An Efficient Algorithm for Placing Electric Vehicle Charging Stations*

Pankaj K. Agarwal[1], Jiangwei Pan[2], and Will Victor[3]

1 **Duke University, Durham, USA**
pankaj@cs.duke.edu
2 **Duke University, Durham, USA**
jwpan@cs.duke.edu
3 **Duke University, Durham, USA**
william.victor@duke.edu

Abstract

Motivated by the increasing popularity of electric vehicles (EV) and a lack of charging stations in the road network, we study the shortest path hitting set (SPHS) problem. Roughly speaking, given an input graph G, the goal is to compute a small-size subset H of vertices of G such that by placing charging stations at vertices in H, every shortest path in G becomes *EV-feasible*, i.e., an EV can travel between any two vertices of G through the shortest path with a full charge. In this paper, we propose a bi-criteria approximation algorithm with running time *near-linear* in the size of G that has a logarithmic approximation on $|H|$ and may require the EV to slightly deviate from the shortest path. We also present a data structure for computing an EV-feasible path between two query vertices of G.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Computations on Discrete Structures

Keywords and phrases Shortest path hitting set, Charging station placement, Electric vehicle

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.7

1 Introduction

Motivation. Electric vehicles (EVs) are becoming increasingly popular as we transition from fossil fuels to cleaner energy. One of the main challenges in the popularization of EVs is the lack of charging facilities in the road network. Ideally, one should be able to reach a charging station quickly anywhere on the road network, as in the case of gas stations. However, due to resource constraints and the relatively small fraction of EVs currently on the road, it is desirable to first build a small number of charging stations to satisfy the most basic transportation needs of EV owners. One natural such need is that an EV, with an initial full charge, should be able to travel between any two locations via the shortest path without draining the battery. In other words, any shortest path in the road network contains sufficient number of charging stations. We study the problem of placing the minimum number of charging stations to satisfy the above condition.

* Work on this paper is supported by NSF under grants CCF-11-61359, IIS-14-08846, CCF-15-13816, and ISS-14-47554 by an ARO grant W911NF-15-1-0408, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

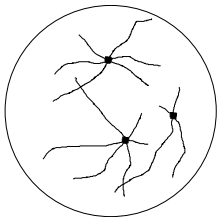

Figure 1 Illustration of the definition of highway dimension: a ball of radius r, and three vertices that intersect all shortest paths within the ball whose lengths are $\Omega(r)$.

Problem statement. The input consists of a graph $G = (V, E)$ of $n = |V|$ vertices and $m = |E|$ edges, which represents a road network, and a positive length function $\ell(\cdot)$ on the edges in E. We assume that an EV can travel a fixed distance r (e.g., 200km) in G with a full charge. More sophisticated models have been proposed for the battery capacity, which not only consider the distance but also the topography of the underlying terrain. But we use this simpler model because the problem is challenging even in this model and furthermore, on realistic terrains the EV will travel distance in the range $[\frac{r}{c}, cr]$, for some small constant $c \geq 1$, with a full charge. For any two vertices $u, v \in V$, let $\pi_G(u, v)$ denote the shortest path from u to v in graph G; it is abbreviated $\pi(u, v)$ when there is no ambiguity. For convenience, we set $\mu(u, v) := \ell(\pi(u, v))$. For a subset $X \subseteq V$ and a vertex $v \in V$, let $\mu(v, X) := \min_{x \in X} \mu(v, x)$.

Given a set X of vertices, a path P is said to be *hit* by X if X contains an *interior* vertex of P — a vertex of P other than its starting and ending vertices. We say a path P is *r-EV-feasible* with respect to X (charging stations) if every contiguous subpath of P of length more than r is hit by X. An *r-shortest-path hitting set* (r-SPHS) of G is a subset $H \subseteq V$ such that for all $u, v \in V$, $\pi(u, v)$ is r-EV-feasible with respect to H. Similarly, given $\delta \in (0, 1)$, a *δ-approximate* r-SPHS of G is a subset $\tilde{H} \subseteq V$ such that for all $u, v \in V$, there exists an r-EV-feasible path P (with respect to $\tilde{H}$) between u, v with $\ell(P) \leq (1 + \delta)\mu(u, v)$. The goal of the *shortest-path hitting-set (SPHS) problem* is to compute a minimum-size r-SPHS of G. In the rest of the paper, for simplicity, we may leave out parameter r and just write SPHS and EV-feasible.

The problem of computing minimum number of charging stations reduces to an instance of the classical hitting-set problem, and is NP-complete by a simple reduction from the vertex-cover problem. Since we are not aware of a proof of the NP-completeness in the literature, we describe the details of the reduction in Section 2. We propose an efficient approximation algorithm for the SPHS problem that exploits the structure of road networks.

Related work. In the last few years, there has been extensive work on a variety of optimization problems on road networks, which are modeled as "sparse" graphs with additional structural properties. In particular, Abraham *et al.* [3, 1, 2] introduced the notion of *highway dimension* to give provable guarantees of efficiency for many popular shortest-path heuristics, such as reach, contraction hierarchies, and transit node; see also [6]. Roughly speaking, the graph G has highway dimension h if, for any $x > 0$ and any vertex $v \in V$, there exist h vertices that intersect all shortest paths of length at least x that are within $O(x)$ distance of v. See Figure 1 for illustration and Section 2 for the definition. Abraham *et al.* argued that real-world road networks have small highway dimension.

