Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

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: Distance Oracles; Highway Dimension; Provable Guarantees.

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 queries admit polynomial-time algorithms (in particular, the famous Dijkstra's algorithm), while CSP computation 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 ([Samitha] Added relatively here because poor for SP is quite fast compared to other problems.) 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* ([Samitha] changed to probabilistic because robust may imply worst case.) travel-time estimates: if $\mathcal{P}_{s,t}$ denotes the set of (s,t)-paths and ℓ the (possibly random) length of an edge or path, given s,t and parameters p,δ, we want to find an $P \in \mathcal{P}_{s,t}$ 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 ([Samitha] There is also no polynomial time algorithm for doing this even if you had a black box for solving the convolutions.). However, if we condition on 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 $\mathbb{P}(\sum_e \ell_e > \mathbb{E}[\sum_e \ell_e] + \delta) \leq \sum_e \mathbb{V}\mathrm{ar}(\ell_e)/\delta^2$. Using this, we can reformulate the robust trip-time estimation problem ([Samitha] This is not a trip-time estimation problem, but a path optimization problem. Also, I would once again not use robust for the reasons mentioned above.) 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. 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., accuracy of ETA is more important than optimality.

Another problem that can be modelled 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 and edge-lengths, and a budget upper bound, we want to use preprocessing to create a data-structure supporting arbitrary source-destination-budget queries; moreover, we want 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. 2013, 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 an be related under an additional partial witness condition* (Definition 6); this implies that in settings where SP computation is scalable, we can 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, we introduce the SP and CSP problems, and extend the notion of the HD (as defined in (Abraham et al. 2013)) 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 mention before, our work is inspired by the recent developments in shortest path algorithms (Abraham et al. 2013, 2011, 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. 2013, 2010) (this was experimentally confirmed in (Abraham et al. 2011), (Bast et al. 2016, Figure 7)). Finally, the HD-based bounds 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.

Finally, 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 avoided 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.

2. Preliminaries

We consider a directed graph $G = (V, E)$, where each edge $e \in E$ has an associated *length* $\ell(e) \in \mathbb{N}_+$, and *cost* $c(e) \in \mathbb{N}_+ \cup \{0\}$. For each node v, we denote its degree $\Delta(v)$ as the sum of the in-degree and out-degree, and define the *maximum degree* $\Delta := \max_v \Delta(v)$. 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.

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. Note that any path P with more than one node has length at least 1 (since we assume lengths are integers). For $s,t \in V$, the distance from s to t, denoted $\text{dist}(s,t)$, is the smallest length among all paths $P \in \mathcal{P}_{s,t}$. 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 denote the shortest (s,t)-path (if it exists) ([Samitha] Do any of the results require the graph to be strongly connected?) as $P(s,t)$, and denote the set of all shortest paths in G as $\mathcal{P}^*$. Finally, we define $D := \max_{P \in \mathcal{P}^*} \ell(P)$ to be the diameter of G.

Our goal is to develop a data-structure to answer *Constrained Shortest-Path* (CSP) queries efficiently: Given a source-terminal pair s,t and a budget b, we want to return a path P solving

$$\min\{\ell(P) : c(P) \leq b, P \in \mathcal{P}_{s,t}\}.$$

We define $\text{dist}(s,t|b)$ to be the minimum of this problem. If there is no feasible solution, we define $\text{dist}(s,t|b) = \infty$. 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 $\ell(P') \leq \ell(P)$ and $c(P') \leq c(P)$ with at least one inequality strict.

We denote the set of all efficient paths as $\mathcal{P}^E$, and define the *Pareto frontier* ([Samitha] of efficient paths?) from s to t as $\mathcal{P}_{s,t} \cap \mathcal{P}^E$. Observe that every subpath of an efficient path is also efficient (if not, we could improve the path by replacing the subpath). For $r > 0$ and $v \in V$, we define the *forward and reverse balls of radius* r by $B_r^+(v) := \{u \in V : \text{dist}(v,u) \leq r\}$ and $B_r^-(v) := \{u \in V : \text{dist}(u,v) \leq r\}$, and also define $B_r(v) := B_r^+(v) \cup B_r^-(v)$. Finally, a graph G is said to have a *doubling dimension* 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.

