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 consider the problem of computing constrained shortest paths at scale, motivated primarily by the need for accurate and robust routing and travel-time estimates in modern transportation platforms. In recent times, the use of preprocessing techniques and network augmentation has led to dramatic improvements in the speed and scalability of shortest path queries in road networks. These techniques however do not extend to constrained shortest-path (CSP) computations, which are necessary in order to provide robust estimates which incorporate uncertainties in travel times. To this end, we make three main contributions:

- We provide a theoretical characterization of settings where CSP queries can support fast query-times via preprocessing. In particular, we extend the idea of the *highway-dimension*, which encodes the complexity of shortest-path speedup techniques, and show that under an additional condition, we can get similar bounds for CSP problems.
- We develop a practical algorithm for scalable CSP computation, based on combining the hub-labels technique for shortest-path computations with an augmented graph that encodes efficient paths for the CSP problem.
- We perform experimental evaluations of our techniques on datasets with detailed travel-time information for San Francisco and Luxembourg. Our experiments show that our algorithms are orders of magnitude faster than Dijkstra, have small additional storage requirements, and good preprocessing times even on a single machine.

1. INTRODUCTION

Assume each arc has, besides a certain length, an associated cost. The problem of computing the shortest path, with cost at most a given value, is known to be NP-Hard. A dynamic programming approach can compute the shortest path in pseudo-polynomial time, which is the best bound known so far TODO: add reference.

In the face of uncertainty, a natural problem is to compute the shortest path subject to a reliability constraint. We want to estimate how much time it will take to travel from s to t, but, for the answer to be robust, we ask that the path exceeds the estimate with small probability. Formally, given s, t and parameters p, δ, the goal is to find an (s,t)-path P minimizing $\mathbb{E}(\ell(P))$, but satisfying $\mathbb{P}(\ell(P) > \mathbb{E}(\ell(P)) + \delta) \leq p$. This is a variant of the stochastic on time arrival (SOTA) problem TODO: add reference. We propose an approximate solution, which respects the reliability, but may not be optimal. Observe that, using Chebyshov's Inequality, $\mathbb{P}(X > \mathbb{E}(X) + \delta) \leq \mathbb{V}\mathrm{ar}(X)/\delta^2$, hence we can solve the problem

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

The CSP can also capture other natural problems in transit networks. Consider the case where each edge has a probability p_e of triggering a bad event, which results in a fee of F. In this case the agents want to minimize the travel cost and the expected fee. For example, [4] consider a model for fare evasion where p_e is the probability of encountering an inspector. In this case, assuming independence, the natural object is to minimize $\ell(P) + F \cdot \mathbb{P}(\text{trigger bad event})$, i.e.,

$$\min_{P \in \mathcal{P}_{s,t}} \ell(P) + F\Big(1 - \prod_{e \in P}(1 - p_e)\Big).$$

In [4] it is suggested to use a CSP to solve this problem. Indeed, after taking logarithms, the non-linear part of the objective function becomes a linear constraint.

The concept of Highway Dimension (HD) [3, 1] allows to prove the efficiency of shortest path computations in undirected graphs. We will see that directed graphs are fundamental for the analysis of constrained paths. As a first step, we extend the notions of HD to directed graphs.

2. THEORY

2.1 Directed HD

We consider a directed graph $G = (V, E)$ with length function $\ell : E \to \mathbb{N}$. A *path* P is a sequence of nodes $\{u_1 u_2 u_3 \dots u_k\}$ with $(u_i, u_{i+1}) \in E$. The length $\ell(P)$ is the sum of edge lengths in P; by convention, a path with a single node v has length zero. For $s, t \in V$, the distance from s to t, denoted $\mathrm{dist}(s,t)$, is the smallest length among all paths P that start at s and terminate at t. The shortest (s,t)-path, if it exists, is denoted $P(s,t)$. The set of all shortest paths in G is denoted $\mathcal{P}$.

For $r > 0$ and $v \in V$, we define the *forward and reverse balls* by $B_r^+(v) := \{u \in V : \mathrm{dist}(v,u) \leq r\}$ and $B_r^-(v) :=$

$\{u \in V : \mathrm{dist}(u, v) \leq r\}$. We also denote $B_r(v) := B_r^+(v) \cup B_r^-(v)$. A *path system* $\mathcal{Q}$ is any collection of paths, but cycles are not allowed. Given a set $C \subseteq V$ and a path Q, we say that *C hits Q* if some node in Q belongs to C. Similarly, *C hits a path system $\mathcal{Q}$* if it hits every $Q \in \mathcal{Q}$. We describe now some notions that will be central for the definition of HD.