Storandt and Funke [21] formulated the problem of placing minimum number of charging stations such that there exists some EV-feasible path between any two vertices. They gave a

polynomial-time algorithm that achieves $O(\log n)$ approximation. However, the EV-feasible path computed by their algorithm can be much longer than the shortest path. This drawback was addressed by Funke *et al.* [13]. They require the *shortest path* between any two vertices to be EV-feasible. They modeled the problem as a hitting-set problem (defined in Section 2), and obtained an $O(\log n)$ approximation using a greedy algorithm. Constructing the hitting set instance requires computing as many as $O(n^2)$ shortest paths, and can take $O(n^3)$ time in the worst case, which is formidable when the road network is large. Funke *et al.* [13] applied a number of techniques to speed up the computation, but without provable guarantees of the running time and approximation.

Several variants of the SPHS problem have been studied. For example, the road network may be small so that one can always drive from one location to another without recharging. In these cases, the charging stations are placed to satisfy other constraints. For example, Xiong *et al.* [26] take EV drivers' behavior into consideration and compute a set of charging stations in Singapore that optimizes the equilibrium utility of a congestion game. There are other optimization criteria considered in the literature, such as charging demand coverage [11] and EV access cost [22].

Another set of literature study the EV routing problem. Baum *et al.* [7] gave an algorithm that plans routes minimizing overall trip time, including time for necessary rechargings on the way. Their model allows the charging time to be a function of the remaining battery level. Goodrich and Pszona [14] formulated a bi-criteria path optimization problem, where two objectives (e.g., travel time and energy cost) are optimized, and their algorithm outputs a path that optimizes one objective before reaching some vertex and switches to the other objective afterwards. See also [8, 19] for work on computing energy-efficient paths.

As discussed above, the SPHS problem is an instance of the hitting-set problem, one of the twenty-one problems in Karp's original list of NP-complete problems [17]. The natural greedy algorithm that chooses the element that hits the most remaining sets gives an $O(\log n)$ approximation [9], which is optimal up to $o(1)$ factor unless P=NP [10]. For geometric instances, however, where the input consists of points and shapes (e.g., disks, rectangles), better approximation guarantees can be obtained. For example, a PTAS exists when the shapes are half-spaces in $\mathbb{R}^3$ [18] and $O(\log \log \mathrm{OPT})$ approximation can be obtained when the shapes are axis-parallel rectangles [5]. Recently, Agarwal and Pan [4] gave near-linear-time approximation algorithms for computing hitting set and set cover of many geometric instances. The hitting-set problem has also been used to compute a subset of vertices that intersect every path [12] or every shortest path [23] that contains at least k vertices.

Our result. We present a bi-criteria approximation algorithm for the SPHS problem by allowing an EV to slightly deviate from the shortest path. Our result is summarized as follows.

▶ **Theorem 1.** *Let $G = (V, E)$, $\ell : V \to \mathbb{R}^+$ be a weighted graph of constant highway dimension h, with $|V| = n$ and $|E| = O(|V|)$. Let $r > 0$ and $\delta \in [\frac{10\alpha}{r}, \frac{2}{15}]$ be two parameters where $\alpha = \max_{e \in E} \ell(e)$, and let κ be the size of a smallest r-SPHS of G. A δ-approximate r-SPHS $\tilde{H} \subseteq V$ of size $O(\kappa \log \kappa)$ can be computed in randomized expected time $O(c_\delta n \log^2 n \log \kappa)$, where $c_\delta = h^{-\log_2 \delta}$.*

In this paper, we assume $10\alpha/r \le \delta \le 2/15$, where the constants 10 and $2/15$ are chosen for convenience of the analysis. Since G represents a road network, the length of a road edge in E is much smaller than the range of an average EV. Hence, $10\alpha/r \ll 1$ and δ can be set to a small constant under the assumption. We also assume $|E| = O(|V|)$ since G represents a road network and the average degree of a vertex is usually small.

At a high level, we improve the running time from $O(n^3)$ to near linear by relaxing the shortest-path requirement slightly. The algorithm works in two stages. The first stage computes a small set C of "center" vertices such that there exists a path between any pair of vertices in G that is not much longer than the shortest path and can be decomposed into shortest paths between center vertices, called *critical paths*. Furthermore, C is a δ-approximate r-SPHS, but the size of C may be much larger than κ. The second stage chooses a smaller δ-approximate r-SPHS. In particular, it computes a small-size hitting set for the critical paths. With the assumption that G has constant highway dimension, we show that the number of center vertices is small, and the optimal hitting set for the critical paths has similar size as the optimal SPHS. The algorithm uses the framework in [4], together with the dynamic trees [20] data structure, to efficiently compute a hitting set for critical paths.

Finally, we describe a data structure for the *feasible path* queries that, given two query vertices u, v, computes in $O(\kappa \log^2 \kappa)$ time the sequence of charging stations on an r-EV-feasible path P between u, v with $\ell(P) \leq (1+\delta)\mu(u,v)$. The actual path in G can be recovered by performing shortest-path queries in G between adjacent charging stations. Since the highway dimension of G is bounded, each shortest-path query can be answered quickly [3].

2 Preliminaries

In this section, we define several concepts that are used by our algorithm, including the highway dimension and doubling dimension of a graph, and the hitting set and ε-net of a range space. We also describe a proof of the NP-completeness of the SPHS problem.

Given $x > 0$ and a vertex $u \in V$, let $\mathcal{B}(u,x) = \{v \in V \mid \mu(u,v) \leq x\}$ be the *ball* of radius x centered at u under the shortest path metric on G.

