Computing Constrained Shortest-Paths at Scale

Alberto Vera
Cornell University
aav39@cornell.edu

Siddhartha Banerjee
Cornell University
sbanerjee@cornell.edu

Samitha Samaranayake
Cornell University
samitha@cornell.edu

ABSTRACT

We study the problem of computing constrained shortest paths at scale, motivated by the needs of modern transportation platforms for travel-time estimates with probabilistic guarantees. The use of preprocessing techniques and network augmentation has led to dramatic improvements in the speed and scalability of shortest path (SP) computations. Our work extends these ideas to CSP queries, via two main contributions:

- We adapt existing preprocessing techniques for fast SP queries (in particular, hub labels) to handle CSP queries; we also extend the *highway-dimension*, a measure of graph structure that enables SP speedup techniques, to an analogous *constrained highway dimension* for CSP queries. Finally, we show that the two notions can be related under an additional partial witness condition.
- We develop a practical algorithm for fast CSP queries, combining hub labels with an augmented graph that encodes efficient paths. We evaluate our algorithm on datasets with detailed travel-time information for San Francisco and Luxembourg, and show that our algorithms are orders of magnitude faster compared to existing techniques, and also have small per-node storage requirements, and good preprocessing times even on a single machine.

1. INTRODUCTION

We consider the fast computation of constrained shortest paths in large-scale graphs, using preprocessing and graph augmentation techniques. The Shortest-path (SP) problem is one of the canonical algorithm design problems. In recent years, however, it has been revolutionized both in terms of academic research as well as real-world applications (cf. [7, 11] for surveys). In particular, the use of preprocessing techniques and network augmentation has led to dramatic improvements in the speed and scalability of SP computation in road networks. These techniques however do not extend to constrained shortest-path (CSP) problems, and our work aims to bridge this gap.

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*. Now, given any two nodes s, t, the SP problem requires finding an $s \to t$ path of minimum length; the CSP problem takes in a further input of a budget b, and requires finding a minimum length $s \to t$ path *which moreover has a total cost less than* b. The two problems, though similar, have very different basic complexity: SP computation admits a polynomial-time algorithm (in particular, the famous Dijkstra's algorithm), while the problem of CSP computation is known to be NP-Hard [14]. That said, a standard dynamic programming approach allows CSP computation in pseudo-polynomial time when the costs are discrete (in particular, polynomial in the maximum budget), and also admits a natural scaling-based FPTAS for continuous costs (akin to the knapsack FPTAS).

Although there is a rich literature on CSP problems (cf. [14] for a survey), there is little work on studying ways of using preprocessing and graph augmentation to speed up CSP computation. Our work here aims to address this gap, motivated by the success of similar approaches for SP computation. Moreover, we believe that there is a need for such techniques in several modern applications, as we discuss next.

Applications of large-scale CSP computation. Our interest in CSP computation is motivated primarily by the requirements of transportation platforms like Lyft, Uber, Waze etc., for accurate routing and travel-time estimates with probabilistic guarantees. Fast routing and trip-time estimation, driven by fast SP engines (for example, OSRM [1]), have provided impetus to the increasing use of mobile maps. These techniques however do not make full use of available real-time traffic information; in particular, SP queries are unable to incorporate uncertainties in travel times, and hence often give inaccurate trip-time estimates in settings with high traffic uncertainty.

A natural way to incorporate travel-time uncertainty is to compute the shortest path subject to a reliability constraint. In particular, in many settings, we want *robust* travel time estimates: formally, we require that given s, t and parameters p, δ, a routing algorithm returns an (s, t)-path P minimizing $\mathbb{E}[\ell(P)]$, subject to $\mathbb{P}[\ell(P) > \mathbb{E}[\ell(P)] + \delta] \le p$. Even assuming that travel times on different road segments are independent, computing this exactly for general distributions is expensive due to the need for computing convolutions of distributions. However, for uncorrelated travel-times, we have $\mathbb{P}[\sum_e T_e > \mathbb{E}[\sum_e T_e]) + \delta] \le \sum_e \mathbb{V}\text{ar}(T_e)/\delta^2$ by Chebyshev's inequality; using this, we can replace the robust trip-time estimation problem with the following

$$\min_{P\in\mathcal{P}_{s,t}} \sum_{i\in P} \mu_i \qquad \text{s.t.} \qquad \sum_{i\in P} \sigma_i^2 \le \delta^2 p.$$

This is now a CSP problem. Note that the solution, though not necessarily optimal, always respects the reliability constraint – this is often more critical for practical applications.

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. In this case, 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 [10] (for routing with fare evasion, where q_e is the probability of encountering an inspector, and p the penalty), where the authors suggest using a CSP formulation, wherein the non-linear objective is replaced by a linear constraint by taking logarithms.

Finally, another class of problems which is related to the CSP is that of routing under *label constraints* [6, 21], wherein we want shortest routes which satisfy certain properties (for example, those that avoid toll roads). The main idea in such problems (as in the CSP) is to use an appropriate augmented graph that converts feasible shortest paths to shortest paths. Our results extend to these applications as well.

1.1 Our Contributions

We aim to address the following overarching questions:

1. How can we use preprocessing and graph augmentation techniques to speed up CSP computation.
2. How can we give preprocessing/storage/query time guarantees for such techniques.