2.1. Hitting sets and the highway dimension

We now define some additional network primitives, which we need to parametrize the performance of our CSP algorithms. In particular, we use the *highway dimension*, introduced by (Abraham et al. 2013, 2010) to parametrize shortest-path computations in undirected graphs. A few of our results are generalizations of those in (Abraham et al. 2013); the technical challenges of these extensions may not be clear to a non-expert reader, thus we differ all discussions on the matter to Section A ([Samitha] This should be Appendix A). Very broadly speaking, Abraham et al. (2013) deals only with undirected graphs and shortest paths, whereas our approach covers directed graphs and general sets of paths.

We define a *path system* $\mathcal{Q}$ as any collection of paths. 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 $Q \in \mathcal{Q}$. 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 $\mathcal{Q}$. Hitting sets are useful for compressing path systems. 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 2 (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 $v \in V$, $|B_{2r}(v) \cap C| \leq h$.

As we discuss in Section 2.2, 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, 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-neighbourhood* with respect to a path system $\mathcal{Q}$ is $S_r^+(v,\mathcal{Q}) := \{Q \in \mathcal{Q}_r : \operatorname{dist}(v,Q) \leq 2r\}$ and similarly $S_r^-(v,\mathcal{Q}) := \{Q \in \mathcal{Q}_r : \operatorname{dist}(Q,v) \leq 2r\}$ is the reverse neighbourhood. 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 3 (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. 2013, 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.

2.2. 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 $L^+(v)$ and reverse hubs $L^-(v)$. We also store $\text{dist}(v,w)\,\forall\, w \in L^+(v)$ and $\text{dist}(u,v)\,\forall\, u \in L^-(v)$. The hub labels are said to satisfy the *cover property* if, 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 graph (G,ℓ) admits a collection of sets $\{C_i : i = 1, \ldots, \log D\}$, 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. 2013, Theorem 5.1) for the undirected case.

Proposition 2. *For (G,ℓ), given a multi-scale LSHS collection $\{C_i : i = 0, \ldots, \log D\}$, 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, let i be such that $2^{i-1} < \ell(P(s,t)) \leq 2^i$. 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 SP of length $\geq 2^{i-1}$). □

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 $h' = O(h\Delta \log(h\Delta))$. 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 now turn to the problem of constructing hub labels for constrained shortest-paths queries and, in particular, for computing efficient paths in G. We want to develop a data-structure that supports fast queries for *efficient paths*.

In Section 2.2 we discussed that, if a graph G has HD h, we can simultaneously bound, the preprocessing time, storage requirements and query time for constructing hub labels as functions of h. 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 4 (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 ([Samitha] *B* is not defined prior to being used here.) 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.

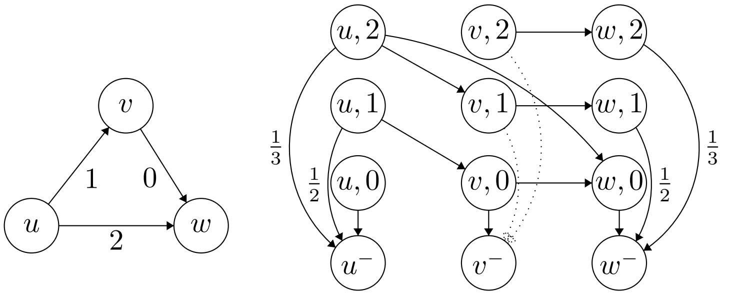

Figure 1 Example of a graph augmentation: The original graph G has all paths of unit length, and costs as labelled on the edges. In the augmented graph G^B, the labels represent the edge lengths (unlabelled edges have length 1). Note the additional edges from (w,b) to the sink node w^- (and similarly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

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 5 (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 $\ell(\langle v,b\rangle,\langle w,x\rangle) := \ell(vv')$.

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.