▶ **Definition 2.** The *highway dimension* of a graph $G = (V, E)$ is the smallest integer h that satisfies the following condition: for all $x > 0$ and $u \in V$, there exists a set $S \subseteq \mathcal{B}(u, 6x)$ of at most h vertices that contains a vertex from every shortest path inside $\mathcal{B}(u, 6x)$ of length more than x.[1]

A metric space has *doubling dimension* d if any ball of radius x is contained in the union of at most 2^d balls of radius $x/2$. We will always use d to denote the doubling dimension of the shortest path metric of G and h to denote the highway dimension of G. Lemma 3 relates these two quantities.

▶ **Lemma 3** ([2]). $d \leq \log_2(h+1)$.

Let $\Sigma = (X, \mathcal{R})$ be a finite range space where X is a finite set of elements and $\mathcal{R}$ is a family of subsets of X called *ranges*. A subset $H \subseteq X$ is called a *hitting set* of Σ if H intersects every range in $\mathcal{R}$. Given a parameter $\varepsilon \in (0, 1]$ and a weight function $w(\cdot)$ on elements of X, an *ε-net* for Σ is a subset $N \subseteq X$ that intersects every *ε-heavy* range, i.e., every range that has weight at least $\varepsilon w(X)$.

The *VC-dimension* [24] of a range space $\Sigma = (X, \mathcal{R})$ is the largest positive integer b satisfying the following condition: there exists a subset $Y \subseteq X$ with $|Y| = b$ such that $|\{S \cap Y \mid S \in \mathcal{R}\}| = 2^b$. The following ε-net theorem was proved in [16] (see also [15]).

[1] We remark that the original paper [3] that introduces highway dimension uses a constant 4 as the multiplier of the radius of the ball, but leaves open the possibility of larger constants (with adjusted constants in other bounds). We use a larger constant 6 for convenience of our analysis.

▶ **Lemma 4** ([16]). *Given a range space $(X, \mathcal{R})$ of VC-dimension β and parameters $\varepsilon, \phi \in (0, 1)$, a set of $O(\frac{\beta}{\varepsilon}(\log \frac{1}{\varepsilon} + \log \frac{1}{\phi})$ independent random samples of X is an ε-net of $(X, \mathcal{R})$ with probability at least $1 - \phi$.*

In this paper, we will be interested in range spaces $\Sigma_G = (V, \mathcal{R})$ where each range in $\mathcal{R}$ corresponds to the vertices on a shortest path in G. Abraham *et al.* [1] showed that the VC-dimension of Σ_G is two when $\mathcal{R}$ contains all shortest paths in G. By the definition of VC-dimension, it is easy to check that the VC-dimension of Σ_G is no more than two when $\mathcal{R}$ contains a subset of all shortest paths in G. It is summarized in the following lemma.

▶ **Lemma 5** ([1]). *The VC-dimension of Σ_G is at most two.*

The decision version of the SPHS problem is as follows: given a graph G, a parameter $r > 0$ and an integer k, determine whether there exists an r-SPHS of G of size at most k.

▶ **Theorem 6.** *The SPHS problem is NP-complete.*

Proof. We reduce the vertex-cover problem to the SPHS problem. Recall that given a graph $G_1 = (V_1, E_1)$, a subset $A \subseteq V_1$ is a vertex cover if $\{u, v\} \cap A \neq \emptyset$ for all $(u, v) \in E_1$. We construct another undirected graph $G_2 = (V_2, E_2)$, where $V_2 = V_1 \cup \{u_e, v_e \mid e = (u, v) \in E_1\}$ and $E_2 = E_1 \cup \{(u, u_e), (v, v_e) \mid e = (u, v) \in E_1\}$, and $\ell(e) = 1$ $\forall e \in E_2$. We claim that a vertex cover in G_1 corresponds to a 2-SPHS of G_2. Suppose $S_1 \subseteq V_1$ is a vertex cover for G_1. Then S_1 must be a 2-SPHS of G_2 because every shortest path of length more than 2 in G_2 must contain at least one edge from E_1 in its interior. On the other hand, suppose $S_2 \subseteq V_2$ is a 2-SPHS of G_2. Then every edge $e = (u, v) \in E_1$ is covered by S_2 because u, v are the only interior vertices of the shortest path from u_e to v_e, and one of them must be in S_2. The claim is proved.

Finally, the SPHS problem is in NP because one can verify whether a given set of vertices hits every shortest path of a graph of length more than r in polynomial time. ◀

3 The algorithm

In this section, we describe a bi-criteria approximation algorithm for the SPHS problem, whose worst-case running time is near-linear in the size of the input graph.

Let $\delta \in [10\alpha/r, 2/15]$ be a parameter. We assume that the highway dimension h and the doubling dimension d are constants. We first give a high level overview of the algorithm, which consists of three main steps.

(i) Compute a set $C \subseteq V$ of "center" vertices of size $O(\kappa/\delta^d)$, such that every vertex of V is within distance $O(\delta r)$ from some center in C.

(ii) Construct a set of shortest paths, called *critical paths*, between center vertices of length roughly $r/2$, such that between every pair of vertices in V, an approximately shortest path can be constructed by concatenating critical paths.

(iii) Compute hitting set $\tilde{H}$ for critical paths, and return $\tilde{H}$.

Next, we describe the details of each step in the following subsections.

3.1 Computing centers

We compute the set C of center vertices using a greedy algorithm, which was originally proposed for the k-center problem (i.e., find k vertices so that the distance to the farthest vertex from them is as small as possible). Initially, add an arbitrary vertex c_1 to C; in the

i-th iteration, add to C the vertex c_i that is farthest from C. The algorithm terminates when $\mu(v, C) \leq \delta r/8$ for all $v \in V$.