For the SP problem, there was significant initial development on the first question, leading to many different techniques [11], which worked well in practice, but had no performance guarantees. Abraham et al. [2, 4] then proposed the *Highway Dimension* (HD), a graph structural metric, which they showed could parametrize the preprocessing, storage and query time of several of these heuristics (hub labels, contraction hierarchies, etc.). Our work adopts a similar program; in particular, our contributions are as follows:

- **Theoretical contributions**: We extend the HD to directed graphs and general path systems (Section 2.2), and define an analogous notion of a *constrained highway dimension* (CHD) for Pareto efficient paths (Section 3). We then construct an augmented graph (Section 3.1) under which efficient paths are mapped to shortest paths, and use it to develop hub labels for CSP queries, whose complexity is parameterized by the CHD (Section 3.2). Finally, we show that although the CHD can be much bigger than the HD in general, the two can be related under an additional *partial witness condition* (Section 3.3). This provides some justification for the use of preprocessing techniques for CSP, as it suggests that they should work well in settings where SP computation admits good speedup techniques.
- **Practical contributions**: Although the theoretical results justify the use of preprocessing techniques for CSP, they are impractical for real networks. We show how we can adapt our theoretical ideas to develop a practical construction technique for hub labels that support CSP queries (Section 4). We then evaluate our algorithm on datasets with detailed travel-time information for San Francisco and Luxembourg (Section 4.2). Our experiments show that our hub labels support query times which are orders of magnitude faster than existing techniques (without preprocessing), have small per-node storage requirements, and good preprocessing times even on a single machine.

1.2 Related work

As we mention before, our work is inspired by the recent developments (both theoretical and practical) in shortest path algorithms [2, 3, 4, 11, 15, 18]; refer [7] for an excellent survey of these developments.

The problem of finding robust shortest paths is closely related to the stochastic on-time arrival (SOTA) problem [13]. The SOTA problem with Gaussian travel times was solved via exhaustive search [20]. The optimal policy under discrete distributions was given in [23], and pre-processing policies were considered in [22]. Finally, [16] provides an approximation technique based on discretization for DAGs.

The primary speedup technique we use in our work is that of hub labels (HL). This was introduced first in [9]. More recently, HL was shown to admit the best query time bounds in low HD graphs [2, 4], and this observation was experimentally confirmed in [3] (see also Figure 7 in [7]). Finally, the HD-based bounds for hub labels was shown to be tight in [5, 25], and it was also shown that finding optimal hub labels is NP hard.

There is an extensive literature on CSP problems, which is well surveyed in [14]; most of this work however considers only query-time techniques without preprocessing. Our graph augmentation technique is similar to that proposed in [8], which is based on a dynamic programming approach.

Finally, a related class of problems to CSP is that of shortest paths under label constraints, first introduced in [6]. Contraction hierarchies were adapted to solve a restricted class of such problems in [21], where the aim was to find shortest paths that avoided certain labels (for example, toll roads/ferries). Our work is similar in spirit to this paper. Note however that the label-constrained problem in [21] is a concatenation of parallel shortest-path problems, and involves only local constraints. In contrast, the CSP is NP-hard as it involves global constraints on paths; consequently, a small number of edges with costs can affect all nodes. Thus, though the techniques in [21] do not apply to our setting, our theoretical results do in fact shed light on why contraction hierarchies work well in their setting.

2. PRELIMINARIES

2.1 Basic Setting

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

A simple *path* P in G is an acyclic sequence of nodes $u_1u_2u_3\dots u_k$ with $u_iu_{i+1} \in E$. For any source-terminal pair $s, t \in V$, we denote by $\mathcal{P}_{s,t}$ the set of all simple paths from s to t. Throughout this work, we only consider simple paths, which for brevity, we henceforth refer to as paths.

The length $\ell(P)$ is the sum of edge lengths in P. Note also that any path P with more than one node has length at least 1 (since $\ell : E \to \mathbb{N}_+$). For $s, t \in V$, the distance from s to t, denoted $\text{dist}(s,t)$, is the smallest length among all paths $P \in \mathcal{P}_{s,t}$. The distance from a node v to a path P, denoted $\text{dist}(v, P)$, is measured as the minimum distance from v to a node $w \in P$. The distance to v, $\text{dist}(P, v)$, is defined analogously.[1] 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 $\mathcal{P}^*$. Finally, we define $D = \max_{P\in\mathcal{P}^*} \ell(P)$ to be the diameter of G.

Analogous to length, we define the cost $c(P)$ as the sum of edge costs on path P. Our goal is to develop a data-structure that can solve the following *Constrained Shortest-Path* (CSP) problem: Given a source-terminal pair s, t and a budget b, we want to return a path P that solves

$$\min_{P\in\mathcal{P}_{s,t}} \quad \ell(P) \quad \text{s.t.} \quad c(P) \le b$$

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 (with different feasible costs), 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 $\mathcal{P}^E$, and define the *Pareto frontier* 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 $B^+_r(v) := \{u \in V : \text{dist}(v,u) \le r\}$ and $B^-_r(v) := \{u \in V : \text{dist}(u,v) \le r\}$, and also define $B_r(v) := B^+_r(v) \cup B^-_r(v)$. 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 $B_r(w)$ of radius r.

2.2 Hitting sets and the highway dimension

We now define the notions of hitting sets for path systems, and the highway dimension (HD), which we use to parametrize the performance of our algorithms. The highway dimension was introduced by Abraham et al. [2, 4] for undirected graphs and the shortest-path set system. Our presentation closely follows these works. However, we need to extend the definitions to directed graphs and general path systems for our purposes.

We define a *path system* $\mathcal{Q}$ as any collection of paths. Given a set $C \subseteq V$ and a path Q, we say that C *hits* 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 $Q \in \mathcal{Q}$. Also, 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 $v \in V$, $|B_{2r}(v) \cap C| \le h$.