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

Proof. 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 $\langle t,b'\rangle$, $b' \geq 0$. 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 $\ell(\bar{P}') \leq \ell(\bar{P})$ 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 $\ell(P)$. □

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 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

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

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

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

We know that $|B^+_{2r}(s) \cap C| \leq h_c$, therefore $|B^+_{2r}(\langle s, b\rangle) \cap C^B| \leq h_c b \leq h_c B$. 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 $H := H_{v,r} \times \{0, 1, \ldots, B\}$. 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 $\text{dist}(\langle v, b\rangle, P) \leq 2r$, it holds $\text{dist}(v, \bar{P}) \leq 2r$, 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 2.2). 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 2.2; 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, such that their preprocessing time and storage is 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 critical observation in Abraham et al. (2013) is that the set system of *unique* shortest paths has a VC-dimension of 2. Directed or non-shortest paths break the arguments in Abraham et al. (2013), so we need a different formulation of the set system. We recall the concept of VC-dimension. A set system $(X,\mathcal{X})$ is a ground set X together with a family $\mathcal{X}\subseteq 2^X$. A set $Y\in\mathcal{X}$ is shattered if $\{Z\cap Y : Z\in\mathcal{X}\}=2^Y$. If d is the smallest integer such that no $Y\in\mathcal{X}$ with $|Y|=d+1$ can be shattered, then this number d is the VC-dimension of $(X,\mathcal{X})$. The critical observation in Abraham et al. (2013) is that the set system of *unique* shortest paths has a VC-dimension of 2, but their argument does not hold in directed graphs or general path systems.

Let $\mathcal{Q}$ be any path system. We can obtain a set system with small VC-dimension by considering the ground set as E (instead of the usual choice of V), and mapping a path $Q=e_1e_2\dots e_k$ to $\pi(Q)=\{e_1,e_2,\dots,e_k\}$. Note that, since path systems contain no cycles, each set $\{e_1,e_2,\dots,e_k\}$ corresponds uniquely to one path.

Proposition 6. *Given a path system $\mathcal{Q}$, the corresponding set system $(E,\{\pi(Q) : Q\in\mathcal{Q}\})$ has VC-dimension 2.*

Note that this argument also can be used for shortest paths in undirected graphs to remove the uniqueness requirement. Finally, polynomial-time preprocessing now follows from combining Proposition 6 and Even et al. (2005). The desired result is stated in Proposition 7, we defer the proof to Appendix D.

Proposition 7. *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 $h' = O(h\Delta \log(h\Delta))$.*

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.

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 Section 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 forward HL; these marked nodes are then used to construct the reverse HL. For brevity, the complete algorithmic details and proof are deferred to Section 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 8. *There are networks with HD 1 where the CHD, and also the sparsity of an EPHS, is n.*

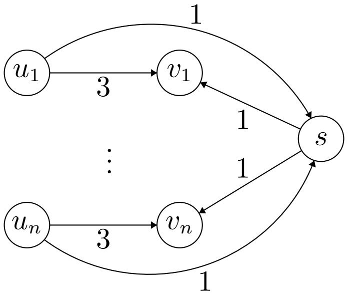

Figure 2 Graph with small HD but large CHD: The graph comprises $2n+1$ nodes, with the edge labels representing the lengths. Note that the shortest paths in the graph are of the form sv_i, u_is and u_isv_j (for all combinations i,j). Thus, the HD is 1 as $H_{v,r}=\{s\}$ is a hitting set for all these 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.

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. Moreover, the same argument shows that the sparsity of the best LSHS for $\mathcal{P}^*$ is 1, whereas the sparsity of *any* EPHS is also lower bounded by n. ([Samitha] It seems strange to be comparing the best LSHS to any EPHS.) □

Remark 1. 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 Section 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.

Intuitively, the separation between HD and CHD (or more particularly, the hub labels for computing SPs and CSPs) 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 6 (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 $\ell(Q')\geq 2^{-\beta}\ell(Q)$.

The partial-witness property and Theorem 3 provide a nice 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. As an example, consider a *multimodal* *mapping service* which gives transit routes combining walking, bus and subway, while ensuring at most k transfers. For $k = 3$, each route has at most 4 segments, which gives use the partial witness property with $\beta = 2$. Now if each individual network has small HD, then the CHD is also small (cf. Proposition 15).