For $i \geq 1$, let C_i be the set of chosen vertices after i iterations.

▶ **Lemma 7.** *During the entire algorithm, for any pair $c_i \neq c_j \in C$, $\mu(c_i, c_j) \geq \delta r/8$.*

Proof. Suppose there exist two centers $c_i, c_j \in C$ with $i < j$ such that $\mu(c_i, c_j) < \delta r/8$. Then $\mu(c_j, C_{j-1}) < \delta r/8$, which means the algorithm terminates before adding c_j to C. ◀

The next lemma upper bounds the number of center vertices added to C.

▶ **Lemma 8.** $|C| = O(\kappa/\delta^d)$.

Proof. Let H^* denote the optimal r-SPHS of size κ. Then by the definition of r-SPHS, $\mu(v, H^*) \leq r$ for all $v \in V$ because otherwise a shortest path with v as an endpoint is not r-EV feasible. By the same analysis of the greedy algorithm for the k-center problem [25], we can claim that for all $v \in V$, $\mu(v, C_\kappa) \leq 2r$. In other words, $V \subseteq \bigcup_{c \in C_\kappa} \mathcal{B}(c, 2r)$. Recall that the doubling dimension of the shortest path metric of G is d. By definition, a radius-$2r$ ball can be covered by $O(\delta^{-d})$ balls of radius $\delta r/16$. Thus, V can be covered by $x = O(\kappa\delta^{-d})$ balls of radius $\delta r/16$. Again by the property of the k-center greedy algorithm, adding x centers greedily to C can guarantee that every vertex of V is within distance $2 \times (\delta r/16) = \delta r/8$ of some center in C. ◀

The greedy algorithm can be implemented efficiently, as follows. Let D denote the diameter of G; then $D \leq \alpha n < n\delta r$. We maintain the distance from each vertex of V to C in a priority queue; initially, the distance is ∞ as $C = \emptyset$. Suppose the shortest path distance from c_i to C is x_i when c_i is added to C. To find c_{i+1}, we compute the shortest path tree rooted at c_i that contains vertices of V whose distances to c_i are less than x_i, and updates the priority queue if the distance from some vertex v to C is decreased because of c_i. We then choose the first vertex of the priority queue (farthest from C) to be c_{i+1}.

▶ **Lemma 9.** *The greedy algorithm for computing the set C of centers takes $O(n \log^2 n + m \log n)$ time.*

Proof. To analyze the running time, we divide the above implementation into $O(\log \frac{D}{\delta r})$ phases. In phase j, the farthest distance from a vertex to C lies in $(\frac{D}{2^j}, \frac{D}{2^{j-1}}]$. If a vertex v is traversed when computing the shortest path tree rooted at a center c, then $\mu(v, c) \leq D/2^{j-1}$. On the other hand, any two centers chosen in phase j have distance more than $\frac{D}{2^j}$. So there can be at most $2^d = O(1)$ centers that traverse v when computing shortest path tree in phase j. Similarly, each edge is also traversed $O(1)$ times in phase j. It takes $O(\log n)$ time to traverse a vertex and $O(1)$ time to traverse an edge in Dijkstra's algorithm and $O(\log n)$ time to update the priority queue. Therefore, the running time is $O((m + n \log n) \log \frac{D}{\delta r}) = O(n \log^2 n + m \log n)$. ◀

We remark that the set C is a δ-approximate r-SPHS. However, the size of C, $O(\kappa/\delta^d)$, can be very large when δ is small. Our algorithm computes a solution of size $O(\kappa \log \kappa)$.

3.2 Computing critical paths

For each vertex $c \in C$, we construct a shortest path tree T_c, called a *center tree*, rooted at c with radius $r/2$, i.e., T_c contains all vertices of V that are no more than $r/2$ away from c. For every $c' \in C$ with $\mu(c, c') \in [(1 - \delta)\frac{r}{2} - \alpha, \frac{r}{2}]$, we add the shortest path $\pi(c, c')$ as a critical path.

▶ **Lemma 10.** *The number of critical paths is* $O(\frac{\kappa}{\delta^{2d}})$*, and they can be computed in* $O(\frac{1}{\delta^d}(m + n \log n))$ *time.*

Proof. Consider any center $c \in C$. We bound the number of critical paths that has c as one endpoint. By construction, if there is a critical path between c and some $c' \in C$, then $\mu(c, c') \le r/2$, i.e., $c' \in \mathcal{B} = \mathcal{B}(c, r/2)$. By definition of doubling dimension, $\mathcal{B}$ can be covered by $O(1/\delta^d)$ smaller balls of radius $\delta r/16$. By Lemma 7, there can be at most one center inside each smaller ball, so, there are $O(1/\delta^d)$ centers in $\mathcal{B}$. The bound on the number of critical paths follows. Similarly, a vertex $v \in V$ or an edge $e \in E$ is traversed during the construction of $O(1/\delta^d)$ center trees. Summing over all center trees, the total time is $O(\frac{1}{\delta^d}(m + n \log n))$. ◀

3.3 Computing approximate hitting set

We compute an approximate hitting set of the critical paths using an algorithm framework by Agarwal and Pan [4]: Let $\mathcal{R}$ denote the set of ranges induced by the critical paths, i.e., each range in $\mathcal{R}$ corresponds to the set of interior vertices of a critical path. Let $\mathcal{C} = (V, \mathcal{R})$ be the resulting range space. By Lemma 5, $\mathcal{C}$ has VC-dimension 2. Let λ denote the size of the optimal hitting set of $\mathcal{C}$. We guess an integer $\tilde{\lambda}$ via binary search such that $\lambda \le \tilde{\lambda} < 2\lambda$.