The existence of (h, r)-LSHS enables the compression of path system $\mathcal{Q}$ via the construction of *hub labels* (cf. Section 2.3). However, this existence does not guarantee the ability to efficiently compute it. To address this, Abraham et al. [2] introduced the notion of the *highway dimension* – a property of the graph which ensures both existence and efficient computation of LSHS.

To define the highway dimension, we first need two additional definitions: for $v \in V, r > 0$, we define the *forward and reverse path-neighborhoods* with respect to path system $\mathcal{Q}$ as:

$$S^+_r(v, \mathcal{Q}) := \{Q \in \mathcal{Q}_r : \text{dist}(v, Q) \le 2r\},$$
$$S^-_r(v, \mathcal{Q}) := \{Q \in \mathcal{Q}_r : \text{dist}(Q, v) \le 2r\}.$$

As before, we have $S_r(v, \mathcal{Q}) = S^+_r(v, \mathcal{Q}) \cup S^-_r(v, \mathcal{Q})$. Now we can define the highway dimension (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, for every $r > 0$, and for every $v \in V$, there exists a set $H_{v,r} \subseteq V$ such that $|H_{v,r}| \le h$ and $H_{v,r}$ is a hitting set for $S^+_r(v, \mathcal{Q}) \cup S^-_r(v, \mathcal{Q})$.

The HD was defined in [2] for undirected graphs and the shortest-path system $\mathcal{P}^*$; the above definition is the natural analog for directed graphs and general path systems. As shorthand, we refer to the HD of (G, ℓ) as that of $\mathcal{P}^*$. To see that $HD \le h$ is a more stringent requirement than the existence of an (h,r)-LSHS C, note that $C \cap B_{2r}(v)$ need not hit all the paths in $S_r(v, \mathcal{Q})$. However, if G has $HD \le h$, then this guarantees the existence of a (h, r)-LSHS.

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

This follows directly from Theorem 4.2 in [2], which showed it for the shortest-path system. More importantly, as we discuss in Section 3.2.2, if G has HD$\le h$, then this permits efficient computation of locally-sparse hitting sets. In particular, for any r, a simple greedy algorithm gives an $(O(h \log n), r)$-LSHS, and a more complex algorithm (which is still poly-time) gives an $(O(h\Delta \log h\Delta), r)$-LSHS.

Finally, we note that for pedagogical reasons, Definition 3 differs slightly from [2]. We consider a less restrictive definition of path neighborhoods (albeit for general path systems) that is appropriate for our needs. Consequently the highway dimension as returned by our definition is smaller than that of [2].[2] We discuss the relation between the two definitions in detail in Appendix A.

[1]Note that $\text{dist}(P, v)$ and $\text{dist}(v, P)$ need not be the same as the graph is directed.

[2]In particular, unlike [2], the HD of G as per our definition is not an upper bound to the maximum degree Δ or the doubling dimension α.

2.3 Shortest-Paths via Hub Labels

Two of the most successful data-structures enabling fast shortest path queries at scale are *contraction hierarchies* (CH) [15] and *hub labels* (HL) [9]. These are general techniques which always guarantee correct SP computation, but have no uniform storage/query-time bounds for all graphs. However, Abraham et al. [2] show that for a graph with HD $\leq h$, the preprocessing time, storage requirements and query time of particular implementations of the CH and HL algorithms can be bounded solely in terms of h, the maximum degree Δ and the diameter D. We now explain the construction for HL for directed graphs; for the construction of CH refer to our technical report [24]. HL was also shown to admit the best query-time guarantees in [2], consequently, we adapt the HL technique for scaling CSP computations in Section 3.

The basic HL technique is as follows: Every node v is associated with *hub labels*, $L(v) = \{L^+(v), L^-(v)\}$, comprising of a set of forward hubs $L^+(v)$ and reverse hubs $L^-(v)$. We also store $\mathrm{dist}(v,w), \mathrm{dist}(u,v)$ for every $w \in L^+(v), u \in L^-(v)$. 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 $\mathrm{dist}(s,t)$ by searching for the minimum value of $\mathrm{dist}(s,w) + \mathrm{dist}(w,t)$ 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 second-last node in $P(w,t)$ for each $w \in L^-(t)$, we can also recover the shortest path. Now, each time we run a HL query, at least one new node $w \in P(s,t)$, $w \neq s,t$, is returned; we can then recurse to find subpaths $P(s,w)$ and $P(w,t)$. Note that we need to store this extra information, as otherwise, we could have $L^+(s) \cap L^-(t) = \{s\}$. Let $L_{\max} := \max_v |L(v)|$ be the size of the maximum HL. The per-node storage requirement is $O(L_{\max})$, while the query time is $O(L_{\max}\ell(P(s,t)))$.

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 [2] (Theorem 5.1) for the undirected case.

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

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

$$L^+(v) := \bigcup_{i=1}^{\log D} C_i \cap B^+_{2^i}(v)\ ,\ L^-(v) := \bigcup_{i=1}^{\log D} C_i \cap B^-_{2^i}(v).$$

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 $2^{i-1} < \ell(P(s,t)) \leq 2^i$. Now any point in the path belongs to both $B^+_{2^i}(s)$ and $B^-_{2^i}(t)$, and hence $C_i \cap P(s,t)$ (which is not empty since C_i hits all SP of length $\geq 2^{i-1}$) is then in both hubs, which shows the result. □

Finally, we need to compute the desired multi-scale LSHS in polynomial time. As we mentioned before, a greedy algorithm returns a $O(h \log n)$ approximation to any (h,r)-LSHS, however, if the HD $\leq h$, this can be improved to obtain a $O(h\Delta \log(h\Delta))$ approximation. This argument, which extends Corollary 7.3 in [2], is presented in Section 3.2.2. A more subtle point is that the resulting algorithm, even though admitting a polynomial time guarantee, 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. Recall we are interested in settings where edges contain both lengths and costs, and wherein all edge-costs are non-negative integers. Now, we want to develop a data-structure that supports fast queries for *efficient paths*.

