Computational Geometry: Theory and Applications 98 (2021) 101787

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Sparse convex hull coverage

Georgiy Klimenko [1], Benjamin Raichel [*,1], Gregory Van Buskirk [1]

Department of Computer Science, University of Texas at Dallas, 800 W. Campbell Road, Richardson, TX 75080, USA

ARTICLE INFO

Article history:
Received 21 October 2020
Received in revised form 30 April 2021
Accepted 3 May 2021
Available online 14 May 2021

Keywords:
Convex hull
Approximation
Hardness

ABSTRACT

Given a set P of n data points and an integer k, a fundamental computational task is to find a smaller subset $Q \subseteq P$ of only k points which approximately preserves the geometry of P. Here we consider the problem of finding the subset Q of k points which best captures the convex hull of P, where our error measure is the sum of the distances of the points in P to the convex hull of Q. We generalize the problem to allow the set R that we must select Q from to differ from P, as well as to allow more general functions of the distances of the uncovered points of P, such as other norms or weighted distance functions.

We prove that approximating the convex hull in this manner in the plane can be solved by either a simple graph based or dynamic programming based algorithm in polynomial time. Complementing this result we show that in three dimensions and higher the problem is NP-hard. Moreover, we give an algorithm which in three dimensions selects $O(k \log(n/\varepsilon))$ points to get a solution whose error is at most $1+\varepsilon$ times the optimal k point error. This generalizes to $O(k^{\lfloor d/2 \rfloor} \log(n/\varepsilon))$ points for any constant dimension d.

1. Introduction

Given a point set $P \subset \mathbb{R}^d$, the convex hull of P, denoted by $\mathcal{CH}(P)$, is a fundamental geometric structure, intuitively capturing the region covered by P. Here we consider the problem of covering P as best as possible by the convex hull of a subset of only k points from P, in effect sparsely approximating $\mathcal{CH}(P)$. This natural problem relates to the problem of approximating convex sets by polytopes, for which countless papers have been written (see the extensive survey [6]). Much of this previous work has focused on the objective of minimizing the maximum distance of an uncovered point from the hull of the selected points (i.e. Hausdorff distance), or approximating the volume in the case of smooth convex bodies. Here we instead study approximating the convex hull of a discrete point set under the objective of minimizing the sum of the distances of the uncovered points, an objective which when compared to the max objective is more robust to outliers as the error is no longer determined solely by the single furthest point. Our framework also allows for much more general cost functions of the distances, and in particular allows for any ℓ_p norm or weighted distance functions. We further generalize the problem such that the selected k points defining our hull are required to come from a set R that can differ from P, thus capturing scenarios where the covering objects differ from the covered ones. This is natural from a feature selection standpoint, where R represents a set of known possible features which we wish to represent a set of observed objects P. For such problems the convex hull is a particularly relevant structure as it represents the set of all weighted averages of the selected points. Moreover, the Carathéodory theorem states that any point in the convex hull of the chosen subset can be

* Corresponding author.
E-mail addresses: gik140030@utdallas.edu (G. Klimenko), benjamin.raichel@utdallas.edu (B. Raichel), greg.vanbuskirk@utdallas.edu (G. Van Buskirk).
[1] This work was partially supported by NSF CAREER Award 1750780.

https://doi.org/10.1016/j.comgeo.2021.101787
0925-7721/

represented as a convex combination of $d+1$ of the chosen points, yielding a sparse representation in low dimensions. (In higher dimensions one can use the approximate Carathéodory theorem [3].)

More generally, given a set $P \subset \mathbb{R}^d$ of n points, finding a smaller set of only k points which approximately captures the geometry of P under some measure is a ubiquitous computational task. Two standard such problems of interest are k-clustering and subspace fitting. In k-clustering the objective is to select a subset of k center points so as to minimize some norm of the vector of distances from each point in P to its nearest center. For example, k-means seeks to minimize the ℓ_2 norm [2], where it is known that even planar k-means is NP-hard [14]. For subspace fitting the objective is to select the k-dimensional subspace minimizing some norm of the distances to the linear subspace, e.g. the solution under the ℓ_2 norm is known to be the top k singular vectors when viewing P as a matrix. If one restricts the selected k points to come from P, then the clustering and subspace fitting problems become the standard discrete k-clustering and CUR-decomposition [5] problems.