At a high level, the algorithm works in three stages: the *preprocessing* stage removes some vertices and ranges such that no remaining range contains too many vertices; the *weight-assignment* stage assigns a non-negative weight to each vertex so that every range in $\mathcal{R}$ is $(1/2\tilde{\lambda}e)$-heavy; and the *net-construction* stage computes an $(1/2\tilde{\lambda}e)$-net of $\mathcal{C}$. Since every range in $\mathcal{R}$ is $(1/2\tilde{\lambda}e)$-heavy, the third stage computes a hitting set of $\mathcal{C}$.

Preprocessing stage. In this stage, we compute a $\frac{1}{\tilde{\lambda}}$-net H_0 of $(V, \mathcal{R})$ with uniform weights on V, and include H_0 in the final hitting set. We then (conceptually) remove H_0 and all ranges in $\mathcal{R}$ hit by H_0 from consideration. By definition of ε-net, no remaining range in $\mathcal{R}$ contains more than $n/\tilde{\lambda}$ vertices. This property ensures that the weight-assignment stage has small running time. A simple ε-net construction algorithm is described in the net-construction stage. To remove ranges of $\mathcal{R}$ hit by H_0, we traverse all the center trees and mark every critial path hit by H_0, which takes $O(\sum_{c \in C} |T_c|) = O(n/\delta^d)$ time.

Weight-assignment stage. Recall that given a weight function $w : V \to \mathbb{R}_{\ge 0}$, a range $R \in \mathcal{R}$ is called *ε-heavy* if $w(R) \ge \varepsilon w(V)$; otherwise, R is *ε-light*. The algorithm assigns the weights in $O(\log(n/\tilde{\lambda}))$ rounds. Initially, the $w(v) = 1$ for all $v \in V$.

In each round, the algorithm processes every range $R \in \mathcal{R}$ one by one. If R is $\frac{1}{2\tilde{\lambda}}$-light, it doubles the weights of all vertices in R, the so-called *weight-doubling* step, repeatedly until R becomes $\frac{1}{2\tilde{\lambda}}$-heavy. Once R becomes $\frac{1}{2\tilde{\lambda}}$-heavy, it is not processed again in the current round, even though it may become $\frac{1}{2\tilde{\lambda}}$-light again later in the current round while $w(V)$ increases. If $2\tilde{\lambda}$ weight-doubling steps have been performed in the current round, the algorithm aborts the current round and moves to the next round. On the other hand, if all ranges have been processed with less than $2\tilde{\lambda}$ weight-doubling steps, the algorithm terminates.

The argument in [4] shows that if $\tilde{\lambda} \ge \lambda$, the algorithm always terminates with all ranges being ε-heavy with $\varepsilon = \frac{1}{2\tilde{\lambda}e}$. If the algorithm terminates and some ranges are still ε-light, we double the value of $\tilde{\lambda}$ and repeat the algorithm. The data structure described below will be used to compute the current weight of a range and to double it efficiently, the only two nontrivial steps in this stage.

Net-construction stage. The algorithm returns a ε-net, for $\varepsilon = \frac{1}{2\tilde{\lambda}e}$, of $\mathcal{C}$ as a hitting set of $\mathcal{C}$. By Lemma 4, a natural algorithm for computing an ε-net of $(V, \mathcal{R})$ is to draw $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ random samples from V, with respect to the final weights on the vertices in V. We then verify whether the set of samples is an ε-net of $\mathcal{C}$: traverse all the center trees and check whether each ε-heavy critical path is hit. This takes $O(\sum_{c\in C}|T_c|) = O(n/\delta^d)$ time. If the samples do not form an ε-net, we repeat the above steps. In expectation, $O(1)$ repetitions are required. Therefore, an ε-net of the range space $\mathcal{C}$ of size $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ can be computed in $O(\frac{n}{\delta^d} + \frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ expected time.

Data structure. We maintain all the center trees and the weights of vertices in these trees using the *dynamic trees* data structure [20]. The data structure was proposed to maintain a forest of rooted trees where each tree vertex has an arbitrary number of unordered child vertices and the vertices have weights. The main operations supported include:

- *root*(v): Return the root of the tree containing vertex v.
- *link*(v, u): Make vertex v a new child of vertex u by adding edge (v, u). This assumes v, u are in different trees and v is the root of its tree.
- *cut*(v): Delete the edge between vertex v and its parent.
- *path-aggregate*(v): Return an aggregate, such as max/min/sum, of the weights of vertices on the path from v to *root*(v).
- *update*(v, x): Add x to the weight of each vertex on the path from v to *root*(v).

Each of the above operation takes $O(\log\sum_{c\in C}|T_c|) = O(\log n)$ time [20]. In our case, the structure of the center trees remain the same, so we do not use the *link, cut* operations.

We *retrieve the weight* of a critical path using the *path-aggregate* operation, which is the sum of weights of the vertices along a path from some center vertex c to *root*(c). We *double the weight* of an individual vertex v by running *update*$(v, w(v))$ and *update*$(parent(v), -w(v))$. Note that a vertex v can appear in as many as $O(1/\delta^d)$ center trees. Thus, when we update the weight of a vertex v, we make the update for all copies of v in $O(1/\delta^d)$ center trees.

The results of computing an approximate hitting set of $\mathcal{C}$ is summarized as follows.