In Section 2.3 we discussed that, if a graph G has HD $\leq h$, we can simultaneously bound, as functions of h, the preprocessing time, storage requirements and query time for constructing HL. This suggests that for the construction of provably efficient HL for the CSP problem, one needs to define 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 every shortest path is efficient, thus $h_c \geq h$, where h is the HD of G under shortest paths.

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 $g(b)$ is measure of 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*, i.e. it satisfies

1. All paths starting from a given node are feasible, i.e., they satisfy the budget constraint in G.
2. Efficient paths become shortest paths.
3. Inefficient paths are not shortest paths.

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

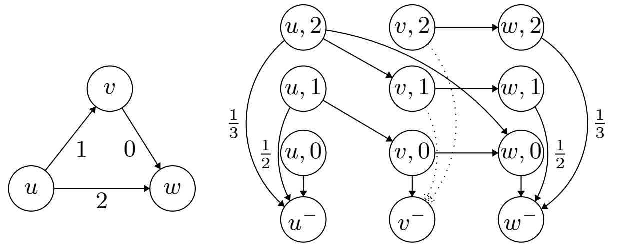

Figure 1: Example of a graph augmentation: The original graph G has all paths of unit length, and costs as labeled on the edges. In the augmented graph G^B, the labels represent the edge lengths (unlabeled edges have length 1). Note the additional edges from (w, b) to the sink node w^- (and similarly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

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. Given a network (G, ℓ) and $B \in \mathbb{N}$, the augmented version $G^B = (V^B, E^B, \ell)$ is defined by

$$V^B \coloneqq \{\langle v, b\rangle : v \in V, b = 0, 1, \dots, B\} \cup \{v^- : v \in V\},$$
$$E^B \coloneqq \{(\langle v, b\rangle, \langle v', b'\rangle) : vv' \in E, b' = b - c_{vv'}, b' \geq 0\} \cup E^B_-,$$
$$E^B_- \coloneqq \{(\langle v, b\rangle, v^-) : v \in V, B \geq b \geq 0\}$$

and with length function $\ell(\cdot)$,

$$\ell(\langle v, b\rangle, v^-) \coloneqq \frac{1}{b+1}\ , \ \ell(\langle v, b\rangle, \langle v', b'\rangle) \coloneqq \ell(vv').$$

Paths in G^B are mapped to paths in G in the intuitive way, by removing the budget labels and sink nodes.

Definition 6. Let P be a path in G^B. If P is of the form $\langle v_1, b_1\rangle \dots \langle v_k, b_k\rangle$, then the projection of P, denoted $\bar{P}$, is the path $\bar{P} \coloneqq v_1 v_2 \dots v_k$. Analogously, if P is of the form $\langle v_1, b_1\rangle \dots \langle v_k, b_k\rangle v_k^-$, then $\bar{P} \coloneqq v_1 \dots v_k$.

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 $\ell(P)$. □

We note that the augmented graph is similar to existing techniques for solving CSP via dynamic programming [8]. Such an approach, however, requires that all costs are nonzero, thus the augmented graph is a DAG. In contrast, we allow edge costs to be zero, and therefore, our augmented graph may contain cycles. The following construction will depend on LSHS for the system $\mathcal{P}^E$, we call such object an efficient path hitting set (EPHS).

The above result now 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 project 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 $\mathcal{P}^E$ can be used to obtain a hitting set for $\mathcal{P}^B$, but, since the augmented graph has more nodes, the sparsity increases.

Proposition 4. *If the path system $\mathcal{P}^E$ admits (h_c, r)-EPHS, then $\mathcal{P}^B$ admits $(h_c B, r)$-LSHS.*

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

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

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

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

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 Equation (1) 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. Moreover, we can also relate the highway dimensions of the path systems $\mathcal{P}^E$ and $\mathcal{P}^B$ (Note: this does not follow from above, 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 $H \coloneqq H_{v,r} \times \{0, 1, \dots, B\}$. 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 $\text{dist}(\langle v, b\rangle, P) \leq 2r$, 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 $S^-_r(\langle v, b\rangle, \mathcal{P}^B)$. □

In the next section, we discuss how the above augmented graph can be used to construct hub labels for answering CSP queries. We note, however, that this technique is primarily a theoretical proof-of-concept, that shows the CHD (and in fact, the HD of G under an additional condition (cf. Section 3.3) to be a good parametrization for the complexity of HL construction. In practice (cf. Section 4.1), we use a similar idea of an augmented graph, but with a construction more amenable for implementation.

3.2 Hub Labels For CSP

We now describe how to use the augmented graph G^B to get hub labels that allow us to find efficient paths for any query $SP(\langle s, b\rangle, t^-)$. This is similar to HL construction for shortest-paths (cf. Section 2.3). A subtle difference is that we are only interested in paths ending in a sink node.

We associate each node $\langle v, b\rangle$ with a forward hub label $L^+(\langle v, b\rangle) \subseteq V^B$, and *only sink nodes* u^- with a reverse hub $L^-(u^-) \subseteq V^B$. The cover property must be satisfied for every $\langle s, b\rangle$ and t^-. Finally, we can augment the hub labels with the next-hop node, and compute the entire path recursively. To reconstruct the path, we can proceed similarly as in Section 2.3.

