Z, Ding X, Englot B, Frewen TA, Sisson WM, Vladimirsky A (2015) A bi-criteria path planning algorithm for robotics applications. *arXiv preprint arXiv:1511.01166* .

Cohen E, Halperin E, Kaplan H, Zwick U (2003) Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing* 32(5).

Correa J, Harks T, Kreuzen VJC, Matuschke J (2017) Fare evasion in transit networks. *Operations Research* 65(1):165–183.

Demetrescu C, Goldberg AV, Johnson DS (2009) *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74 (American Mathematical Soc.).

Disser Y, Feldmann AE, Klimm M, Könemann J (2019) Travelling on graphs with small highway dimension. *Graph-Theoretic Concepts in Computer Science LNCS 11789* 175.

Elkin M, Pettie S (2016) A linear-size logarithmic stretch path-reporting distance oracle for general graphs. *ACM Transactions on Algorithms (TALG)* 12(4):1–31.

Even G, Rawitz D, Shahar SM (2005) Hitting sets when the vc-dimension is small. *Information Processing Letters* 95(2).

Fan Y, Kalaba R, Moore II J (2005) Arriving on time. *Journal of Optimization Theory and Applications* 127(3).

Feldmann AE (2019) Fixed-parameter approximations for k-center problems in low highway dimension graphs. *Algorithmica* 81(3):1031–1052.

Feldmann AE, Fung WS, Könemann J, Post I (2018) A $(1+\varepsilon)$-embedding of low highway dimension graphs into bounded treewidth graphs. *SIAM Journal on Computing* 47(4):1667–1704.

Feldmann AE, Marx D (2018) The parameterized hardness of the k-center problem in transportation networks. *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik).

Festa P (2015) Constrained shortest path problems: state-of-the-art and recent advances. *Transparent Optical Networks (ICTON), 2015 17th International Conference on.*

Funke S, Nusser A, Storandt S (2014) On k-path covers and their applications. *Proceedings of the VLDB Endowment* 7(10):893–902.

Geisberger R, Sanders P, Schultes D, Delling D (2008) Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *International Workshop on Experimental and Efficient Algorithms.*

Hoy D, Nikolova E (2015) Approximately optimal risk-averse routing policies via adaptive discretization. *AAAI.*

Hunter T, Abbeel P, Bayen AM (2013) The path inference filter: model-based low-latency map matching of probe vehicle data. *Algorithmic Foundations of Robotics X.*

Kosowski A, Viennot L (2017) Beyond highway dimension: Small distance labels using tree skeletons. *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.*

Niknami M, Samaranayake S (2016) Tractable pathfinding for the stochastic on-time arrival problem. *International Symposium on Experimental Algorithms.*

Nikolova E, Kelner JA, Brand M, Mitzenmacher M (2006) Stochastic shortest paths via quasi-convex maximization. *European Symposium on Algorithms.*

Rice M, Tsotras VJ (2010) Graph indexing of road networks for shortest path queries with label restrictions. *Proceedings of the VLDB Endowment* 4(2).

Sabran G, Samaranayake S, Bayen A (2014) Precomputation techniques for the stochastic on-time arrival problem. *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX).*

White C (2015) Lower bounds in the preprocessing and query phases of routing algorithms. *Algorithms-ESA 2015*, 1013–1024 (Springer).

Woodard D, Nogin G, Koch P, Racz D, Goldszmidt M, Horvitz E (2017) Predicting travel time reliability using mobile phone gps data. *Transportation Research Part C: Emerging Technologies* 75:30–44.

Appendix A: Different notions of HD

We give below the main notions of HD. We note that they are all similar and they are interlinked in terms of their properties, see (Feldmann et al. 2018, Section 9) for several results on this. We limit the discussion here to highlight the main differences.