Our problem of approximating the convex hull can be viewed as naturally lying between clustering and subspace fitting, when restricting the selected subset to come from a set R. Specifically, viewing the selected subset of k points $Q \subseteq R$ as a basis, the problems are defined by how we allow each point in P to be represented by Q. In subspace fitting, any linear combination is allowed, in convex hull coverage only convex combinations are allowed (i.e. non-negative coefficients that sum to 1), and in clustering not only are the combinations convex, but the coefficients are all zero except for a single coefficient being 1 (i.e. the nearest center). That is, one can define an entire spectrum of problems based on how one restricts reconstruction from the basis, and convex hull coverage is a natural setpoint on this spectrum. In this sense, other standard problems such as non-negative matrix factorization (NMF), which is known to be NP-hard [19], can be seen as another setpoint on this spectrum. (NMF typically restricts the basis to non-negative vectors, though restricting to input points is also commonly studied [13].)

Another related topic is coresets, which are small subsets of the input which can be used as a proxy for the full set. There are numerous coresets results (see chapter 48 in [16]). Relevant to the current paper, it is known that for any point set P contained in the unit ball,[2] there is a subset $S \subseteq P$ of $O(1/\varepsilon^{(d-1)/2})$ points such that all of P is with distance ε from $\mathcal{CH}(S)$. Worst case point sets require such an exponential dependence on d, and thus Blum *et al.* [4] considered coresets whose size is measured relative to the given instance, showing that if some set of k points achieves ε error, then a greedy algorithm selecting $O(k/\varepsilon^{2/3})$ points achieves $O(\varepsilon^{1/3})$ error. This result was later extended by Van Buskirk *et al.* [17] to get analogous results for approximating the conic hull, which consists of all non-negative combinations, and so relates to NMF.

Given two nested convex polygons, Aggarwal *et al.* [1] gave a near linear time algorithm to find the convex polygon nested between them with the fewest number of vertices. In higher dimensions, Clarkson [8] gave approximations (on the number of points) when the polytope must be fully covered by the hull of a subset of a discrete point set, or by a subset of points with bounded Hausdorff distance. Among other things, these problems differ from ours in that they require full coverage and attempt to minimize k. Our problem also loosely relates to curve simplification, for which there are many popular heuristics such as Douglas-Peucker [12]. Curve simplification with provable guarantees was considered by van Kreveld *et al.* [18], who showed hardness for Hausdorff distance and gave a polynomial time algorithm for Fréchet distance.

1.1. Our contribution

For point sets $R, P \subset \mathbb{R}^d$ of m and n points, respectively, we initiate the rigorous study of the convex hull coverage problem, where the goal is to find a subset $Q \subseteq R$ of k points minimizing the sum of distances from the points in P to their projection onto the convex hull of Q, that is $\sum_{p \in P} ||p - \mathcal{CH}(Q)||$, where $||p - \mathcal{CH}(Q)||$ denotes the distance from p to the convex hull of Q. Furthermore, we generalize the problem to allow any cost function of the form $\sum_{p \in P} g_p(||p - \mathcal{CH}(Q)||)$, where each g_p can be any monotonically increasing real valued function such that $g_p(\alpha) = 0$ if and only if $\alpha = 0$. Thus we can model for example weighted sums or other ℓ_p norms of the distances of the points in P to the hull (by taking the pth power of the norm).

We prove that convex hull coverage can be solved exactly in the plane in $O(m^3k + m^2n + mn\log(n))$ time via dynamic programming. Interestingly, for the special case when $P = R$, we can show that by carefully assigning weights, the problem nicely reduces to the problem of finding a minimum cost k length cycle in a directed graph. This yields a simpler graph based algorithm with $O(n^3 \log k)$ running time. To complement our results in the plane, we argue that the convex hull coverage problem is NP-hard for $d \geq 3$, even when restricting our objective to the sum of distances (i.e. the g_p are all the identity function). Furthermore, we argue that even if one restricts to instances where $P = R$, the problem remains NP-hard for $d \geq 4$. Finally, we argue that a geometric set cover based algorithm yields an approximation in constant dimensions for the sum of distances. Namely, for $d = 3$ greedily selecting $O(k \log(n/\varepsilon))$ points in an appropriate way gives a solution whose error is at most $1 + \varepsilon$ times the optimal k point error. This generalizes to $O(k^{\lfloor d/2 \rfloor} \log(n/\varepsilon))$ points for any constant dimension d.