3.2.1 Query Time and Data Requirements

Putting things together, we now show how we can construct hub labels for answering CSP queries, such that their preprocessing time and storage is parameterized by the CHD h_c. Note that we use the augmented graph primarily as a theoretical construction; from a computational point of view, we can obtain the HL using only an EPHS of the original graph.

Theorem 1. *For graph (G, ℓ, c), given a multi-scale EPHS $\{C_i : i = 0, 1, \ldots, \log D\}$, where C_i is an $(h_c, 2^{i-1})$-EPHS, we can construct hub labels for answering CSP queries such that, for every $u \in V$ and $b \geq 0$, $|L(\langle u, b\rangle)^+| \leq bh_c \log D$ and $|L(u^-)^-| \leq Bh_c \log D$.*

Proof. Create C_i^B as in Equation (1) and define

$$L(\langle v, b\rangle)^+ \coloneqq \bigcup_{i=1}^{\log D} C_i^B \cap B_{2^i}^+(\langle v, b\rangle)$$

$$L(u^-)^- \coloneqq \bigcup_{i=1}^{\log D} C_i^B \cap B_{2^i}^-(u^-).$$

The cover property is proved in an identical manner as in Proposition 2; we now bound the hub size. For a reverse hub we use that

$$\begin{aligned} B_{2^i}^-(t^-) &= \{\langle s, x\rangle : \exists P \in \mathcal{P}_{s,t}, c(P) = x, \ell(P) \leq 2^i\} \\ &\subseteq B_{2^i}^-(t) \times \{0, 1, \ldots, B\}. \end{aligned}$$

Thus, $B_{2^i}^-(t^-) \cap C_i^B \leq Bh_c$. For forward hubs, the size follows from observing that $|C_i^B \cap B_{2^i}^+(\langle v, b\rangle)| \leq bh_c$. □

This immediately gives us the following corollary for the storage and query time.

Corollary 1. *Using the HL given by Theorem 1, we can implement queries for s, t, b in time $O(bh_c \log D)$. The total space requirement is $O(nB \cdot Bh_c \log D)$.*

3.2.2 Preprocessing

Next, we show how the preprocessing requirements for HL construction can be parameterized in terms of h_c. For this, we need to modify similar arguments for shortest paths in [2] to handle efficient paths.

The critical observation in [2] is that the set system of *unique* shortest paths has a VC-dimension of 2. [3] To see this, note that if the shortest path between a and c either passes through b or not – hence $\{a, b, c\}$ and $\{a, c\}$ can not be in the system and hence $\{a, b, c\}$ cannot be shattered. This does not hold in directed graph, as both paths abc and ca may be shortest paths. However, we can overcome this by considering the ground set as E instead of V, and mapping a path $e_1e_2 \ldots e_k$ to $\{e_1, e_2, \ldots, e_k\}$. Note that each $\{e_1, e_2, \ldots, e_k\}$ corresponds uniquely to one path (since we consider acyclic paths). Let $\pi(Q)$ denote the set of edges in a path Q.

[3] Formally, paths $v_1v_2 \ldots v_k$ are mapped to sets $\{v_1, \ldots, v_k\}$; note if G is undirected, each subset represents exactly two paths, the (s, t)-path and (t, s)-path. Therefore, a hitting set for this set system corresponds to a hitting set for $\mathcal{P}^*$.

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 a similar argument as Theorem 8.2 in [2]; we defer the proof to Appendix D.

Proposition 7. *If $\mathcal{Q}$ has HD h, then in polynomial time we can obtain 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 value of budget. 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. *Suppose (G, ℓ, c) admits a function $g : \mathbb{N} \to \mathbb{N}$ that satisfies for every $s, t \in V$, $b \in \mathbb{N}$,*

$$\left|\left\{P \in \mathcal{P}_{s,t}^E : c(P) \leq b\right\}\right| \leq g(b).$$

Then, we can construct hub labels of size $O(g(B)h_c \log D)$, and answer queries with budget b in time $O(g(b)h_c \log D)$.

The proof depends on a different technique for constructing HL (refer 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 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 above sections thus 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 [5]. 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 in general, the CHD (and in fact, 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 any EPHS, is n.*

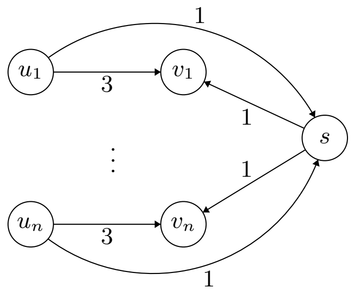

Figure 2: Graph with small HD but large CHD: The graph comprises $2n+1$ nodes, with the edge labels representing the lengths. Note that the shortest paths in the graph are of the form sv_i, u_is and u_isv_j (for all combinations i, j). Thus, the HD is 1 as $H_{v,r} = \{s\}$ is a hitting set for all these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

Remark 1. One criticism of the graph in Figure 2 is that it has a maximum degree of n; the result however continues to hold even for bounded degree graphs. In Appendix A, where we discuss alternate notions of HD, we give a more involved example with bounded degrees that exhibits the same separation between HD and CHD for even stronger notions of HD.

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 notion of a *partial witness*

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

Given this definition, we can now ask if the hub labels for computing SPs and CSPs can be related in settings where the shortest-path set 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 also 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 really want this property only for 'long-enough' paths.

We now show that if costs are such that the graph indeed has the partial witness property (for long-enough paths) for some 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.