- (Abraham et al. 2016): this is the closest to our definition, but we weaken it (make it easier to satisfy). The difference is that they make the notion of r-significant stronger by also calling r-significant paths P whose length is $\ell(P) < r$, but can be extended by adding at most one node at each end of P to obtain a path P' with $\ell(P')$.
- (Abraham et al. 2011a): it is also very similar, but technically different in that we define path neighborhoods $S_r(v)$ as sets of paths (while they use the union of nodes belonging to those paths), hence in our definition there are fewer elements to hit. On the other hand, they consider paths of lengths between r and $2r$, while we consider paths longer than r. We could also add the restriction of length at most $2r$ without changing any of the results, but at the cost of making the definition even more involved.
- (Abraham et al. 2010): it is slightly weaker, in that they look at paths are r-significant and contained in a larger ball (of radii $4r$) vs the notion of r-significant and within distance $2r$ of a node. Specifically, for each v, they ask for hitting sets of paths $P \subseteq B_{4r}(v)$ such that $\ell(P) > r$.

For $\mathcal{P}^*$, a consequence of considering less-restrictive path-neighborhoods is that the highway dimension returned by our definition is smaller than that of (Abraham et al. 2016). In particular, unlike (Abraham et al. 2016), the HD of G as per our definition is not an upper bound to the maximum degree Δ or the doubling constant α. The notion of (Abraham et al. 2010) does not bound the doubling constant.

With respect to our average HD in Section 3.4, we note the following.

Remark 3. The algorithm in Theorem 4 makes one call to the VC-dimension solver for each C_i. On the other hand, the algorithm in (Abraham et al. 2016) calls up to n times the solver for each C_i. Finally, there is an extra $\log n$ factor in the approximation guarantee, but now the value of h can be much smaller.

We now discuss how our results extend to the definition in (Abraham et al. 2016), which we refer to as strong-HD. The strong-HD defines a path P to be r-significant if, by adding at most one hop at each end, we get a shortest path P' longer than r. The path P' is called an r-witness for P. Intuitively, a path is significant if it represents a long path. Observe that, if $P \in \mathcal{P}^*$ is such that $\ell(P) > r$, then P is r-significant by definition. We remark also that a path can have many r-witnesses.

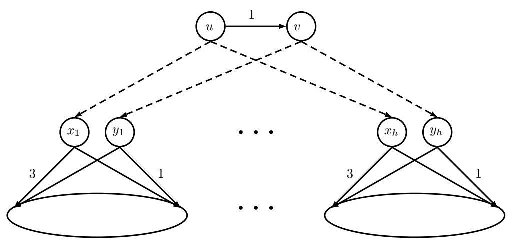


Figure 8 Example where the EPHS is much larger than the LSHS.

Finally, the path neighborhood must also be strengthened. The path $P \in \mathcal{P}^*$ belongs to $S_r^+(v)$ if, P has some r-witness P' such that $\text{dist}(v, P') \leq 2r$. The reverse neighborhood $S_r^-(v)$ is defined analogously. With this modified versions of r-significant and neighborhood, the notions of LSHS and HD are the same as our previous definitions.

Under the strong-HD, we have $\Delta \leq h$ and $\alpha \leq h+1$. Additionally, this definition allows proving results for CH. Finally, we show that even for the strong-HD, CHD and HD can still be off by a factor of n.

Proposition 12. *For any h, we can construct a family of networks such that the sparsity of LSHS is h and that of EPHS is arbitrarily worse than h.*

Proof. First, we construct an example where the sparsity grows from h to h^2. Consider an h-ary tree rooted at u with three levels, i.e., with $1+h+h^2$ nodes. Now add a node v with h children as in Figure 8. The grandchildren of v are the same as the grandchildren of u.

All the edges are bidirectional and have unit cost. The lengths are as follows: ux_i and vy_i (dashed in Figure 8) are zero; uv and from y_i to the leafs is one; from x_i to the leafs is three. It is easy to see that the sparsity of a LSHS is $h+1$.