Definition 1. Given $r > 0$, a path Q is r-significant for the system $\mathcal{Q}$ if $Q \in \mathcal{Q}$ and $\ell(Q) > r$. We denote $\mathcal{Q}_r$ as the set of all r-significant paths for the system $\mathcal{Q}$.

Definition 2. Given a system $\mathcal{Q}$ and $r > 0$, an (h, r)-SPHS is a set $C \subseteq V$ with two properties:

1. Hitting: C hits the system $\mathcal{Q}_r$.
2. Local sparsity: for every $v \in V$, $|B_{2r}^-(v) \cap C| \leq h$ and $|B_{2r}^+(v) \cap C| \leq h$.

We define now de notion of a path neighbourhood around a node. The distance from a node v to a path P, denoted $\mathrm{dist}(v, P)$, is measured as the minimum distance from v to a node $w \in P$. The distance to v, $\mathrm{dist}(P, v)$, is defined analogously.

Definition 3. For $v \in V, r > 0$, the forward and reverse path neighbourhoods, respect to a system $\mathcal{Q}$, are defined as

$$S_r^+(v, \mathcal{Q}) := \{Q \in \mathcal{Q} : \ell(Q) > r, \mathrm{dist}(v, Q) \leq 2r\},$$
$$S_r^-(v, \mathcal{Q}) := \{Q \in \mathcal{Q} : \ell(Q) > r, \mathrm{dist}(Q, v) \leq 2r\}.$$

Definition 4. A system $\mathcal{Q}$ has HD h if, for every $r > 0, v \in V$, there exists $H_{v,r} \subseteq V$ such that $|H_{v,r}| \leq h$ and $H_{v,r}$ hits both neighbourhoods $S_r^+(v, \mathcal{Q})$ and $S_r^-(v, \mathcal{Q})$.

The analogous HD of a network, defined in [1] for the undirected case, is obtained when the system of interest is $\mathcal{P}$, the collection of all shortest paths. When we refer to the HD of a network (G, ℓ), it is understood that the system of interest is $\mathcal{P}$. The idea behind this generalized definition is to capture more structure in a road network.

We briefly explain the relation between HD and SPHS. The existence of (h, r)-SPHS enables the construction of efficient hub labels, but the existence alone allows to compute just a $\log n$ approximation of these sets in polynomial time. The HD permits to construct a $\log h$ approximation of the SPHS in polynomial time. Additionally, HD is an upper bound to the doubling dimension.

The HD of a graph does not bound the maximum degree Δ. On the other hand, the HD of the geometric realization does bound Δ. Alberto: This is because we switched to the weaker definition, now paths of single nodes are not significant.

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

2.1.1 Multi-Scale Hitting Sets

The constructions in the following sections will depend on the existence of SPHS. Let $D = \max_{P \in \mathcal{P}} \ell(P)$ be the diameter. We assume that the network (G, ℓ) admits sets C_i, $i = 1, \ldots, \log D$, such that C_i is an $(h, 2^{i-1})$-SPHS.

Computing the desired hitting sets in polynomial time is an interesting problem in its own right. A greedy algorithm achieves a $O(\log n)$ approximation, meaning that the sparsity will be $O(h \log n)$ instead of h . We will also show how to obtain a $O(\Delta \log(h\Delta))$ approximation using a more sophisticated toolTODO: insert reference. The results will still carry out, but one needs to replace h for this approximation when talking about polynomial time preprocessing.

A more subtle point is that the algorithm for the better approximation, even though runs in polynomial time, is impractical. On the other hand, the greedy algorithm also poses some restrictions. It involves an all shortest path computation, so it can be done either with (i) $\Omega(n^2)$ memory and $\Omega(n^3)$ time or (ii) $O(n)$ memory and $\Omega(n^3 \log n)$ time TODO: double check these times. Alberto: More on this?

Given the previous restrictions, we use instead a heuristic for the approximation. We compute only $k << n$ shortest path trees, leading to $O(kn)$ memory and $O(kn^2)$. It is an open problem to determine the approximation guarantee of this algorithm. For more details see TODO: insert reference.

2.2 Hub Labels Using HD

Arguably the two most successful algorithms for the SP problem are CH and HL. When a network has small HD, it is possible to obtain efficient structures for both of these algorithms. We explain here the construction for HL. For the construction of CH see Section A.

The HL algorithm is described as follows. Every node v is assigned two sets of nodes, $L^+(v)$ and $L^-(v)$, referred to as forward and backward hubs. We store $\mathrm{dist}(v, w), \mathrm{dist}(u, v)$ for every $w \in L^+(v), u \in L^-(v)$. The hubs 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 cover property it is trivial to obtain $\mathrm{dist}(s, t)$; we just compare all nodes $w \in L^+(s) \cap L^-(t)$ and return the minimum of $\mathrm{dist}(s, w) + \mathrm{dist}(w, t)$. If the hubs are ordered by ID, we can obtain the distance with a single coordinated sweep in time $O(|L^+(s)| + |L^-(t)|)$. Note that the size of the HL bounds both the query time and the storage requirements. The following result shows how to obtain small HL, the construction of hubs mimics the undirected case.