▶ **Lemma 11.** *A hitting set of $\mathcal{C}$ of size $O(\lambda\log\lambda)$ can be computed in $O((\frac{1}{\delta^d}n\log^2 n + \lambda\log\lambda)\log\lambda)$ expected time, where λ is the size of the optiml hitting set of $\mathcal{C}$.*

Proof. The size of the hitting set is equal to the size of the $\frac{1}{2\tilde{\lambda}e}$-net computed in the net-construction stage of the algorithm, which is $O(\tilde{\lambda}\log\tilde{\lambda}) = O(\lambda\log\lambda)$. The preprocessing and the net-construction stages both involve computing an ε-net, and take time $O(n/\delta^d + \lambda\log\lambda)$. In each round of the weight-assignment stage, retrieving the weights of the ranges in $\mathcal{R}$ takes $O(\frac{\kappa}{\delta^d}\log n) = (\frac{n}{\delta^d}\log n)$ time. There are at most $2\tilde{\lambda}$ weight-doubling steps, and each weight-doubling step updates the weights of no more than $n/\tilde{\lambda}$ vertices. Therefore, the weight-doubling steps take $O(\frac{n}{\delta^d}\log n)$ time in each round. With $O(\log n)$ rounds in the weight-assignment stage and $O(\log\lambda)$ itertions of guessing $\tilde{\lambda}$, the total running time of the algorithm is $O((\frac{1}{\delta^d}n\log^2 n + \lambda\log\lambda)\log\lambda)$. ◀

4 Analysis

We now analyze the performance of our algorithm.

▶ **Lemma 12.** $\lambda = O(h\kappa)$.

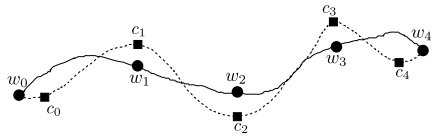

Figure 2 Construction of EV-feasible path $\tilde{P}$ (dashed curve) between w_0 and w_4. The solid curve denotes the shortest path.

Proof. Let H^* denote the optimal r-SPHS of G. By definition, H^* must hit all shortest paths that are longer than r. On the other hand, the critical paths constructed by our algorithm have lengths no more than $r/2$. Let P be a critical path between a pair of vertices u, v. Then there is a vertex $w \in H^*$ with $\mu(u, w) \leq r$. So $P \subseteq \mathcal{B}(w, 3r/2)$. In other words, each critical path is contained in the ball of radius $3r/2$ centered at some vertex in H^*. By definition of highway dimension, for any $w \in H^*$, there exists a subset S of at most h vertices in $\mathcal{B}(w, 3r/2)$ that intersect every shortest path of length more than $r/4$ contained in $\mathcal{B}(w, 3r/2)$. Let $\mathcal{S}$ denote the union of such subsets S in the balls centered at vertices in H^*. With $\delta < 2/15$ and $\alpha \leq \delta r/10$, the interior of each critical path has length more than $r/4$. Therefore, $\mathcal{S}$ hits all the critical paths, and $|\mathcal{S}| = O(h\kappa)$. ◀

Let $\tilde{H}$ denote the hitting set computed by our algorithm. Lemmas 11 and 12 immediately imply the following corollary:

▶ **Corollary 13.** $|\tilde{H}| = O(h\kappa \log(h\kappa))$.

We show that $\tilde{H}$ satisfies the following property.

▶ **Lemma 14.** *$\tilde{H}$ is a δ-approximate r-SPHS of G.*

Proof. If $\mu(u, v) \leq r$, $\pi(u, v)$, the shortest path between u, v, is automatically EV-feasible. We therefore focus on the case $\mu(u, v) > r$. We construct another path $\tilde{P}$ between u, v from $\pi(u, v)$ as follows. For convenience, denote $w_0 = u$ and $w_t = v$. We find vertices $w_1, \cdots, w_{t-1}$ along $\pi(u, v)$ from u to v such that $\mu(w_i, w_{i+1}) \in [(\frac{1}{2} - \frac{\delta}{4})r - \alpha, (\frac{1}{2} - \frac{\delta}{4})r]$, for $i = 1, \cdots, t-1$. Let $c_i \in C$ denote the nearest center to w_i. We set $\tilde{P}$ as the concatenation of the shortest paths $\pi(w_0, c_0), \pi(c_0, c_1), \cdots, \pi(c_{t-1}, c_t), \pi(c_t, w_t)$. See Figure 2. Then

$$\begin{aligned}
\ell(\tilde{P}) &= \mu(w_0, c_0) + \mu(c_t, w_t) + \sum_{i=1}^{t-1} \mu(c_i, c_{i+1}) \\
&\leq \frac{\delta}{4} r + \sum_{i=1}^{t-1} (\mu(c_i, w_i) + \mu(w_i, w_{i+1}) + \mu(w_{i+1}, c_{i+1})) \\
&\leq \frac{\delta}{4} r + \sum_{i=1}^{t-1} (1 + \frac{3\delta}{4}) \mu(w_i, w_{i+1}) \qquad (\delta \leq 2/15) \leq (1 + \delta)\mu(u, v).
\end{aligned}$$

Next, we show that path $\tilde{P}$ is EV-feasible with respect to $\tilde{H}$. By triangle inequality, it is easy to check that $\mu(c_i, c_{i+1}) \in [(\frac{1}{2} - \frac{\delta}{2})r - \alpha, \frac{1}{2}r]$; thus $\pi(c_i, c_{i+1})$ is a critical path and contains a vertex of $\tilde{H}$ in its interior. $\tilde{P}$ is EV-feasible since every subpath of $\tilde{P}$ of length larger than r contains a vertex of $\tilde{H}$. ◀

Putting Lemmas 9, 10, 11, and 12 together, the expected running time of our algorithm is $O(\frac{1}{\delta^d}(m + n \log^2 n \log \kappa) + m \log n) = O(\frac{1}{\delta^d} n \log^2 n \log \kappa)$ with the assumption $m = O(n)$. This bound along with Corollary 13 and Lemma 14 proves Theorem 1.