On the other hand, every leaf w is a 2-efficient path. Indeed, it can be extended to $x_i w$ that is the shortest path from x_i to w with constraint 1. All the leafs are in the ball $B_4(u)$, so the sparsity is at least h^2.

The general case works in the same fashion. We make the sparsity grow to h^k by creating two complete, k-level, h-ary trees T and T'. Connect the root of T to the root of T' and the leafs of both trees are shared. Observe that the number of nodes is

$$\begin{aligned} n &= [k\text{-level } h\text{-ary tree}] + [(k-1)\text{-level } h\text{-ary tree}] \\ &= (h^{k+1}-1)/(h-1) + (h^k-1)/(h-1), \end{aligned}$$

therefore the sparsity is $\Theta(n)$, the worst possible. $\square$

Appendix B: Contraction Hierarchies

We present here how to extend the concept of HD in order to prove the efficiency of CH in directed graphs. Given a rank in the nodes, the shortcut process works as in the non-directed case:

1. Let G' be a temporary copy of G.
2. Remove nodes of G' and its edges in increasing rank.
3. When removing v, if some unique shortest path in G uses uvw, add (u,w) to G' with length $\ell(u,v)+\ell(v,w)$.

Call E^+ the set of edges created in the shortcut process. A source-destination query runs bidirectional Dijkstra, but each search only considers paths of increasing ranks.

As in the non-directed case, let $Q_i = C_i \setminus \cup_{j>i} C_j$ be the partition of V. All the ranks in Q_i are smaller than those in Q_{i+1}, within each Q_i the rank is arbitrary.

Lemma 1. *Let P be a shortest path in the original graph. If P has at least three vertices and $\ell(P) > 2^\gamma$, then some internal vertex of P belongs to a level Q_x, $x > \gamma$.*

Proof. The path P' obtained by removing the endpoints of P is $\ell(P)$-significant. By definition of the C_i's, $C_{\gamma+1}$ hits P' at some node u. By construction of the partition, $u \in Q_x$ with some $x > \gamma$. □

Now we show that each node adds at most h to its out-degree for each Q_i, so the process adds at most $h \log D$ to the out-degree of each node.

Lemma 2. *Assume the network admits the C_i's. For any v and fixed j, the number of shortcuts (v,w) with $w \in Q_j$ is at most h.*

Proof. Let i be the level such that $v \in Q_i$ and define $\gamma := \min(i,j)$. We claim that $w \in B^+_{2^\gamma}(v)$. Assume the claim, then the number of shortcuts is at most $|Q_j \cap B^+_{2^\gamma}(v)|$, but using local sparsity and set inclusion:

$$|Q_j \cap B^+_{2^\gamma}(v)| \le |C_j \cap B^+_{2\cdot 2^{j-1}}(v)| \le h.$$

All that remains is to prove the claim. The shortcut (v,w) was created when the process removed the last internal vertex of the shortest path $P(v,w)$ in G. Necessarily all the internal vertices are in levels at most γ, because they were removed before v and w, hence they have lower rank. Finally, apply Lemma 1 to conclude that $\ell(P(v,w)) \le 2^\gamma$. □

We need to bound the in-degree, because it could be that some node v is receiving many edges. The proof is basically the same.

Lemma 3. *Assume the network admits the C_i's. For any v and fixed j, the number of shortcuts (w,v) with $w \in Q_j$ is at most h.*

Proof. Same as in the previous lemma, but now $w \in B^-_{2^\gamma}(v)$. $\square$

We can conclude now that the number of shortcuts, i.e. $|E^+|$, is at most $2nh\log D$.

As we mentioned before, the query performs Dijkstra from the source and target, but always constructing paths of increasing rank. When scanning a vertex v, the forward search has a label $\text{dist}(s,v)'$. The labels always satisfy $\text{dist}(s,v)' \geq \text{dist}(s,v)$, but, since the algorithm only goes to higher ranks, equality is not guaranteed.