Theorem 1. *If (G, ℓ) has HD h, then we can construct hub labels of size at most $h \log D$.*

Proof. The hubs are defined 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 C_i is an $(h, 2^{i-1})$-SPHS and we intersect with balls of radii $2 \cdot 2^{i-1}$, 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.

In the case that $P(s, t)$ exists, take i such that $2^{i-1} < \ell(P(s, t)) \leq 2^i$. The path $P(s, t)$ is hit by C_i and any point in the path belongs to both $B_{2^i}^+(s)$ and $B_{2^i}^-(t)$. Such point hitting $P(s, t)$ is then in both hubs, which shows the result. $\square$

2.2.1 *Distance Oracles and Path Computation*

The previous construction is for distance oracles, i.e., they return the length of the SP, but not the path itself. It is possible to recover the path with additional data; if we double the storage and compute not only the distance to $w \in L^+(v)$, but also the second node in $P(v, w)$, then it is easy to recover the path. Each time we run a HL query, at least one node $w \in P(s, t)$, $w \neq s, t$, is returned and we sequentially run queries for the subpaths $P(s, w)$ and $P(w, t)$. Note that we do need to store this extra information, for example, it could be that $L^+(s) \cap L^-(t) = \{s\}$, so the process cannot continue without an additional node.

2.3 Constrained HD

We have established that small HD explains the remarkable performance of HL. In the same spirit, we introduce a notion that allows for the construction of provably efficient HL for the CSP problem. The system of interest will be that of efficient paths, in contrast to shortest paths.

Let $c : E \to \mathbb{N} \cup \{0\}$ be the cost function. Analogously as the length of a path P, the cost $c(P)$ is the sum of edge costs. For any source-terminal pair $s, t \in V$, denote by $\mathcal{P}_{s,t}$ the set of all simple paths from s to t. 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. The set of all efficient paths is denoted $\mathcal{P}^E$. In particular, all paths in $\mathcal{P}_{s,t} \cap \mathcal{P}^E$ form the Pareto frontier from s to t. Observe that every subpath of an efficient path is also efficient. Indeed, if that is not the case, we could improve the path by replacing the subpath.

The definition of Constrained Highway Dimension (CHD) is the natural object when we take the system of efficient paths. Note that every shortest path is efficient, thus the CHD will always be bigger than the HD of a network.

Definition 5. The CHD of (G, ℓ, c), denoted h_c, is the HD of the system $\mathcal{P}^E$.

The first question we want to tackle is: how worse is the sparsity in an EPHS compared to a SPHS? We show that, in general, the sparsity of an EPHS is arbitrarily worse. Intuitively, this occurs when shortest and efficient paths are unrelated. Alberto: The current example uses the stronger notion of r-significant. I can create a very simple example with the weaker definition, but it looks like cheating.

PROPOSITION 2. *For any h, we can construct a family of networks such that the sparsity of SPHS 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 1. 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 1) 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 SPHS is $h+1$.

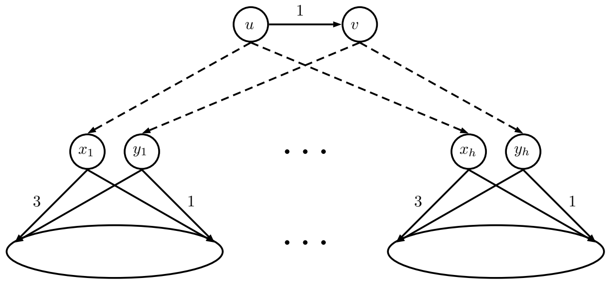
On the other hand, every leaf w is a 2-efficient path. Indeed, it can be extended to $x_i w$ that is the shortest path from x_i to w with constraint 1. All the leafs are in the ball $B_4(u)$, so the sparsity is at least h^2.

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

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

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

Figure 1: Example where the CHD is much larger than the HD



It is conjectured that road networks have small HD. Is there evidence to conjecture small CHD in these networks? We show how doubling dimension plus an additional property, defined bellow, do imply a positive answer. A network is said to have doubling dimension α if, for any $r > 0$, every ball of radius $2r$ can be covered by at most α balls of radius r Alberto: A tad vague because of forward and backward balls, I didn't find references with doubling dimension and directed graphs. A path P' *partially witnesses* P if $P' \subseteq P$ and P' is "long enough". Formally, we define the following relation for path systems.