One of the main challenges of convex hull coverage for $d \geq 3$ is that it lacks certain independence properties of related problems. For example, in k-clustering, the cluster centers partition the points based on their nearest center, whereas the

[2] Any point set can be scaled to lie in the unit ball, effectively meaning ε is measured relative to the diameter before scaling, which is in some sense necessary. Via an affine transformation, one can ague such coresets exist for directional width where error is relative to the diameter in each direction, see [11].

projection of a point onto the convex hull is determined by several hull vertices. For subspace approximation under the Frobenius norm there is independence among the dimensions, in the sense that the k-th singular vector is determined by finding the optimum vector in the orthogonal subspace of the first $k-1$ singular vectors. Note also that previous coreset results focused on the max measure, where a given error ε represents a precise constraint that all points must satisfy. On the other hand, for our sum measure, an error ε represents a budget that the algorithm must now decide how to allocate amongst the various points.

1.2. Preliminaries

Given a point set X in $\mathbb{R}^d$, let $\mathcal{CH}(X)$ denote its convex hull. For two points $x, y \in \mathbb{R}^d$, let xy denote their line segment, that is $xy = \mathcal{CH}(\{x, y\})$. Throughout, given points $x, y \in \mathbb{R}^d$, $||x-y||$ denotes their Euclidean distance. Given two compact sets $X, Y \subset \mathbb{R}^d$, $||X-Y|| = \min_{x\in X, y\in Y} ||x-y||$ denotes their distance. For a single point x we write $||x-Y|| = ||\{x\}-Y||$.

Definition 1.1. Let $P \subset \mathbb{R}^d$ be a set of n points, where for each point $x \in P$, there is an associated monotonically increasing real valued function g_x such that $g_x(\alpha) = 0$ if and only if $\alpha = 0$. Then we call any function of the form $f(Q, P') = \sum_{x\in P'} g_x(||x - \mathcal{CH}(Q)||)$, where $Q \subset \mathbb{R}^d$ and $P' \subseteq P$, a *hull coverage function*. We let $\mathcal{F}_P$ denote the set of all such functions.

In the above definition we assume the g_x functions can be evaluated in constant time. The following is the main problem studied in this paper.

Problem 1.2. Given a set $P \subset \mathbb{R}^d$ of n points, a set $R \subset \mathbb{R}^d$ of m points, and a function $f \in \mathcal{F}_P$, select a subset $Q \subseteq R$ of at most k points which minimizes $f(Q, P)$. That is, $Q = \arg\min_{Q\subseteq R, |Q|\leq k} f(Q, P)$.

2. Exact computation in the plane

In this section we give polynomial time algorithms for Problem 1.2 when $d = 2$. First, we give a simple graph based algorithm for the special case when $P = R$, followed by a slightly more involved dynamic programming algorithm for the general case.

2.1. A graph algorithm for a simpler case

In this section we argue that by assuming $P = R$, one can solve Problem 1.2 in the plane by converting it into a corresponding graph problem. Specifically, construct a weighted and fully connected directed graph $G_P = (V, E)$ where $V = P$. Given an order pair of points p, q, let $P_{p,q}$ denote the subset of P in the closed halfspace whose boundary is the line through p and q and lies to the left of the ray from p to q. Then we define the weight of the directed edge (p, q) to be $w(p, q) = f(\{p, q\}, P_{p,q})$. For a cycle of vertices $C = \{p_1, \ldots, p_k\}$, let $w(C)$ denote the sum of the weights of the directed edges around the cycle. Throughout, we only consider non-trivial cycles, that is cycles must have at least two vertices.

For a set of points Q, let $\mathcal{CH}_L(Q)$ denote the clockwise list of vertices on the boundary of $\mathcal{CH}(Q)$. Observe that any subset $Q \subseteq P$ corresponds to the cycle $\mathcal{CH}_L(Q)$ in G_P. Moreover, any cycle C corresponds to the convex hull $\mathcal{CH}(C)$.

Lemma 2.1. *Consider an instance P, R, f, k of Problem 1.2 in the plane where $P = R$. Let C be any cycle in G_P, and let Q be an optimal solution. Then,*
1) $w(C) \geq f(C, P)$, 2) $w(\mathcal{CH}_L(Q)) = f(Q, P)$.