We add a pruning rule analogous to the non-directed case: when the forward search scans a node v, if $(v,w) \in E \cup E^+$ and $w \in Q_i$, then w is added to the priority queue only if $\text{rank}(w) > \text{rank}(v)$ and $\text{dist}(s,v)' + \ell(v,w) \leq 2^i$. For the reverse search, the condition is the analogous $\text{dist}(v,t)' + \ell(w,v) \leq 2^i$ when $(w,v) \in E \cup E^+$.

Proposition 13. *The query with additional pruning returns the correct distance. Additionally, each Dijkstra scans at most h nodes in each level.*

Proof. Let us analyse the forward search. Say the node v is being scanned, $w \in Q_i$ is a candidate and $\text{dist}(s,v)' + \ell(v,w) > 2^i$. If the current path P' to w is optimal, then $P(s,w)$ is 2^i-significant and it is hit by C_{i+1}. As a consequence, $P(s,w)$ contains an internal vertex with higher rank than w. This vertex cannot be in P' nor a shortcut containing it, thus contradicting the optimality of P'. We conclude that P' is not optimal and w can be ignored.

Bounding the number of scanned nodes is easy; every $w \in Q_i$ added to the queue satisfies $w \in B^+_{2^i}(s)$, so applying local sparsity we finish the proof. $\square$

As a result, the forward search adds at most $h\log D$ nodes to the queue; each of node amounts to $O(\text{outdeg}(G^+))$ operations, i.e., $O(\text{outdeg}(G) + h\log D)$ operations.

Appendix C: CHD vs. HD: Extensions

So far we have only used the structure of shortest paths in G, which, naturally, does not capture all the information in the network. It is natural to think that there is structure if we look at, for example, only zero cost edges.

Let G_0 be obtained from G by removing all the edges with cost. The networks G and G_0 define two hierarchies of roads; shortest paths in G_0 are free, but not as fast as the ones in G. Our main hypothesis now is that an efficient path P does not alternate between these two hierarchies. For example, a path that enters and exits multiple times a highway is not desirable because of turning costs.

Proposition 14. *Let $\mathcal{Q}, \mathcal{Q}'$ be two path systems with HD h and h' respectively. The HD of the system $\mathcal{Q} \cup \mathcal{Q}'$ is at most $h + h'$.*

Proof. Given $v \in V$, the union of $H_{v,r}$ and $H'_{v,r}$ hits all the paths in $S_r(v, \mathcal{Q}) \cup S_r(v, \mathcal{Q}')$. $\square$

We now relax the assumption that a system witnesses another. It could be that the efficient paths are sometimes witnessed by free paths and sometimes by shortest paths.

Theorem 6. *Assume that G has doubling constant α and $\mathcal{Q}, \mathcal{Q}'$ are systems with HD h, h' respectively. Moreover, suppose $\mathcal{P}^E$ does not alternate between $\mathcal{Q}$ and $\mathcal{Q}'$, that is, for some $\beta, \beta' > 0$, each path $P \in \mathcal{P}^E$ is either β-witnessed by some $Q \in \mathcal{Q}$ or β'-witnessed by $Q' \in \mathcal{Q}'$. Then G admits $(\alpha^\beta h + \alpha^{\beta'} h', r)$-EPHS.*

C.1. Correlated Costs

We have studied so far the case where $c(P)$ is just the sum of individual edge costs. In practice it could be that the cost depends on combinations of arcs. Think of a turn in a road network; we can turn right quickly, but turning left means waiting for a green arrow in most cases. Another example is minimizing expectation subject to bounded variance. If there is no independence, the variance of a path is not the sum of individual variances.