Definition 6. Let $\beta \geq 1$. We say that a path system $\mathcal{Q}'$ is a β-witness of the path system $\mathcal{Q}$ if, for every $Q \in \mathcal{Q}$, exists $Q' \in \mathcal{Q}'$ such that $Q' \subseteq Q$ and $\ell(Q') \geq \frac{1}{\beta}\ell(Q)$.

We explore first when the system of shortest paths $\mathcal{P}$ is a partial witness for the system of efficient paths $\mathcal{P}^E$. At an intuitive level, the partial witness property says that efficient and shortest paths are not completely different, i.e., if Q is efficient, some fraction of Q is shortest. As a consequence, an important node hitting numerous paths in $\mathcal{P}$, should also hit many paths in $\mathcal{P}^E$. The bad examples in Proposition 2 exploit networks that do not have partial witnesses. We stress that doubling dimension depends only on G and ℓ; the partial witness property depends on the interplay between G, c and ℓ.

PROPOSITION 3. *Assume G is α-doubling and $\mathcal{P}$ is a β-witness for $\mathcal{P}^E$. If (G, ℓ) admits an (h, r)-SPHS for any r, then (G, ℓ, c) admits an $(\alpha^{\lceil \log_2 \beta \rceil} h, r)$-EPHS for any r.*

PROOF. Let $r > 0$, we need to construct a hitting set C^E for $\mathcal{P}^E_r$. Define $k := \lceil \log_2 \beta \rceil$ and observe that $\frac{1}{\beta} \geq 2^{-k}$. Take C as the hitting set for $\mathcal{P}_{2^{-k}r}$, which is guaranteed to be sparse with respect to balls of radius $2^{-k+1}r$. 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^{-k}-witness for $\mathcal{P}^E$, 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^k balls of radius $2^{-k+1}r$. Each of these balls contains at most h elements of C, therefore the sparsity is as claimed. The argument for backward balls is identical. $\square$

Remark 1. The existence of a β-witness is not enough to bound the CHD. Nevertheless, as discussed in Section 2.1.1, the existence of (h, r)-SPHS already allows the construction of HL.

2.4 Augmented Graph

In order to describe the main structure used to construct HL, we start with a discussion on efficient paths. Given a source-terminal pair s, t and a budget b, we want to solve the problem

$$\begin{aligned} \min \quad & \ell(P) \\ \text{s.t.} \quad & c(P) \le b \\ & P \in \mathcal{P}_{s,t}. \end{aligned}$$

Call $\operatorname{dist}(s, t|b)$ the minimum of this problem, potentially $+\infty$ if there is no feasible solution. Note that a solution to $\operatorname{dist}(s, t|b)$ is not necessarily efficient, i.e., there could be another path with the same length and lower cost. Additionally, the budget constraint may not be tight and there could be multiple minimizers. As a consequence, a direct approach with EPHS would not work; there are potentially many more solutions than efficient paths.

For a fixed parameter $B \in \mathbb{N}$, we construct a structure that allows to obtain $\operatorname{dist}(s, d|b)$ for any $b \le B$. Additionally, we will always answer with an efficient path, so there could be a "surplus of budget". On an intuitive level, we create a graph with the following properties:

1. All paths starting from a given node are feasible, i.e., they satisfy the budget constraint.
2. Efficient paths become shortest paths.
3. The use of "unnecessary resource" is penalized, hence inefficient paths are not shortest.

We give now the formal description of the augmented graph. Consider nodes as pairs of the form $\langle v, b\rangle$. This encodes the information of remaining resource $b \ge 0$ and location $v \in V$. A node is connected to neighbours (according to E) as long as the remaining resource of that transition is non-negative. Finally, we create n sink nodes, denoted $\langle v, -1\rangle$.

Definition 7. Given a network (G, ℓ) and $B \in \mathbb{N}$, the augmented version (G^B, ℓ), where $G^B = (V^B, E^B)$, is defined by

$$\begin{aligned} V^B &:= \{\langle v, b\rangle : v \in V, b = -1, 0, 1, \dots, B\}, \\ E^B &:= \{\langle v, b\rangle\langle v', b'\rangle : vv' \in E, b' = b - c_{vv'}, b' \ge 0\} \cup E^B_-, \\ E^B_- &:= \{\langle v, b\rangle\langle v, -1\rangle : v \in V, B \ge b \ge 0\}. \end{aligned}$$

The lengths to sink nodes are $\ell(\langle v, b\rangle, \langle v, -1\rangle) = \frac{1}{b+1}$. The other lengths are preserved, i.e., $\ell(\langle v, b\rangle, \langle v', b'\rangle) = \ell(vv')$.