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 dimension α. Note that the doubling dimension 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 $r_\beta = 2^{\log_\alpha(\alpha^{\beta-2}h)}$. Then, for any $r > 0$, given an (h, r)-LSHS, we can construct, in polynomial time, an $(\alpha^\beta h, r)$-EPHS for (G, ℓ, c). ([Samitha] Would be nice to be consistent with $\alpha^\beta h$ vs $h\alpha^\beta$. The latter is used most of the time.)*

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 dimension 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 neigborhoods 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, a simple computation shows that $|B_{2r}(v)| \leq \alpha^{\log_2 r_\beta+2} \leq h\alpha^\beta$. □

Remark 2. The existence of a β-witness is not enough to bound the CHD. Nevertheless, as we discussed in Section 2.2, the existence of (h,r)-LSHS already allows the construction of HL. Moreover, the above argument does indeed give a bound for a weaker definition of the highway dimension Abraham et al. (2010).

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 7 (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 8 (Average HD). The system $\mathcal{Q}$ has average HD h if, for every $r>0$, there exists an average (h,r)-LSHS.

From the definition is clear that average HD is a strictly weaker property than HD; in particular, as we discuss before, the existence of LSHS does not guarantee their efficient computation. Nevertheless, the above definition turns out to suffice to parametrize the average behavior of HL:

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 $h'=O(\Delta h\log(hn\Delta))$.*

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 which is an average $(h',2^i)$-LSHS. This turns out to be a minimum-cost hitting set problem. Indeed, we want to solve

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

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. Now we perform a mapping from paths to sets, where the ground set is E and paths are sequences of edges. This system has VC-dimension 2, and now the minimum is at most $h\Delta n$. We apply the algorithm in Even et al. (2005) and obtain a solution C_i with cost at most $O(h\Delta n \log(h\Delta n))$; 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, which is clearly true in road networks, and 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 9 (Stretch). An algorithm for CSP has stretch $\mathrm{St} \geq 1$ if, $\forall s,t \in V$ and $b \leq B$, outputs $\mathrm{dist}(s,t|b)$ whenever $\mathrm{dist}(s,t|b) \leq \mathrm{St}\,\mathrm{dist}(s,t)$ and outputs "infeasible" when $\mathrm{dist}(s,t|b) > \mathrm{St}\,\mathrm{dist}(s,t)$.

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

Definition 10 (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. Our main requirement is the following *bounded growth* condition, controlling the number of r-overpasses for every scale $r > 0$.

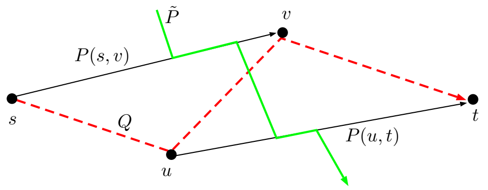

Figure 3 Edge uv here is an overpass, lying on efficient path Q (dashed red), connecting a pair of long shortest paths $P(s,v)$ and $P(u,t)$, and close to costly edges $\tilde{P}$ (solid green).

Definition 11 (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. If the bounded growth is satisfied, we can obtain, in polynomial time, hub labels for CSP queries that guarantee average query-time $O((b+1)\alpha h'\log D)$ and total storage $O(nB\cdot B\alpha h'\log D)$, where $h'=O(\Delta h\log(hn\Delta))$.*

We henceforth refer to the average HD of $\mathcal{P}^E$ as average CHD. Since the average HD is enough to obtain good preprocessing algorithms (cf. Theorem 4), an average CHD should intuitively suffice. However, to prove Theorem 5, we need to show that it is possible to obtain small average CHD under even less restrictive assumptions than the partial-witness. To do this, we first allow 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.

Definition 12 (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 now show that our requirement on $|D_r|$ guarantees a bound on the average HD.