We explain now how to deal with more general cases using the same framework. Assume the cost function $c_2 : E \times E \to \mathbb{N} \cup \{0\}$ depends on pairs of edges, so if a path is $P = e_0 e_1 \dots e_k$, then the cost would be $c_2(P) = \sum_{i=1}^{k} c_2(e_{i-1}, e_i)$. The nodes in the augmented graph will be triplets $\langle u, v, b \rangle$, where v is the current state, u is the previous state and b is the available budget. The arcs are given by

$$(\langle u, v, b \rangle, \langle v, w, b' \rangle), \quad uv, vw \in E, b' = b - c_2(u, v, 2).$$

Define analogously the concept of efficient paths. It is easy to see that, as in the previous case, shortest paths in the augmented graph are efficient paths. The system $\tilde{\mathcal{P}^E}$ of such paths may also allow for a β-witness. With the previous properties we can construct the hub labels in the same fashion to prove the following result.

Theorem 7. *Assume the system $\tilde{\mathcal{P}^E}$ has HD $\tilde{h}$. Then, there exists HL such that queries s, t, b can be answered in time $O(b\tilde{h} \log D)$ and the space requirement is $O(Bn \cdot B\tilde{h} \log D)$. In particular, if $\mathcal{P}^*$ is a β-witness for $\tilde{\mathcal{P}^E}$, then $\tilde{h} \le h\alpha^\beta$.*

Appendix D: Additional Proofs

Proof of Proposition 3. Let P be the shortest path from $\langle s, b \rangle$ to t^-, and $\bar{P}$ its projection. To reach t^-, P must pass through some $\langle t, b' \rangle$, $b' \ge 0$. By construction, P consumes $b - b'$ units of resource, hence it is feasible; moreover, $\bar{P}$ is the shortest among (s,t)-paths with cost $b - b'$. Now assume, by way of contradiction, that $\bar{P}$ is not efficient. As $\bar{P}$ is the shortest using $b - b'$ units of resource, there exists P' such that $\ell(\bar{P}') \le \ell(\bar{P})$ and $c(\bar{P}') < c(\bar{P})$. It must be that P' passes through $\langle t, b''\rangle$, with $b'' > b'$. We argue that, in this case, P would not be a shortest path to t^-. Indeed,

$$\ell(P') = \ell(\bar{P}') + \frac{1}{1+b''} \leq \ell(\bar{P}) + \frac{1}{1+b''} < \ell(\bar{P}) + \frac{1}{1+b'},$$

where the last expression is exactly $\ell(P)$. □

Proof of Theorem 2. To get this stronger bound, we need to modify the HL construction. The algorithm for forward hub construction is given in Algorithm 1, and for reverse hubs in Algorithm 2. Note that the two must be run sequentially, as the latter uses the nodes marked in the former. We make the forward hubs $L^+(\langle v, b\rangle)$ slightly bigger by storing, for each node the distance from $\langle v, b\rangle$ and also the *budget surplus*. Let C_i be the $(h_c, 2^{i-1})$-EPHS and $\mathcal{P}^E_{s,t}$ the efficient paths from s to t.

Observe that, whenever a node $v \in C_i$ is added, $v \in B^+_{2^i}(s)$ guarantees that at most h_c such points are needed for the whole process. Additionally, every such v is added at most $g(b)$ times in the hub of $\langle s, b\rangle$. The data requirement guarantee follows.

The bound for data requirements is $g(B)h_c \log D$, the argument is analogous to the forward case. Finally, we need to prove the cover property. Take any query $SP(\langle s, b\rangle, t^-)$ and let P be the solution. In $Lf(\langle s, b\rangle)$ there is a node v_P added by Algorithm 1. By construction, the same node v_P was added to $L^-(\langle d, 0\rangle)$. The result follows. □