Remark 2. The edge costs are allowed to be zero, therefore the augmented graph may contain cycles. On the other hand, the dynamic programming approach requires a DAG TODO: insert reference; the topological order is what ensures the efficiency of the algorithm.

Definition 8. Let $P = \langle v_1, b_1\rangle\langle v_2, b_2\rangle \dots \langle v_k, b_k\rangle$ be a path in G^B. The projection of P, denoted $\bar{P}$, is the path in G induced by P. Formally, in the case $b_k \ge 0$, $\bar{P} := v_1 v_2 \dots v_k$ and $\bar{P} := v_1 v_2 \dots v_{k-1}$ in the case $b_k = -1$.

PROPOSITION 4. *A shortest path from $\langle s, b\rangle$ to a sink node $\langle t, -1\rangle$ projects to an efficient path in G solving $\operatorname{dist}(s, t|b)$.*

PROOF. Let P be the shortest path from $\langle s, b\rangle$ to $\langle t, -1\rangle$. To reach $\langle t, -1\rangle$, P must pass trough some $\langle t, b'\rangle$, $b' \ge 0$. By construction, P consumes $b - b'$ units of resource, hence it is feasible. Note that $\bar{P}$ is the shortest among (s, t)-paths with cost $b - b'$.

Assume, by way of contradiction, that $\bar{P}$ is not efficient. As $\bar{P}$ is the shortest using $b - b'$ units of resource, there exists P' such that $\ell(\bar{P}') \le \ell(\bar{P})$ and $c(\bar{P}') < c(\bar{P})$. It must be that P' passes through $\langle t, b''\rangle$, with $b'' > b'$. We argue that, in this case, P would not be a shortest path to $\langle t, -1\rangle$. Indeed,

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

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

Using the previous result, we can construct SPHS in the augmented graph using the EPHS of the original graph. It is interesting that, assuming doubling dimension α and β-witness, we obtain sparsity of $\alpha^{\lceil \log_2 \beta \rceil} hB$ and thus query times of $O(\alpha^{\lceil \log_2 \beta \rceil} hB \log D)$. This is much better than the $O(Bn\Delta)$ of dynamic programming, but it is still pseudo-polynomial. We observe that the query time can be made strongly polynomial if and only if the Pareto frontier contains at most a poly-logarithmic number of points. See Section 2.6 for a further discussion on this.

In G^B we are not interested in every shortest path, but only on those ending in sink nodes, since these are projected to efficient paths. Let $\mathcal{P}^B$ be such system, i.e., 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 we require more information by augmenting the graph, the sparsity increases.

PROPOSITION 5. *Assume the system $\mathcal{P}^E$ admits (h_c, r)-EPHS. Then, $\mathcal{P}^B$ admits $(h_c B, r)$-SPHS.*

PROOF. We define a set C^B and prove that hits $\mathcal{P}^B_r$ and that it is locally sparse. Call C the (h_c, r)-EPHS for $\mathcal{P}^E$. Define

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

By Proposition 4, we know that shortest paths are efficient, hence C^B hits all the desired paths. Finally, we prove local sparsity. Take any node $\langle u, b\rangle$ and observe that

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

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

Remark 3. The proof shows a stronger result. In Equation (1) we see that the sparsity around the node $\langle u, b\rangle$ is $h_c b$. This will be key for the query time guarantee.

The last result shows how SPHS translate to the augmented graph. Since the notion of HD is much stronger, it is not always possible to make the analogous between two path systems. Surprisingly, in this case we are able to relate the HD of both path systems.

Proposition 6. *If the HD of the system $\mathcal{P}^E$ is h_c, then the HD of the system $\mathcal{P}^B$ is Bh_c.*

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

Take $P \in S^+_r(\langle v, b\rangle, \mathcal{P}^B)$. Since $\text{dist}(\langle v, b\rangle, P) \leq 2r$, 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 $S^-_r(\langle v, b\rangle, \mathcal{P}^B)$. □

2.5 Hub Labels For CSP

We use a variant of HL to find the efficient paths. There is a subtle difference, we are only interested in paths ending in a sink node. We explain how to create the HL and use them to answer the value of $\text{dist}(s, t|b)$. Afterwards we show how to use the previous to reconstruct the desired path in linear time.