Proof. First, observe that $w(C)$ and $f(C, P)$ can be decomposed into the contribution of each point.

$$f(C, P) = \sum_{p\in P} g_p(||p - \mathcal{CH}(C)||) \qquad \text{and} \qquad w(C) = \sum_{(a,b)\in C} f(\{a, b\}, P_{a,b}) = \sum_{p\in P} \sum_{\substack{(a,b)\in C \\ \text{s.t. } p\in P_{a,b}}} g_p(||p - ab||).$$

To prove the first part of the lemma, we thus argue that for any $p \in P$, its contribution to $w(C)$ is at least as large as its contribution to $f(C, P)$. Assume $p \notin \mathcal{CH}(C)$, since otherwise it does not contribute to $f(C, P)$. It suffices to argue there exists an edge $(a, b) \in C$, such that $p \in P_{a,b}$, since $||p - ab|| \geq ||p - \mathcal{CH}(C)||$ and g_p is a monotonically increasing function. So assume otherwise that there is some point $p \in P$ such that p lies strictly to the right of all edges in C. Create a line ℓ that passes through p and any interior point of any edge $(a, b) \in C$, but does not pass through any point in P. The line ℓ splits the plane into two halfspaces. As p is to the right of any edge and is outside the convex hull of the points, all edges intersecting ℓ have to begin at the same halfspace and end at the other halfspace. This implies C is not a cycle, which is a contradiction.

To prove the second part of the lemma for an optimal solution Q, we argue that for any $p \in P$, its contribution to $f(Q, P)$ is equal to its contribution to $w(\mathcal{CH}_L(Q))$. If $p \in \mathcal{CH}(Q)$ then it lies to the right of all edges in $\mathcal{CH}_L(Q)$, and so

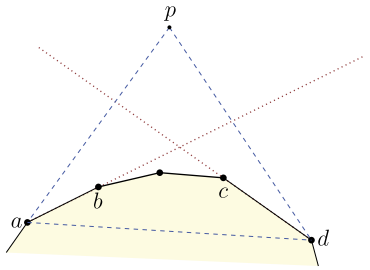

Fig. 2.1. $b, c \in \mathcal{CH}(\{a, d, p\})$ when $p \in P_{ab}$ and $p \in P_{cd}$.

its contributions to both $w(\mathcal{CH}_L(Q))$ and $f(Q, P)$ are zero. So consider a point $p \notin \mathcal{CH}(Q)$. Let ab be the closest edge of $\mathcal{CH}(Q)$ (where b follows a in clockwise order). Note that $||p - \mathcal{CH}(Q)|| = ||p - ab||$ and $p \in P_{ab}$, and thus the contributions of p to $f(Q, P)$ and $w(\mathcal{CH}_L(Q))$ are equal if and only if p lies to right of all other edges in $\mathcal{CH}_L(Q)$, as otherwise p has a positive contribution to another edge since by definition $g_p(\alpha) > 0$ for $\alpha > 0$. So suppose otherwise, that p lies to the left of some other edge cd (note it may be that $b = c$). Thus p is in the intersection of the halfspace to the left of the line from a through b and to the left of the line from c through d, see Fig. 2.1. This implies that $b, c \in \mathcal{CH}(\{a, d, p\})$. So let $Q' = Q \cup \{p\} \setminus \{b, c\}$, then $\mathcal{CH}(Q) \subset \mathcal{CH}(Q')$. This implies $f(Q', P) < f(Q, P)$ as Q' contains p but Q does not, which is a contradiction with Q being an optimal solution as $|Q'| \leq |Q|$. (Note that assuming $P = R$ was used to ensure that Q' was a possible solution.) □

Theorem 2.2. *Given an instance P, R, f, k of Problem 1.2 in the plane where $P = R$, it can be solved in $O(n^3 \log k)$ time, where $n = |P| = |R|$.*

Proof. Let C be a minimum cost cycle in G_P subject to having at most k vertices. The claim is that the set of vertices in C is an optimal solution to Problem 1.2, that is, $f(C, P) = \min_{X \subseteq P, |X| \leq k} f(X, P)$. By part 1) of Lemma 2.1, $w(C) \geq f(C, P)$, and thus if C is not optimal, then the optimal solution must have cost strictly less than $w(C)$. However, by part 2) of Lemma 2.1, the optimal solution corresponds to a cycle in G_P with the same cost, which contradicts C being minimum cost.