Proposition 9. *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 $h'\leq 2\alpha^\beta h$*

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 is a hitting set for $\mathcal{Q}'_r$ and we can compute

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

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 dimension, balls of radius r have at most $\alpha^{\log_2 r+1}$ elements. Since $|D_r| \leq \phi(r)$, the result follows. □

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

Proposition 10. *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 $\ell(Q[s,v]), \ell(Q[u,t]) \geq r$. If either $Q[s,v]$ or $Q[u,t]$ ([Samitha] The directed edge is (u,v), but the discussion here talks about (s,v) and (u,t). There is a mismatch in the notation. This carries over in the next paragraph.) 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 $\frac{2-\text{St}}{\text{St}}$. 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 $\text{St}\,\ell(P(s,t)) \geq \ell(Q)$, hence $\frac{2r}{\text{St}} \leq \ell(P(s,t)) \leq \ell(P(s,v)) + \ell(Q[v,t])$ and $\frac{2r}{\text{St}} \leq \ell(P(s,t)) \leq \ell(Q[s,u]) + \ell(P(u,t))$. 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 $\frac{2-\text{St}}{\text{St}} r$. Finally, to show condition 3, we have $\ell(Q[s,v]) + \ell(Q[u,t]) = r + \ell_{uv}$ and since $\ell_{uv} < r/2$, one of $Q[s,v]$ or $Q[u,t]$ has length at most $3r/4$. 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 $3r/4$ 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. □

We are almost ready to prove that bounded growth allows to solve the CSP. The last piece is Proposition 11, the proof of which follows form similar arguments as those in Theorems 1 and 4.

Proposition 11. *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 $h_c \leq 2\alpha h$. By Proposition 10, $\mathcal{P}^E$ is weakly 1-witnessed by $\mathcal{P}^*$. It follows by Proposition 9 that the average CHD is at most $2\alpha h$ as needed. Applying Proposition 11 yields the result. □

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 13. 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

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

([Samitha] Shouldn't $\bar{P}$ be P?) 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 12. *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. (2011) 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. (2011). 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

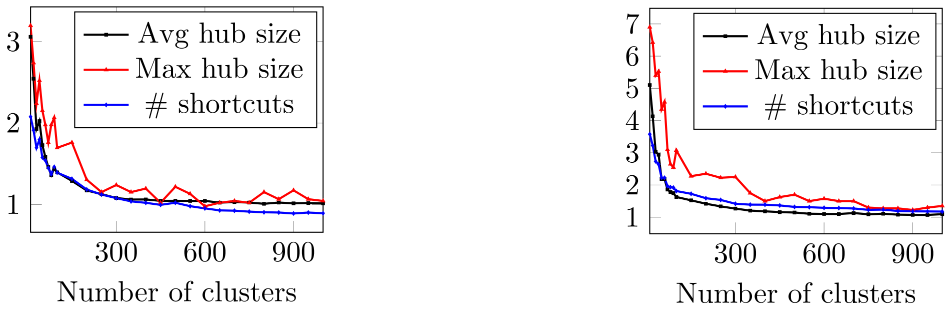

Figure 4 Performance of clustering for San Francisco (left) and Luxembourg (right). In the y-axis the quantities are normalized by the best greedy cover, i.e., using $k = n$. Note that, while the number of short-cuts is an indicator of performance, it is not perfectly correlated with the hub size.

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 $b = B, \ldots, 0$, 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. The code is 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, labelled as 'f', the query times are measured as the average of 1000 random s, t. For specific queries, labelled 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 preprocess 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.

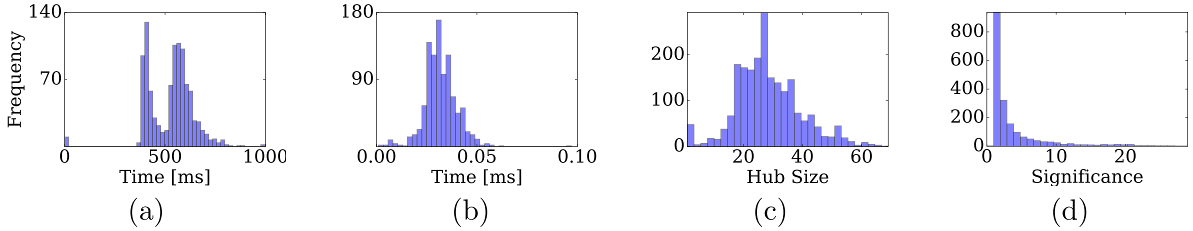

Figure 5 Histogram frontier queries for Dijkstra (a) and HL (b). Times are 1000 random pairs in San Francisco augmented with $B = 25$. Size of reverse hubs (c) and significance (d) for frontier queries in San Francisco augmented with $B = 25$.