Nodes $\langle v, b\rangle$ have associated a forward hub $L^+(\langle v, b\rangle) \subseteq V^B$. A sink node has a backward hub $L^-(\langle u, -1\rangle) \subseteq V^B$. The hubs must satisfy the cover property: for every pair $\langle s, b\rangle$, $b \geq 0$, and $\langle t, -1\rangle$, there exists a node in $\langle u, x\rangle \in L(\langle s, b\rangle)^+ \cap L(\langle t, -1\rangle)^-$ such that $\langle u, x\rangle$ is in the shortest path from $\langle s, b\rangle$ to $\langle t, -1\rangle$. As before, if the intersection contains many nodes, we take the one minimizing the quantity $\text{dist}(\langle s, b\rangle, \langle u, x\rangle) + \text{dist}(\langle u, x\rangle, \langle t, -1\rangle)$. Additionally, if there is no such $\langle u, x\rangle$, then there exists no path, i.e., $\text{dist}(s, t|b) = \infty$.

The algorithm to construct the path is very simple if, as discussed in Section 2.2.1, we also store the first hop to every node in the hub. Assume we can answer queries of the form $SP(\langle s, b\rangle, \langle t, -1\rangle)$ and output either "not possible" or a node $\langle v, x\rangle$ in the SP and the budget consumed. Consider the case when all the budget was used. We can find a node in the SP from $\langle s, b\rangle$ to $\langle v, x\rangle$ by running a query $SP(\langle s, b-x\rangle, \langle v, -1\rangle)$. This process can be repeated until the path is recovered. In the other hand, if there was a surplus of budget, we adjust b to the smaller quantity; the adjustment is necessary only in the very first query, the subsequent queries will always consume all the budget.

2.5.1 Query Time and Data Requirements

We now prove that the desired HL exist and show how much extra information is needed. It is interesting to note that the augmented graph is only a theoretical construction. From a computational point of view, we can obtain the HL using only the EPHS of the original graph. Nevertheless, the augmented graph allows us to prove that all the pieces fit together.

Theorem 2. *Assume the system $\mathcal{P}^E$ admits an (h_c, r)-EPHS for every r. Then, we can construct HL such that, for every $u \in V$ and $b \geq 0$, $|L(\langle u, b\rangle)^+| \leq bh_c \log D$ and $|L(\langle u, -1\rangle)^-| \leq Bh_c \log D$.*

Proof. For $i = 0, 1, \dots, \log D$, let C_i be an $(h_c, 2^{i-1})$-EPHS. Now augment C_i to C^B_i as in Equation (1). Define the HL by

$$L(\langle v, b\rangle)^+ := \bigcup_{i=1}^{\log D} C^B_i \cap B^+_{2^i}(\langle v, b\rangle)$$
$$L(\langle u, -1\rangle)^- := \bigcup_{i=1}^{\log D} C^B_i \cap B^-_{2^i}(\langle u, -1\rangle).$$

The cover property is proved as in Theorem 1, we just bound the size here. The number of elements in each forward hub is as claimed because $|C^B_i \cap B^+_{2^i}(\langle v, b\rangle)| \leq bh_c$. For a backward hub we use that

$$B^-_{2^i}(\langle u, -1\rangle) = \{\langle v, x\rangle : \exists P \in \mathcal{P}_{v,u}, c(P) = x, \ell(P) \leq 2^i\}$$
$$\subseteq B^-_{2^i}(u) \times \{0, 1, \dots, B\}.$$

Thus, $B^-_{2^i}(\langle u, -1\rangle) \cap C^B_i \leq Bh_c$. □

Proposition 7. *Using the HL given by Theorem 2, we can implement queries for s, t, b in time $O(bh_c \log D)$. The total space requirement is $O(nB \cdot Bh_c \log D)$.*

Proof. The backward hub for a sink $\langle u, -1\rangle$ has at most $Bh_c \log D$ nodes. For each such node $\langle v, b\rangle \in L^-(\langle u, -1\rangle)$ we store: the distance to the sink $\text{dist}(v, u|b)$ and the efficient cost, i.e., we know how much surplus there is. In the case of forward hub for a node $\langle u, b\rangle$ we store only distances.

To perform a query for s, t, b, we look only for nodes $\langle u, x\rangle$ with $x \leq b$ in the backward hub of $\langle t, -1\rangle$ Alberto: this requires a data structure to know the level of each node, but I guess it's clear from the context. As we compare two ordered lists of size at most $h_c b \log D$, the result follows. □

2.5.2 Preprocessing

It was shown in [1] that the set system of shortest paths has VC-dimension of 2. More formally, paths $v_1 v_2 \dots v_k$ are mapped to sets $\{v_1, \dots, v_k\}$ and this new set system has small VC-dimension. Observe that, if G is undirected, then each subset represents exactly two paths, namely the (s, t)-path and the (t, s)-path. Therefore, a hitting set for the set system corresponds to a hitting set for $\mathcal{P}$.