Theorem 3. *Assume that G is α-doubling and $\mathcal{P}^*$ is a β-witness for $\mathcal{P}^E_r$ for every $r \geq \frac{\log(\alpha^\beta h)}{2\log\Delta}$. Then, given an (h,r)-LSHS for any $r > 0$, we can construct (in polynomial time) an $(\alpha^\beta h, r)$-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 $r \geq \frac{\log(\alpha^\beta h)}{2\log\Delta}$. Take C as 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 $r < \frac{\log(\alpha^\beta h)}{2\log\Delta}$. It is no longer true that the efficient paths are witnessed, but now the neigborhoods are small. 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 and the fact that the lengths are integers, it follows that $|B_{2r}(v)| \leq \Delta^{2r} \leq \alpha^\beta h$. $\square$

Remark 2. The existence of a β-witness is not enough to bound the CHD. Nevertheless, as we discuss in Section 2.3, 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 [4].

Remark 3. We state the above result in terms of α, Δ and h, in order to make it more transparent. The three quantities are however somewhat related – in particular, it can be shown that $\alpha \leq 2h\Delta$ (cf. [18]).

We finish by observing that the β-witness property can be relaxed further if we assume something about neighborhoods' growth rate. We used the crude bound $|B_r(u)| \leq \Delta^r$, but some networks exhibit a much different behaviour. For example, a grid satisfies $|B_r(u)| = O(r^2)$, so the bound we used could have an exponential gap. This implies that we need a witness only for very long paths.

Corollary 2. *Let f be a function such that, for every $u \in V, r \geq 0$, $|B_r(u)| \leq f(r)$. Then, assuming only that $\mathcal{P}^E_r$ is β-witnessed for $r \geq f^{-1}(\alpha^\beta h)$,we can get the same guarantee as in Theorem 3.*

4. SCALABLE CSP ALGORITHMS: PRACTICAL IMPLEMENTATION

4.1 Practical HL construction

We start by defining a similar structure as the augmented graph, but more amenable to practical purposes. The graph G^B defined in Section 3.1 may contain a lot of redundant information, hence it is not a minimal representation. 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 can reduce the size of the augmented graph and obtain considerable improvements both in query time and in data storage.

We construct an augmented graph without this duplicated information as follows. The nodes are, as before, pairs $\langle v,b\rangle$, but an edge $\langle v,b\rangle,\langle v',b'\rangle$ is placed only if it is essential for some efficient path, i.e., removing said edge changes the correctness of a query. 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. The tracing is such that the augmented path is forced to end at $\langle t,0\rangle$. In other words, all efficient paths $P \in \mathcal{P}^E_{s,t}$ can be represented by a set a set of efficient paths ending at $\langle t,0\rangle$.

Definition 8. The pruned augmented graph is defined by $\tilde{G}^B = (\tilde{V}^B, \tilde{E}^B)$, where

$$\begin{aligned}\tilde{V}^B &:= \{\langle v,b\rangle : v \in V, b = 0,1,\dots,B\},\\ \tilde{E}^B &:= \{\langle v,b\rangle\langle v',b'\rangle : \exists s,t \in V, P \in \mathcal{P}^E_{s,t}, c(P) \le B,\\ &\quad vv' \in P, b = c(P[v,t]), b' = c(P[v',t])\}.\end{aligned}$$

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 40% smaller than G^B. Observe that, by running Dijkstra in G^B, $\tilde{G}^B$ can be computed in time $O(n^2 B \log(nB))$.

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

4.1.2 HL construction via Contraction Hierarchies

We use techniques described in [3] combined with an approach tailored for augmented graphs. The main idea is to first use contraction hierarchies to get a node ranking and then define the forward hubs of a node as the nodes visited during a contraction-based forward search. The reverse hub is defined analogously. These are valid hubs, since the highest rank node in a path is guaranteed to be in both hubs.

The most important parameter in CH is the rank; results vary greatly from one choice of ranks to another. We obtain a rank in G by running a greedy shortest-path cover, defined as follows. 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 [3].

We work with the pruned augmented graph $\tilde{G}_B$, which takes some time to compute, but yields considerably better hubs. Recall that in $\tilde{G}_B$ there are no sink-nodes nor replicated information, since efficient paths are stored just once. Given a rank for nodes in G, we contract $\tilde{G}_B$ as follows. Say that V is ordered according to the rank, 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.

To obtain better hubs we prune the results obtained by contraction-based searches. If w is in the forward search of v with distance d, it might be that $\mathrm{dist}(v,w) < d$, this occurs because the search goes only to higher rank nodes and the discovered path is missing some node. When $\mathrm{dist}(v,w) < d$, we can safely remove w from the hub of v, since the highest ranked node in a shortest path will have the correct distance. 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 $L^-(v)$.

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.

Remark 4. For the first step, a direct approach requires storing all the shortest paths in memory. In many cases, this can impose a prohibitive memory requirement. There are two ways around this issue. We can compute the cover by processing one shortest path tree at a time, this translates to $\Omega(n^3 \log n)$ running time, but $O(n)$ memory. The second approach we suggest is to compute only $k << n$ shortest path trees and cover 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 3, the clustering approach provides a very good approximation of the hubs with even a small k. Note that we can use any existing technique for Contraction Hierarchies [21, 3].

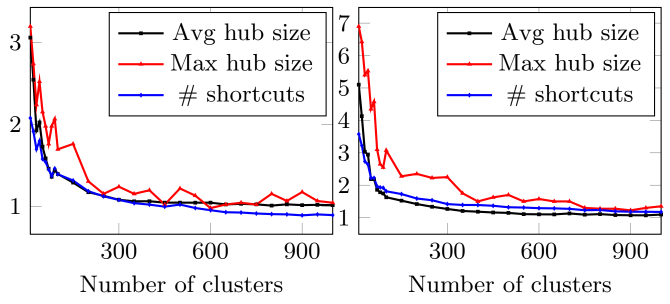

Figure 3: Performance of clustering for San Francisco (left) and Luxembourg (right). In the y-axis the quantities are normalized by the best greedy cover, i.e., using $k = n$.