Now we analyze the running time. Computing G_P takes $O(n^3)$ time as there are $O(n^2)$ edges, and computing the weight of each edge takes $O(n)$ time, as it is a sum of at most n constant time computable functions. To compute the minimum cost cycle with $\leq k$ edges, it suffices to compute the all pairs shortest path distances for paths with $\leq k - 1$ edges, since afterwards in $O(n^2)$ time we can add the final edge of each cycle. It is known that for a graph with n vertices the all pairs shortest path distances for paths with $\leq k - 1$ edges can be computed in $O(n^3 \log k)$ time, see for example the matrix multiplication algorithm in [9]. Thus the overall running time is $O(n^3 \log k)$. □

2.2. Dynamic programming for the general case

We now argue that when P is allowed to differ from R we can still compute the optimal solution in the plane in polynomial time by using a slightly more involved and slightly slower dynamic program.

Let $V = \{v_1, \ldots, v_k\} \subseteq R$ be the vertices of some convex hull of points from R, labeled in clockwise order, where v_1 is the vertex of V with smallest y-coordinate. Consider our cost function $f(V, P) = \sum_{x \in P} g_x(||x - \mathcal{CH}(V)||)$. Any point $x \in \mathcal{CH}(V)$ contributes zero to f, as we required $g_x(0) = 0$. So consider any point $x \in P$ lying outside of $\mathcal{CH}(V)$. The projection of x onto $\mathcal{CH}(V)$ is either a vertex v_i or a point on the interior of an edge $v_{i-1}v_i$, for some i. Thus the edges and vertices of the hull define a partition of points in P which lie outside the hull, which we now formally describe.

Consider the ray with base point v_{i-1} and directed from v_{i-1} towards v_i. Define $r_l(v_{i-1}, v_i)$ to be the rotation of this ray by $\pi/2$ to the left, that is the ray with base point v_{i-1} and direction $(v_{i-1}.y - v_i.y, v_i.x - v_{i-1}.x)$. Define $r_r(v_{i-1}, v_i)$ to be the ray with the same direction, but with base point v_i. Then $slab(v_{i-1}, v_i)$ is defined as the region of the plane interior to and bounded by the edge $v_{i-1}v_i$ and (between) the rays $r_l(v_{i-1}, v_i)$ and $r_r(v_{i-1}, v_i)$. See Fig. 2.2. Define $cone(v_{i-1}, v_i, v_{i+1})$ as the closed region bounded $r_r(v_{i-1}, v_i)$ and $r_l(v_i, v_{i+1})$, again see Fig. 2.2. In other words, $slab(v_{i-1}, v_i)$ and $cone(v_{i-1}, v_i, v_{i+1})$ are the subsets of points in the plane outside of $\mathcal{CH}(V)$ whose projection onto $\mathcal{CH}(V)$ lies on the interior of $v_{i-1}v_i$ or on the vertex v_i, respectively. In particular, for a point set P, define

$$sum_{slab}(v_{i-1}, v_i) = f(\{v_{i-1}, v_i\}, P \cap slab(v_{i-1}, v_i)) = \sum_{p \in P \cap slab(v_{i-1}, v_i)} g_p(||p - \mathcal{CH}(\{v_{i-1}, v_i\})||)$$

$$sum_{cone}(v_{i-1}, v_i, v_{i+1}) = f(\{v_i\}, P \cap cone(v_{i-1}, v_i, v_{i+1})) = \sum_{p \in P \cap cone(v_{i-1}, v_i, v_{i+1})} g_p(||p - v_i||)$$

Observe that $sum_{slab}(v_{i-1}, v_i)$ only depends on v_{i-1} and v_i and $sum_{cone}(v_{i-1}, v_i, v_{i+1})$ only depends on v_{i-1}, v_i, and v_{i+1}.

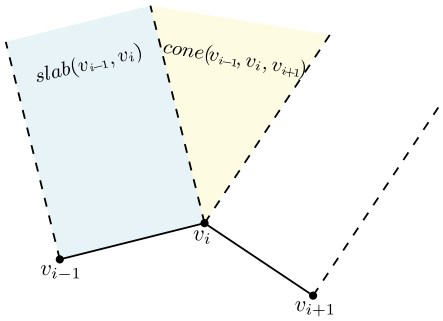

Fig. 2.2. Three consecutive vertices on the hull, and the corresponding defined slabs and cone.

In particular, these quantities are respectively defined for any pair or triple of points in R, and for now assume they have all been precomputed.

By the discussion above, for ordered vertices V of a convex hull we can rewrite our cost function as

$$f(V, P) = \sum_{x \in P} g_x(||x - \mathcal{CH}(V)||) = \sum_{i=1}^{k} (sum_{cone}(v_{i-1}, v_i, v_{i+1}) + sum_{slab}(v_i, v_{i+1})), \tag{2.1}$$

where indices are mod k, i.e. $v_0 = v_k$ and $v_{k+1} = v_1$. This equation suggests a natural recursive strategy to minimize $f(V, P)$ (over choices of V) by guessing the vertices of V in clockwise order.