Recall that we assume uniqueness of shortest paths. It is easy to see that the VC-dimension of the previous set system is 2. Indeed, a path abc is mapped to $\{a, b, c\}$ and the element $\{a, c\}$ would never form a part of the system; the shortest path between a and c must pass through b, therefore $\{a, b, c\}$ cannot be shattered.

In directed graphs the same relation can still be made, but the VC-dimension would not be small. For example, the directed graph can contain the shortest paths abc and ca, mapping to sets $\{a, b, c\}$ and $\{c, a\}$, breaking the previous argument.

To overcome this, we now think of paths as sequences of edges. The ground set for the set system is E, instead of V and a path $e_1 e_2 \dots e_k$ is mapped to $\{e_1, e_2, \dots, e_k\}$. Note that each $\{e_1, e_2, \dots, e_k\}$ corresponds uniquely to one path. Recall that in a system $\mathcal{Q}$ the cycles are not allowed. We denote $\pi(Q)$ the set of edges in a path Q.

Proposition 8. *Given a system $\mathcal{Q}$, the corresponding set system $(E, \{\pi(Q) : Q \in \mathcal{Q}\})$ has VC-dimension 2.*

We now show how to obtain polynomial time preprocessing. Specifically, the algorithms we use are based on the existence of $(h, 2^i)$-SPHS. Assuming a system has HD h, we can obtain an approximation of these set, where the sparsity increases by the maximum degree Δ.

Proposition 9. *If a system $\mathcal{Q}$ has HD h, for some $h' = O(h\Delta \log(h\Delta))$ there exists a polynomial time algorithm to obtain a (h', r)-SPHS.*

Proof. Denote $S_r(v) := S_r^+(v, \mathcal{Q}) \cup S_r^-(v, \mathcal{Q})$, i.e., $S_r(v)$ contains all the r-significant paths "close" to v. 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 $H_{v,r} \subseteq V$, $|H_{v,r}| \leq h$, 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 TODO: add reference 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 $|F_{v,r}| \leq 2h'$.

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 $(2h', r)$-SPHS 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. □

2.6 Using the size of the Pareto Frontier

If all the other edges are used by some efficient path, then we have no more improvements. To avoid this, we assume the following condition on the structure of the efficient paths.

Definition 9. Given a function $g : \mathbb{N} \to \mathbb{N}$, we say that (G, ℓ, c) satisfies the g property if, for every $s, t \in V$, $b \in \mathbb{N}$,

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

Theorem 3. *Under assumptions, we can create hub labels of size $O(g(B)h_c \log D)$ and answer queries with budget b in time $O(g(b)h_c \log D)$.*

Proof. We make the hubs slightly bigger by storing, for each node in $L^+(\langle v, b\rangle)$, the distance from $\langle v, b\rangle$ and surplus of budget. Let C_i be the $(h_c, 2^{i-1})$-EPHS and $\mathcal{P}_{s,t}^E$ the efficient paths from s to t. The hub is obtained by running Algorithm 1. Note that the marked nodes will be used as input for another algorithm.

Algorithm 1 Construction of forward hub

```
Input: Node $s \in V$, set of efficient paths $\mathcal{P}_{s,t}^E$ for every $t$ and sets $C_i$.
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 $\mathcal{P}_{s,t}^E$ by increasing cost and remove paths consuming more than $B$.
 2: for $t \in V \setminus s$ do
 3:     for $P \in \mathcal{P}_{s,t}^E$ 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 the node $v_P$.
 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
```

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 backward hub is easier; we need to store only the distance from a node to the destination. The construction is given in Algortihm 2

Algorithm 2 Construction of backward hub

```
Input: Node $t \in V$, set of efficient paths $\mathcal{P}_{s,t}^E$ for every $s$, marked nodes and sets $C_i$.
Output: Backward hub $L^-(\langle t, 0\rangle)$.
 1: Order each $\mathcal{P}_{s,t}^E$ 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 $P \in \mathcal{P}_{s,t}^E$ 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
```

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 s, t, b 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 $Lb(\langle d, 0\rangle)$. The result follows. □

2.6.1 Extensions

We start by comparing the HD of the union of two systems. 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 10. *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.

Definition 10. We say that $\mathcal{P}^E$ does not alternate between $\mathcal{Q}$ and $\mathcal{Q}'$ if, 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 $Q' \in \mathcal{Q}'$.