B	Preproc [m]	Avg F Size	Avg B Size	Query Dij [ms]	Query HL [ms]
0	0	0	0	0	0.000
5-f	5	17	28	74.71	0.02
5-s	5	57	28	32.67	0.009
10-f	13	17	28	183.18	0.03
10-s	10	68	28	56.56	0.01
15-f	14	17	28	250	0.03
15-s	15	73	28	81.48	0.01
20-f	21	17	28	489.74	0.04
20-s	21	77	28	103.72	0.01
25-f	22	17	28	467.84	0.04
25-s	24	80	28	122.57	0.01
30-f	37	17	28	648.17	0.04
30-s	32	84	28	161.99	0.01

(a) San Francisco network.

B	Preproc [m]	Avg F Size	Avg B Size	Query Dij [ms]	Query HL [ms]
0	6	18	18	16.35	0.004
5-f	23	14	19	151.91	0.019
5-s	21	43	19	72.03	0.008
10-f	36	14	19	305.33	0.021
10-s	35	49	19	102.52	0.008
15-f	51	14	19	490.75	0.024
15-s	46	53	19	140.80	0.008
20-f	69	14	19	808.90	0.030
20-s	60	57	19	183.10	0.009
25-f	89	14	19	1016.71	0.034
25-s	77	60	19	227.16	0.009
30-f	101	14	19	1144.07	0.032
30-s	93	62	19	272.40	0.010

(b) Luxembourg City network

Table 1: Experimental results 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 budget levels $b \leq B$ while the results on rows $B - s$ correspond to computing the solution for the budget level b.

4.1.3 *Frontier and specific queries*

We test our algorithms with two different tasks. Recall that the preprocessing is done for some fixed maximum budget B. The first task 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 is called a specific query, for given s, t, b we return $\text{dist}(s, t|b)$, i.e., only one 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 $\text{dist}(s, t|b)$.

4.2 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. The code is available at `github.com/albvera/HHL_CSP`.

We evaluated the performance of our algorithms with real-world test networks: a downtown San Francisco network with 2643 nodes and 6588 edges for which real-world travel-time data was available as a Gaussian mixture model [17], and a Luxembourg City network with 4026 nodes and 9282 edges for which travel-time distributions were synthesized from speed limits [19], 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*

Tables 1a and 1b present the CSP computation times for

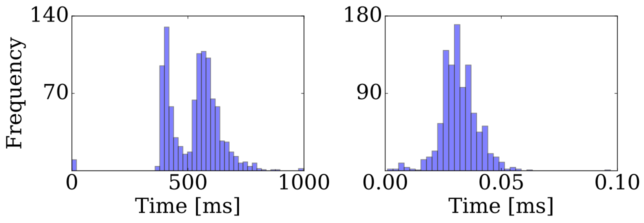

Figure 4: Histogram frontier queries in San Francisco augmented with $B = 25$. Dijkstra (left) and HL (right).

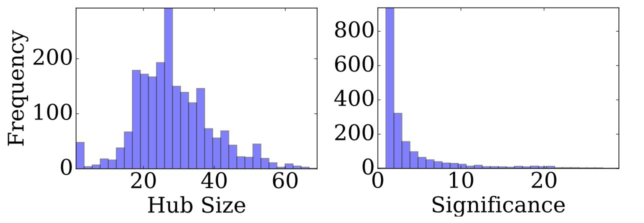

Figure 5: Size of reverse hubs (left) and significance (right) for frontier queries in San Francisco augmented with $B = 25$.

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. On the other hand, for specific queries, $\langle s, b\rangle$ needs to store information for all budgets up to b.

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

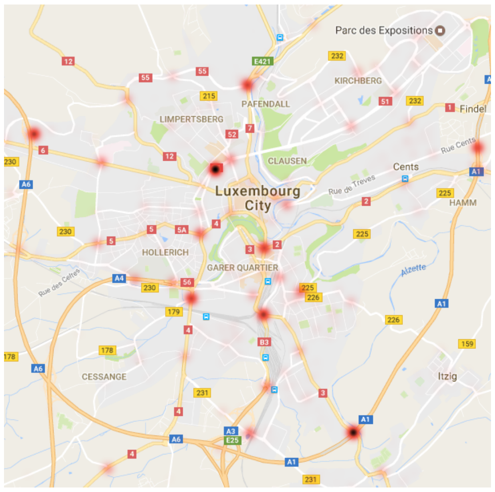

Figure 8: Heat map of significance for frontier queries in Luxembourg City augmented with $B = 25$. Notice how the highly significant nodes are in main road crossings.

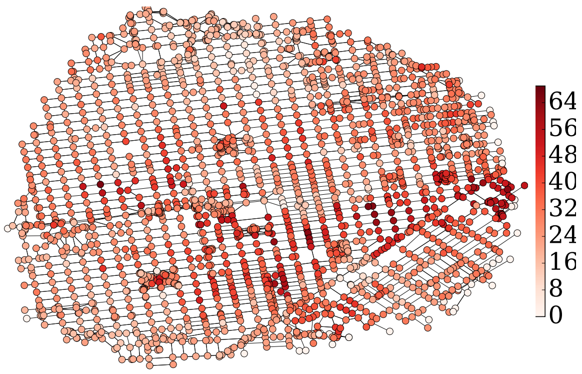

Figure 6: Heat map of hub size for frontier queries in San Francisco augmented with $B = 25$.