First, at the cost of an additional linear factor in the running time, we guess the point with the smallest y-coordinate from the optimal hull.[3] We call this the *starting point* and denote it by s (i.e. $v_1 = s$). Let R_s be the subset of points in R whose y-coordinate is greater than that of s. As we assumed s is the lowest point in the optimal solution, we can disregard points in $R \setminus R_s$. Next, we sort all other points in R_s clockwise radially around s (i.e. from the negative x axis clockwise about s to the positive x axis) and process points in this order.

One issue we must deal with first is that in Equation (2.1), $sum_{cone}(v_k, s, v_2)$ depends both on the choice of v_k and v_2. To break this cyclic behavior we cut the cone for s in two. So cast a ray in the negative y-direction from s and call it r_s, and observe that as s is the lowest vertex, r_s must lie in the cone for s. We cut the cone for s along r_s and assign each piece to its adjacent slab. Specifically, suppose we set $v_2 = u$ for some $u \in R_s$. Then define $sum_{start}(s, u)$ as the summed cost for the points in the union of $slab(s, u)$ with the cone lying between the rays r_s and $r_l(s, u)$ (including r_s). Similarly, if we set $v_k = w$, then define $sum_{end}(w, s)$ as the summed cost for the points in the union of $slab(w, s)$ with the cone lying between the rays $r_r(w, s)$ and r_s (excluding r_s). Then we have,

$$\begin{aligned} f(V, P) =& sum_{start}(s, v_2) + \sum_{i=2}^{k-1} (sum_{cone}(v_{i-1}, v_i, v_{i+1}) + sum_{slab}(v_i, v_{i+1})) \\ &+ (sum_{cone}(v_{k-1}, v_k, s) + sum_{end}(v_k, s)). \end{aligned} \tag{2.2}$$

We now argue that the recursive algorithm shown in Algorithm 1, minimizes Equation (2.2) over all $V \subseteq R$, such that $V = \{v_1 = s, v_2, \ldots, v_k\}$ are the ordered vertices of a convex hull with lowest point s. This algorithm makes use of the function $right(u, v, w)$ which returns true if the ordered triple (u, v, w) represents a right turn and returns false otherwise. The following simple helper lemma ensures we do not need to check for a right turn at s (i.e. where we split the problem), as long as we check everywhere else.

Lemma 2.3. *Let $V = \{v_1, v_2, \ldots, v_k\}$ be a sequence of points such that v_1 is the lowest point, and $v_2, \ldots, v_k$ are in clockwise sorted order around v_1. If for all $1 < i \leq k$, (v_{i-1}, v_i, v_{i+1}) is a right turn, where $v_{k+1} = v_1$, then V are the ordered vertices of a convex hull.*

Proof. By definition, V are the ordered vertices of a convex hull if V represents a simple closed convex chain. First, because the vertices in $V = \{v_1, \ldots, v_k\}$ are given in clockwise sorted order around v_1, the closed chain V must be simple (i.e. when rotating a ray from v_1, the edges of the chain always cross it in the same direction). In order for the chain to be a closed convex chain, it must make a right turn at every vertex. We are already explicitly given that a right turn is made at every vertex except for v_1. To see why (v_k, v_1, v_2) is a right turn, observe that v_1 is lower than both v_k and v_2, and moreover v_k comes after v_2 in clockwise order about v_1. These two facts combined imply the angle $\angle v_k v_1 v_2$ is $< \pi$ (i.e. the angle subtended by rotating $v_1 v_2$ clockwise about v_1 to $v_1 v_k$), that is a right turn. □

[3] We can assume all points have distinct y-coordinates, by applying a small random rotation, which does not affect f.

Algorithm 1 Recursive Algorithm.

```
1: function RecAlg(s, k', u, v)
2:     best = ∞
3:     if right(u, v, s) then
4:         best = sum_cone(u, v, s) + sum_end(v, s)
5:     if k' = 1 then
6:         return best
7:     for w ∈ R_s after v in clockwise order do
8:         if right(u, v, w) then
9:             best = min{best, sum_cone(u, v, w) + sum_slab(v, w) + RecAlg(s, k' − 1, v, w)}
10:    return best

11: function Wrapper(R, P, k)
12:    best = ∞
13:    for s ∈ R do
14:        for v ∈ R_s in clockwise order do
15:            best = min{best, sum_start(s, v) + RecAlg(s, k − 1, s, v)}
16:    return best
```

Given the above discussion about breaking the cost function into cones and slabs according to Equation (2.2), the proof of correctness of Algorithm 1 is now straightforward.