THEOREM 4. *Assume that G has doubling dimension α and $\mathcal{Q}, \mathcal{Q}'$ are systems with HD h, h' respectively. If $\mathcal{P}^E$ does not alternate, then it admits $(\alpha^{\log \beta} h + \alpha^{\log \beta'} h', r)$-EPHS.*

2.6.2 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 $P = e_0 e_1 \ldots e_k$, 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 $O(Bn\Delta \cdot B\tilde{h} \log D)$ TODO: double check this. In particular, if $\mathcal{P}$ is a β-witness for $\tilde{\mathcal{P}^E}$, then $\tilde{h} \leq h 2^{\log_2 \beta}$.*

3. EXPERIMENTS

The augmented graph may contain a lot of unnecessary 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. If there are not too many efficient paths, we can obtain considerable improvements both in query time and in data storage.

The idea is to construct an augmented graph without duplicated information. 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 some efficient path uses it. In other words, 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 $b \leq B$ and trace it in the augmented graph starting at $\langle s, b \rangle$ and ending at $\langle t, 0 \rangle$.

Definition 11. The pruned augmented graph is defined by $\tilde{G}^B = (\tilde{V}^B, \tilde{E}^B)$, where

$$\tilde{V}^B := \{\langle v, b \rangle : v \in V, b = 0, 1, \ldots, B\},$$
$$\tilde{E}^B := \{\langle v, b \rangle \langle v', b' \rangle : \exists s, t \in V, P \in \mathcal{P}^E_{s,t}, c(P) \leq B,$$
$$vv' \in P, b = c(P[v, t]), b' = c(P[v', t])\}.$$

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.

3.1 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 $\langle t, 0 \rangle$. On the other hand, if P projects to an efficient path, then necessarily P is shortest. To bound the HD, the correct system to study is

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

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 6.

PROPOSITION 11. *The HD of $\tilde{\mathcal{P}}^B$ is Bh_c, where h_c is the HD of $\mathcal{P}^E$.*

We use techniques described by [2] together with an approach tailored for augmented graphs. The main idea is to use contraction hierarchies first, then define the forward hub of a node as the nodes visited during a contraction-based forward search. The backward 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.

We work with the pruned augmented graph, i.e. $\tilde{G}_B$, 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 $\tilde{G}_B$, for

$b = B, \ldots, 0$, we contract the nodes $(1, b)$ first, then the nodes $(2, b)$ and so on till the nodes (n, b) 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 $\text{dist}(v, w) < d$, 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. We describe now the process in the following steps.

1. Compute the shortest paths in G and obtain a cover C
2. Compute the pruned augmented graph $\tilde{G}_B$
3. Contract $\tilde{G}_B$ using the rank given 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.

4. CONCLUSIONS

5. ACKNOWLEDGMENTS

6. REFERENCES

[1] 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.

[2] 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*, pages 230–241. Springer, 2011.

[3] 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*, pages 782–793. Society for Industrial and Applied Mathematics, 2010.

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

APPENDIX

A. CONTRACTION HIERARCHIES

We present here how to extend the concept of HD in order to prove the efficiency of CH in directed graphs.

Definition 12. For $r > 0$, the shortest path $P \in \mathcal{P}$ is r-significant if we can add at most one vertex at each end of P to obtain a shortest path $P' \in \mathcal{P}$ such that $\ell(P') > r$. The path P' is called an r-witness of P. We denote $\mathcal{P}_r$ as the set of all r-significant paths.

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.

Definition 13. For $r > 0$, an (h, r)-SPHS is a set $C \subseteq V$ satisfying:

1. The set $\mathcal{P}_r$ is hit by C: for every $P \in \mathcal{P}_r$, $P \cap C \neq \varnothing$.
2. C is h-sparse: for every $v \in V$, $|B^-_{2r}(v) \cap C| \leq h$ and $|B^+_{2r}(v) \cap C| \leq h$.

The following is not the usual notion of a sphere. For $v \in V$, we are interested in paths, not points, close to v. We do not say that P is close to v if $\text{dist}(v, P)$ is small, but rather if P has some r-witness P' with $\text{dist}(v, P')$ small.

Definition 14. For $v \in V, r > 0$, the path neighbourhood $S^+_r(v)$ is defined as the set of paths $P \in \mathcal{P}_r$ such that P has some r-witness P' with $\text{dist}(v, P') \leq 2r$. The neighbourhood $S^-_r(v)$ is defined analogously.

Definition 15. The network (G, ℓ) has HD h if, for every $r > 0, v \in V$, there exists $H_{v,r} \subseteq V$ such that $|H_{v,r}| \leq h$ and $H_{v,r}$ hits every path in $S^+_r(v)$ and every path in $S^-_r(v)$.

A.1 Additional Data Requirements

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:

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

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 $\ell(P(v, w)) \leq 2^\gamma$. □

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 $2nh \log D$.

A.2 Query Time

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 backward search, the condition is the analogous $\text{dist}(v, t)' + \ell(w, v) \leq 2^i$ when $(w, v) \in E \cup E^+$.

Proposition 12. *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.