Algorithm 1 Construction of forward hub

```
Input: Node $s \in V$, efficient paths $\mathcal{P}^E_{s,t}\,\forall t$, EPHS $\{C_i\}$.
Output: Forward hubs $L^+(\langle s,b\rangle)$ for $b = 0,\dots,B$ and a marked node $v_P$ for every path.
 1: Order each $\mathcal{P}^E_{s,t}$ by increasing cost and remove paths consuming more than $B$.
 2: for $t \in V \setminus s$ do
 3:     for $P \in \mathcal{P}^E_{s,t}$ do
 4:         $b \leftarrow c(P)$, $b' \leftarrow c(P')$, where $P'$ is the next path in the list ($b' = B$ if no such path).
 5:         Find the largest $i$ such that $P$ is $2^{i-1}$-efficient.
 6:         Find $v \in C_i$ hitting $P$ and mark $v$ as $v_P$.
 7:         Add $\langle v, c(P[v,t])\rangle$ to $L(\langle s,b\rangle)^+$ with distance $\ell(P[s,v])$ and surplus zero.
 8:         for $x$ between $b$ and $b'$ do
 9:             Add $\langle v, c(P[v,t])\rangle$ to $L(\langle s,x\rangle)^+$ with distance $\ell(P[s,v])$ and surplus $x-b$.
10:         end for
11:     end for
12: end for
```

Proof of Proposition 6. We extend some arguments from (Abraham et al. 2016, Theorem 8.2). Denote $S_r(v) := S^+_r(v, \mathcal{Q}) \cup S^-_r(v, \mathcal{Q})$. Observe that, for fixed $v \in V$, the set of paths $S_r(v)$ admits a hitting set of size h, namely $H_{v,r} \subseteq V$ with $|H_{v,r}| \leq h$ guaranteed by definition.

Algorithm 2 Construction of reverse hub

Input: Node $t \in V$, efficient paths $\mathcal{P}^E_{s,t} \, \forall s$, marked nodes and EPHS C_i.

Output: Backward hub $L^-(\langle t, 0\rangle)$.

```
1: Order each $\mathcal{P}^E_{s,t}$ by increasing cost and remove paths consuming more than $B$.
2: $L^-(\langle t,0\rangle) \leftarrow \varnothing$
3: for $s \in V \setminus t$ do
4:     for $P \in \mathcal{P}^E_{s,t}$ do
5:         Find the largest $i$ such that $P$ is $2^{i-1}$-efficient.
6:         Take $v$ as the marked node $v_P$.
7:         Add $\langle v, c(P[v,t])\rangle$ to $L^-(\langle t,0\rangle)$ with distance $\ell(P[v,t])$.
8:     end for
9: end for
```

If the minimum size of a set system is s and the VC-dimension is d, then the algorithm in (Even et al. 2005) obtains, in polynomial time, a hitting set of size at most $O(sd\log(sd))$. In particular, we can use the algorithm to obtain a set $F_{v,r} \subseteq E$, of size at most $h' = O(h\log(h))$, hitting the set $S_r(v)$.

Assume for now that we know the value of h. Note that the value h' can be computed from h and the guarantee given by the oracle, i.e., the constant inside the big-O. We construct the (h', r)-LSHS iteratively. At each iteration i we maintain the following invariant: C_i hits every path in $\mathcal{Q}_r$. In an iteration we check if C_i is locally sparse, if not, we strictly reduce the cardinality of C_i while maintaining the invariant. Start with $C_0 = V$. Let $B_{2r}(v) := B^+_{2r}(v) \cup B^-_{2r}(v)$. Assume $v \in V$ is such that $|B_{2r}(v) \cap C_i| > h'$ and let $C_{i+1} := (C_i \setminus B_{2r}(v)) \cup F_{v,r}$. The cardinality strictly decreases and we only need to check the invariant. Consider the paths hit by nodes removed in C_i, this set is

$$\{Q \in \mathcal{Q}_r : Q \cap C_i \cap B_{2r}(v) \neq \varnothing\} \subseteq \{Q \in \mathcal{Q}_r : Q \cap B_{2r}(v) \neq \varnothing\} \subseteq S_r(v).$$

Since $F_{v,r}$ hits $S_r(v)$, the proof is completed.

If we do not know the value of h, we can do a doubling search for h'. Indeed, if the guess of h' is low, then at some point it could be that $|F_{v,r}| > h'$, then we double h' and restart the process. □