Lemma 2.4. *Given an instance P, R, f, k of Problem 1.2 in the plane, Algorithm 1 computes the optimal solution cost, namely* $\min_{Q \subseteq R, |Q| \leq k} f(Q, P)$.

Proof. For any $s \in R$, we now argue $cost_s = \min_{v \in R_s}(sum_{start}(s, v) + \text{RecAlg}(s, k-1, s, v))$ is the cost of the minimum cost k length convex hull with lowest point s. This will imply Wrapper computes the optimum solution as it takes the minimum of this quantity over all $s \in R$.

Suppose that $cost_s$ is not infinite. By the structure of the recursive algorithm, this can only happen if in each recursive call determining $cost_s$, *best* is not infinite. The places where *best* can be set to a non-infinite value are lines 4 and 9, and if the return value of *best* is set by line 4 then this represents a terminal call. Moreover, observe that executing line 4 or 9 requires satisfying a right turn check on the preceding line. Thus, there must have been a sequence of recursive calls made with a corresponding sequence of vertices $V = \{v_1 = s, v_2, \ldots, v_\kappa\}$ such that for all $1 < i \leq \kappa$, $right(v_{i-1}, v_i, v_{i+1}) = true$ (where $v_{\kappa+1} = v_1$), which by Lemma 2.3, implies V are the ordered vertices of a convex hull. (Note that s being the lowest is enforced by considering only R_s, and the clockwise ordering of V is enforced by the ordering of the for loops.) Moreover, we have $cost_s = sum_{start}(s, v_2) + \text{RecAlg}(s, k-1, s, v_2)$, and from line 9 for all $1 < i < \kappa$ we have $RecAlg(s, k-i+1, v_{i-1}, v_i) = sum_{cone}(v_{i-1}, v_i, v_{i+1}) + sum_{slab}(v_i, v_{i+1}) + RecAlg(s, k-i, v_i, v_{i+1})$, and from line 4 we have $RecAlg(s, k-\kappa+1, v_{\kappa-1}, v_\kappa) = sum_{cone}(v_{\kappa-1}, v_\kappa, s) + sum_{end}(v_\kappa, s)$. Thus putting all these equations together we have

$$\begin{aligned} cost_s =& sum_{start}(s, v_2) + \sum_{i=2}^{\kappa-1}(sum_{cone}(v_{i-1}, v_i, v_{i+1}) + sum_{slab}(v_i, v_{i+1})) \\ &+ (sum_{cone}(v_{\kappa-1}, v_\kappa, s) + sum_{end}(v_\kappa, s)) = f(V, P) \end{aligned}$$

where the last equality follows from Equation (2.2). Thus if $cost_s$ is not infinite then we know it represents the true cost of some valid set of convex hull vertices V. Conversely, by a similar logic it is easy to see that $cost_s$ is never infinite since for the ordered sequence of vertices of any convex hull all the right turn checks will be satisfied and in the algorithm when looking for the next vertex we try all possible vertices that remain in the sorted order. (Note that ∞ may be returned if there is no non-trivial convex hull, i.e. if s is the highest vertex in R, a case which can be treated separately.) Thus what remains is to argue that the output cost and vertices selected correspond to a minimal cost solution, however, this is immediate from the above. Specifically, let $\mathcal{V}_i$ be the set of all clockwise ordered convex hull vertices such that all have the same prefix $\{v_1, \ldots, v_i\}$. Then $\min_{V \in \mathcal{V}_i} f(V, P)$ is determined by selecting $\{v_{i+1}, \ldots v_k\}$ so as to minimize the cone and slab sums they determine, which as argued above is precisely what lines 9 and 4 do. In particular, because the cones and slabs define an ordered partition of P, minimizing their cost over the remaining vertices, does not affect the cone and slab cost determined by the previously selected vertices, and thus the recursive algorithm correctly returns the minimal cost overall. □

As the correctness of our approach is established by the above lemma, the proof of the following theorem mainly focuses on the running time. The proof saves roughly an $O(m)$ factor over the naive time bound by using sweeping both to batch dynamic programming table entries together and to implicitly precompute the sum_{cone} values.