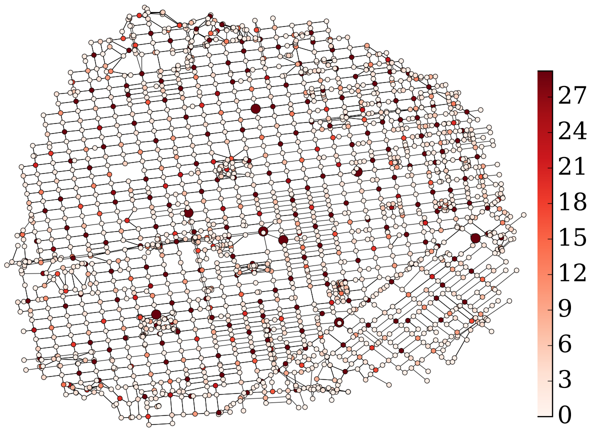

Figure 7: Heat map of significance for frontier queries in San Francisco augmented with $B = 25$. The most significant nodes have been drawn bigger. The top 3 most significant correspond to Geary Blvd & Gough St, Franklin St & O'Farrel St and Market St & Polk St.

average, earlier. In the contraction step, we want to remove a node without altering the shortest path, a process that requires many reachability tests.

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.

Figures 6 and 7 show the spatial relationships between the hub size and significance in the San Francisco 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. ACKNOWLEDGMENTS

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

6. REFERENCES

[1] Project OSRM. `http://project-osrm.org/`. Accessed: 2017-02-01.

[2] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway dimension and provably efficient shortest path algorithms. *Microsoft Research, USA, Tech. Rep*, 9, 2013.

[3] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *International Symposium on Experimental Algorithms*, 2011.

[4] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, 2010.

[5] M. Babenko, A. V. Goldberg, H. Kaplan, R. Savchenko, and M. Weller. On the complexity of hub labeling. In *International Symposium on Mathematical Foundations of Computer Science*, 2015.

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

[7] H. Bast, D. Delling, A. Goldberg, T. Pajor, M. Müller-Hannemann, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. In *Algorithm Engineering*. 2016.

[8] Z. Clawson, X. Ding, B. Englot, T. A. Frewen, W. M. Sisson, and A. Vladimirsky. A bi-criteria path planning algorithm for robotics applications. *arXiv preprint arXiv:1511.01166*, 2015.

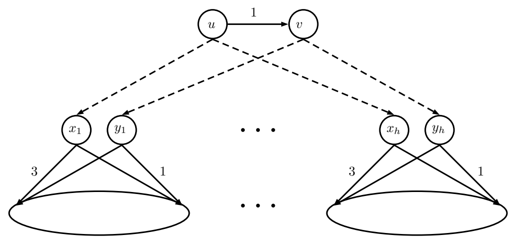

Figure 9: Example where CHD is much larger than the HD.

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

[10] J. R. Correa, T. Harks, V. J. Kreuzen, and J. Matuschke. Fare evasion in transit networks. *arXiv preprint arXiv:1405.2826*, 2014.

[11] C. Demetrescu, A. V. Goldberg, and D. S. Johnson. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, volume 74. 2009.

[12] G. Even, D. Rawitz, and S. M. Shahar. Hitting sets when the vc-dimension is small. *Information Processing Letters*, 95(2), 2005.

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

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

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

[16] D. Hoy and E. Nikolova. Approximately optimal risk-averse routing policies via adaptive discretization. In *AAAI*, 2015.

[17] T. Hunter, P. Abbeel, and A. M. Bayen. The path inference filter: model-based low-latency map matching of probe vehicle data. In *Algorithmic Foundations of Robotics X*. 2013.

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

[19] M. Niknami and S. Samaranayake. Tractable pathfinding for the stochastic on-time arrival problem. In *International Symposium on Experimental Algorithms*, 2016.

[20] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *European Symposium on Algorithms*, 2006.

[21] M. Rice and V. J. Tsotras. Graph indexing of road networks for shortest path queries with label restrictions. *Proceedings of the VLDB Endowment*, 4(2), 2010.

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

[23] S. Samaranayake, S. Blandin, and A. Bayen. Speedup techniques for the stochastic on-time arrival problem. In *OASIcs-OpenAccess Series in Informatics*, volume 25, 2012.

[24] A. Vera, S. Banerjee, and S. Samaranayake. Computing constrained shortest-paths at scale. *Available at https://people.orie.cornell.edu/sbanerjee/CSPTechReport.pdf*, 2017.

[25] C. White. Lower bounds in the preprocessing and query phases of routing algorithms. In *Algorithms-ESA 2015*. 2015.

APPENDIX

A. DIFFERENT NOTIONS OF HD

The definition of HD that we use is tailored for HL note, however, that it is the ones used in [2]. We now discuss how our results extend to the stronger definition.

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

Proposition 10. *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 9. 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 9) 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. □

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 $x > \gamma$. $\square$

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:

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

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 $\ell(P(v, w)) \leq 2^\gamma$. $\square$

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 11. *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.

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, for example, only free 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 12. *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 4. *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.*

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 $b = 0,\dots,B$ 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
```

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
```

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 $P = e_0e_1\dots e_k$, 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 5. *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 $O(Bn\Delta \cdot B\tilde{h}\log D)$. In particular, if $\mathcal{P}^*$ is a β-witness for $\tilde{\mathcal{P}^E}$, then $\tilde{h} \le h2^{\log_2\beta}$.*

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. □

Proof of Proposition 7. 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 [12] 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 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. □