([Samitha] Tried to align the figures a bit with the reference to the figure.)

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 belongs to many efficient paths. Figure 5 shows a histogram of these metrics.

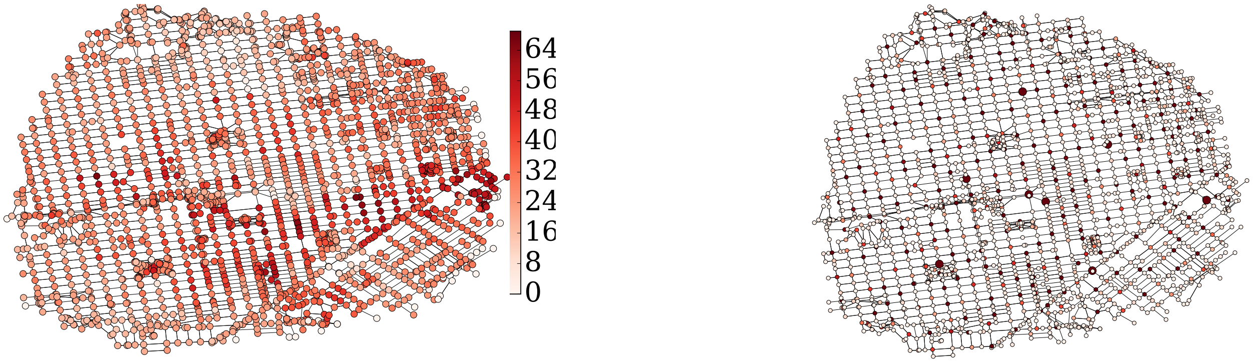

Figure 6 Heat maps for frontier queries in San Francisco augmented with $B=25$. On the left we see the hub size. Note that the size is not homogeneous, but rather we can observe clusters and neighborhoods tend to be similar. On the right the significance. The most significant nodes have been drawn bigger.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

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. ([Samitha] The numerics section is pretty short, but I guess this is okay for an OR submission.)

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

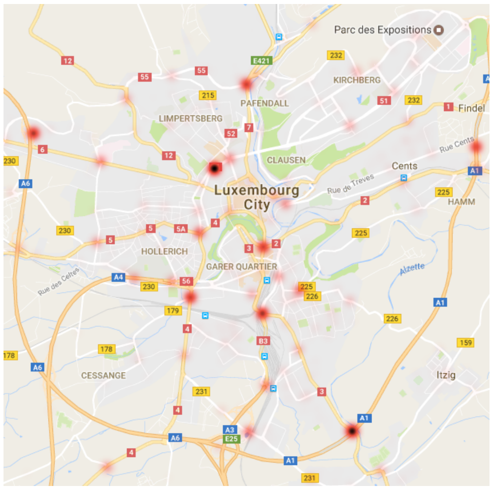

Figure 7 Heat map of significance for frontier queries in Luxembourg City augmented with $B = 25$. Notice how the highly significant nodes are in main road crossings.

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.

Appendix A: Different notions of HD

The original definition of HD was given by Abraham et al. (2010), but this is not the one we extend, but rather we work from the definition of Abraham et al. (2013).([Samitha] Last sentence has ..., but ..., but.) The latter is the same as our notion except that (i) we consider a less restrictive definition of path neighborhoods, that is appropriate for our needs, and (ii) we generalize the notion to directed graphs and general path systems. 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. (2013). In particular, unlike Abraham et al. (2013), the HD of G as per our definition is not an upper bound to the maximum degree Δ or the doubling dimension α.