5 Feasible path query

Given a δ-approximate r-SPHS H, we consider the task of computing the shortest r-EV-feasible path between any two vertices $u, v \in V$ with respect to H. By definition of approximate SPHS, the length of this path is at most $(1+\delta)\mu(u,v)$. A shortest feasible path can be compactly represented by the sequence of charging stations in H it passes through; the distance between any two consecutive stations is at most r. We can recover the whole feasible path by retrieving the shortest paths between consecutive stations in G.

We first show that the δ-approximate r-SPHS $\tilde{H}$ output by our algorithm can be postprocessed and replaced with a smaller δ-approximate r-SPHS $\hat{H}$ such that $|\hat{H} \cap \mathcal{B}(v,r)|$ is small for any $v \in V$.[2] This property of $\hat{H}$ ensures small feasible path query time with respect to a set of charging stations $\hat{H}$.

Postprocessing step. We show that $\tilde{H}$ can be replaced by another approximate r-SPHS $\hat{H}$ such that $|\hat{H}| \leq |\tilde{H}|$ and for any $u \in \hat{H}$, $|\mathcal{B}(u,r) \cap \hat{H}| = O(1)$, where the constant depends on the highway dimension of G. The algorithm works as follows.

The algorithm maintains an r-SPHS H. Initially, $H = \tilde{H}$. For each vertex $v \in V$, it also maintains the set $H_v = \{u \in H \mid \mu(u,v) \leq r\}$, i.e., $\mathcal{B}(v,r) \cap H$, and the value $|H_v|$. We fix a constant c and call a vertex $v \in V$ *heavy* if $|H_v| > ch \ln h$. At each step, the algorithm checks whether there is a heavy vertex in V. If there is no heavy vertex, it returns the current set H as $\hat{H}$. Otherwise, let v be a heavy vertex. Let $\Sigma_v = (V, \mathcal{R}_v)$ be a range space where $\mathcal{R}_v$ corresponds to critical paths intersecting $\mathcal{B}(v,r)$. Since each critical path has length no more than $r/2$, all the critical paths in $\mathcal{R}_v$ lie inside $\mathcal{B}(v, 3r/2)$. By definition of highway dimension, there exists a hitting set of size h for Σ_v. We can use the same hitting-set algorithm [4] to compute a hitting set X_v of Σ_v of size at most $ch \ln h$ in $O(\frac{1}{\delta^d} n \log^2 n)$ expected time. It then replaces H with $(H \setminus H_v) \cup X_v$. Finally, we compute $\mathcal{B}(u,r)$ for each $u \in X_v$ and update the sets H_w for all w in these balls.

Since v is heavy, each step of the algorithm except the last one reduces the size of H by at least one, so it terminates within $|\tilde{H}|$ rounds. $\hat{H}$ is a δ-approximate r-SPHS since it hits every critical path. Hence, we obtain the following.

▶ **Lemma 15.** *A δ-approximate r-SPHS $\hat{H} \subseteq V$ of size $O(\kappa \log \kappa)$ can be computed in $O(\frac{1}{\delta^d} \kappa n \log^2 n \log \kappa)$ time so that $|\mathcal{B}(v,r) \cap \hat{H}| = O(1)$ for all $v \in V$.*

Feasible path query. A shortest r-EV-feasible path must pass through a sequence of charging stations, and any two consecutive charging stations on the path must be at most r apart. Define the graph $\mathcal{N} = (\hat{H}, \hat{E})$ where $\hat{E} = \{(u,v) \mid \mu(u,v) \leq r\}$. For each edge $(u,v) \in \hat{E}$, define $\ell(u,v) = \mu(u,v)$. By Lemma 15, $|\hat{E}| = O(|\hat{H}|) = O(\kappa \log \kappa)$. By constructing $\mathcal{B}(u,r)$ for all $u \in \hat{H}$, we can construct the edges in $\hat{E}$ and their lengths.

As for computing a shortest feasible path between any pair of vertices of G, we maintain, for each $v \in V$, $\hat{H}_v = \mathcal{B}(v,r) \cap \hat{H}$ along with their distances from v. Given $s, t \in V$, we augment $\mathcal{N}$ by adding edges from s to $\hat{H}_s$ and t to $\hat{H}_t$, and compute a shortest path from s to t in $\mathcal{N}$ using the Dijkstra's algorithm. Putting everything together, we obtain the following.

▶ **Theorem 16.** *Let $G = (V, E)$, $\ell : V \to \mathbb{R}^+$ be a weighted graph of constant highway dimension h, with $|V| = n$ and $|E| = O(|V|)$. Let $r > 0$ and $\delta \in [\frac{10\alpha}{r}, \frac{2}{15}]$ be two para-*

[2] We conjecture that $\tilde{H}$ already satisfies $|\mathcal{B}(u,r) \cap \tilde{H}| = O(\log \kappa)$ for all $u \in V$, and no postprocessing is needed, but so far we have run into technical difficulties in proving this conjecture.

meters where $\alpha = \max_{e \in E} \ell(e)$*, and let* κ *be the size of a smallest* r*-SPHS of* G*. In time* $O(\frac{1}{\delta^d} \kappa n \log^2 n \log \kappa)$*, a* δ*-approximate* r*-SPHS* $\hat{H}$ *can be computed and* G *can be processed into a data structure of size* $O(\kappa \log \kappa)$ *such that for any two vertices* $s, t \in V$*, a compact representation of a shortest* r*-EV-feasible path from* s *to* t*, using* $\hat{H}$*, can be computed in* $O(\kappa \log^2 \kappa)$ *time.*

6 Conclusion

In this paper, we presented a bi-criteria approximation algorithm for the r-SPHS problem whose running time is near-linear in n. The algorithm assumes the input graph has constant highway dimension, a concept introduced to give rigorous proofs of efficiency for many popular heuristic shortest path algorithms [3]. Our algorithm is the first for such problems with provable guarantees on the approximation and running time. We also give an algorithm for computing the shortest EV-feasible paths given the set of charging stations computed by the first algorithm.