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. (2013) 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.

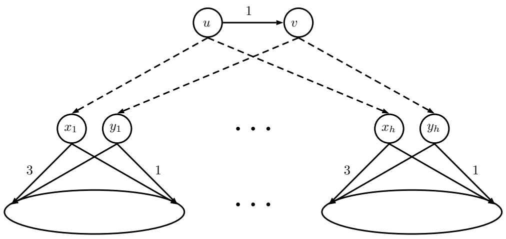

Figure 8 Example where the EPHS is much larger than the LSHS.

We now discuss how our results extend to the definition in Abraham et al. (2013), 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.

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 13. *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_iw 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

$$n = [k\text{-level } h\text{-ary tree}] + [(k-1)\text{-level } h\text{-ary tree}]$$

$$= (h^{k+1} - 1)/(h-1) + (h^k - 1)/(h-1),$$

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

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 $w \in B^-_{2\gamma}(v)$. □

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 14. *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. □

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 15. *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 $S_r(v, \mathcal{Q}) \cup S_r(v, \mathcal{Q}')$. □

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 dimension 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 $O(b\Delta\tilde{h}\log D)$ and the space requirement is $O(Bn \cdot \Delta B\tilde{h}\log D)$. 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 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 $Lf(\langle s,b\rangle)$ for $b = 0,\ldots,B$ and a marked node v_P for every path.

```
Order each $\mathcal{P}^E_{s,t}$ by increasing cost and remove paths consuming more than $B$.
for $t \in V \setminus s$ do
    for $P \in \mathcal{P}^E_{s,t}$ do
        $b \leftarrow c(P)$, $b' \leftarrow c(P')$, where $P'$ is the next path in the list ($b' = B$ if no such path).
        Find the largest $i$ such that $P$ is $2^{i-1}$-efficient.
        Find $v \in C_i$ hitting $P$ and mark $v$ as $v_P$.
        Add $\langle v, c(P[v,t])\rangle$ to $L(\langle s,b\rangle)^+$ with distance $\ell(P[s,v])$ and surplus zero.
        for $x$ between $b$ and $b'$ do
            Add $\langle v, c(P[v,t])\rangle$ to $L(\langle s,x\rangle)^+$ with distance $\ell(P[s,v])$ and surplus $x - b$.
        end for
    end for
end for
```

Proof of Proposition 7. We extend some arguments from (Abraham et al. 2013, 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 system $(E, \{\pi(Q) : Q \in S_r(v)\})$ admits a hitting set of size $h\Delta$. Indeed, we know that exists $H_{v,r} \subseteq V$, $|H_{v,r}| \le h$, hitting every path in $S^+_r(v,\mathcal{Q})$ and in $S^-_r(v,\mathcal{Q})$. The desired hitting set consists of all the edges adjacent to a node in $H_{v,r}$.

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 $\tilde{F}_{v,r} \subseteq E$, of size at most $h' = O(h\Delta\log(h\Delta))$, hitting the set system $(E, \{\pi(Q) : Q \in S_r(v)\})$.

Consider the set $F_{v,r} \subseteq V$ that contains all the endpoints of edges in $\tilde{F}_{v,r}$. It follows that $F_{v,r} \subseteq V$ can be obtained in polynomial time and is a hitting set for $S_r(v)$ of size $|F_{v,r}| \le 2h'$. Assume

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

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 $(2h', r)$-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 $|B_{2r}(v) \cap C_i| > 2h'$ 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 $|F_{v,r}| > 2h'$, then we double h' and restart the process. □

Acknowledgments

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

References

Abraham I, Delling D, Fiat A, Goldberg AV, Werneck RF (2013) Highway dimension and provably efficient shortest path algorithms. *Microsoft Research, USA, Tech. Rep* 9.

Abraham I, Delling D, Goldberg AV, Werneck RF (2011) 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, Pajor T, Müller-Hannemann M, Sanders P, Wagner D, Werneck RF (2016) Route planning in transportation networks. *Algorithm Engineering.*

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.

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).

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

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.*

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*.

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.