It is also interesting to know whether it is possible to improve the size of the ε-net for the range space of shortest paths from $O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ to $O(\frac{1}{\varepsilon})$. If so, it will improve the approximation ratio of our algorithm from $O(\log \kappa)$ to $O(1)$.

Additionally, no efficient algorithm is known for the maximum coverage version of the SPHS problem. Given a collection $\mathcal{P}$ of input paths in a graph G (may or may not be shortest) and an integer k, the goal is to compute a subset of k vertices such that the number of EV-feasible paths in $\mathcal{P}$ (with respect to the subset) is maximized . This problem is not submodular because it can take more than one vertex to make one long path in $\mathcal{P}$ EV-feasible.

Finally, we have shown that the SPHS problem is NP-complete for general graphs, but we do not known whether it is NP-complete for graphs of constant highway dimension. A proof of NP-completeness may require more insights into the structure of such graphs.

Acknowledgements. We thank Dan Halperin and Eli Packer for introducing the problem to us and for helpful discussions.

References

1 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In *Proc. 38th Int'l Colloq. Conf. Automata, Languages and Programming*, pages 690–699, 2011.

2 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V Goldberg, and Renato F Werneck. Highway dimension and provably efficient shortest path algorithms. *Tech. Report MSRTR-2013-91, Microsoft Research*, 2013.

3 Ittai Abraham, Amos Fiat, Andrew V Goldberg, and Renato F Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *Proc. 21st Annual ACM-SIAM Symp. Discrete Algorithms*, pages 782–793, 2010.

4 Pankaj K Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. In *Proc. 30th Annual Symp. Comput. Geo.*, page 271, 2014.

5 Boris Aronov, Esther Ezra, and Micha Sharir. Small-size ε-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010.

6 Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. *arXiv preprint*, 2015.

7 Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. In *Proc. 23rd SIGSPATIAL Int'l Conf. Advances in Geo. Info. Syst.*, page 44, 2015.

8 Moritz Baum, Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. Energy-optimal routes for electric vehicles. In *Proc. 21st ACM SIGSPATIAL Int'l Conf. Advances in Geo. Info. Syst.*, pages 54–63, 2013.

9 Vasek Chvatal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

10 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proc. 46th Annual ACM Symp. Theory of Computing*, pages 624–633. ACM, 2014.

11 Inês Frade, Anabela Ribeiro, Gonçalo Gonçalves, and António Antunes. Optimal location of charging stations for electric vehicles in a neighborhood in lisbon, portugal. *Transportation Res. Record: J. Transportation Res. Board*, 2252(2252):91–98, 2011.

12 Stefan Funke, André Nusser, and Sabine Storandt. On k-path covers and their applications. *Proc. VLDB Endowment*, 7(10):893–902, 2014.

13 Stefan Funke, André Nusser, and Sabine Storandt. Placement of loading stations for electric vehicles: No detours necessary! *J. Arti. Intelli. Res.*, pages 633–658, 2015.

14 Michael T Goodrich and Paweł Pszona. Two-phase bicriterion search for finding fast and efficient electric vehicle routes. In *Proc. 22nd ACM SIGSPATIAL Int'l Conf. Advances in Geo. Info. Syst.*, pages 193–202, 2014.

15 Sariel Har-Peled. *Geometric Approximation Algorithms*. American Math. Soc., 2011.

16 David Haussler and Emo Welzl. ε-nets and simplex range queries. *Discrete & Comput. Geo.*, 2(2):127–151, 1987.

17 RM Karp. Reducibility among combinatorial problems. *Complexity Comp. Comput.*, 1972.

18 Nabil H Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Comput. Geo.*, 44(4):883–895, 2010.

19 Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Proc. 25th AAAI Conf. Arti. Intelli.*, 2011.

20 Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comp. Syst. Sci.*, 26(3):362–391, 1983.

21 Sabine Storandt and Stefab Funke. Enabling e-mobility: Facility location for battery loading stations. In *Proc. 27th AAAI Conf. Arti. Intelli.*, 2013.

22 Moby Khan T. Donna Chen, Kara M. Kockelman. The electric vehicle charging station location problem: a parking-based assignment method for seattle. In *Proc. 92nd Annual Meet. Transportation Res. Board*, 2013.

23 Yufei Tao, Cheng Sheng, and Jian Pei. On k-skip shortest paths. In *Proc. ACM SIGMOD*, pages 421–432. ACM, 2011.

24 VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264, 1971.

25 Vijay V Vazirani. *Approximation algorithms*. Springer Sci. & Business Media, 2013.

26 Yanhai Xiong, Jiarui Gan, Bo An, Chunyan Miao, and Ana LC Bazzan. Optimal electric vehicle charging station placement. In *Proc. 24th Int'l Joint Conf. Arti. Intelli.*, pages 2662–2668, 2015.