Theorem 2.5. *Given an instance P, R, f, k of Problem 1.2 in the plane, it can be solved in $O(m^3k + m^2n + mn\log(n))$ time, where $n = |P|$ and $m = |R|$.*

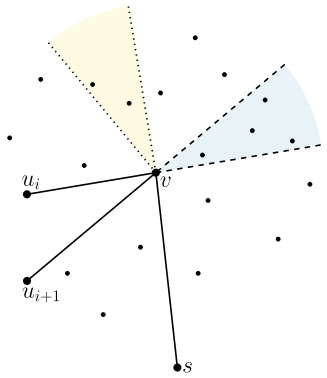

Fig. 2.3. S_{i+1} with 4 points shown shaded blue on the right. The two points determining $x = sum_{cone}(u_{i+1}, v, \cdot) - sum_{cone}(u_i, v, \cdot)$ shown in shaded yellow on the left.

Proof. First, observe that the recursive Algorithm 1 can easily be turned into a dynamic program, as the k' parameter strictly decreases in each recursive call. Moreover, it is easy to modify the code such that it returns the actual vertices instead of just the cost of the hull.

The correctness of this algorithm follows from Lemma 2.4. For the running time, first observe that for every vertex $s \in R$ we can compute R_s and the clockwise sorted order of all points in R around s, in $O(m^2 \log m)$ time. So assume this is done initially, and moreover assume for now that all the cone and slab sums have been precomputed. The dynamic program will compute the value of $\textsc{RecAlg}(s, k', u, v)$ for each quadruple (s, k', u, v). Naively this takes $O(m)$ time per quadruple since the for loop on line 7 requires a table lookup for each point in R. Thus overall the dynamic program takes $O(m^4 k)$ time as the table size is $O(m^3 k)$. However, we can save an $O(m)$ factor in the running time by instead computing for each triple $(s, k', \cdot, v)$, the entire column of u values in $O(m)$ time as follows.

Fix s, k', and v. Define $cost(u, v, w) = sum_{cone}(u, v, w) + sum_{slab}(v, w) + \textsc{RecAlg}(s, k' - 1, v, w)$. For any $u \in R_s$ coming before v in the clockwise order about s, the recursive algorithm computes $\textsc{RecAlg}(s, k', u, v) = \min_{w \in Y(u)} cost(u, v, w)$, where $Y(u)$ is the set of points $w \in R_s$ such that $right(u, v, w) = true$ and moreover w is after v in the clockwise order about s. Specifically, this is all points in the region determined by sweeping the ray from v to s counterclockwise until it hits the line passing through u and v. See Fig. 2.3. So let $u_1, \ldots, u_z$ be the vertices in R_s coming before v in the clockwise order about s, but labeled by their counterclockwise order about v. Then $Y(u_i) \subseteq Y(u_{i+1})$, and in particular $S_{i+1} = Y(u_{i+1}) \setminus Y(u_i)$ are the set of points in the wedge lying between the line through u_i and v and the line through u_{i+1} and v (again see Fig. 2.3). Observe that the S_i are disjoint sets, and moreover,

$$\textsc{RecAlg}(s, k', u_{i+1}, v) = \min_{w \in Y(u_{i+1})} cost(u_{i+1}, v, w) = \min\{\min_{w \in S_{i+1}} cost(u_{i+1}, v, w), \min_{w \in Y(u_i)} cost(u_{i+1}, v, w)\}$$

Observe that $\min_{w \in Y(u_i)} cost(u_{i+1}, v, w) = \min_{w \in Y(u_i)} cost(u_i, v, w) + x$ for a fixed value x that does not depend on w. Namely, $x = sum_{cone}(u_{i+1}, v, \cdot) - sum_{cone}(u_i, v, \cdot)$ (see Fig. 2.3), as the cone sum is the only term in $cost(u, v, w)$ depending on u. Then given we already computed $\textsc{RecAlg}(s, k', u_i, v) = \min_{w \in Y(u_i)} cost(u_i, v, w)$, by the above equation the time to compute $\textsc{RecAlg}(s, k', u_{i+1}, v)$ is proportional to just $|S_i|$. Thus as the S_i are disjoint, this takes $O(m)$ time over all the u_i, resulting in $O(m^3 k)$ time for the entire dynamic program.

Now we must consider the time to precompute the cone and slab sums. For any pair $u, v \in R$, $sum_{slab}(u, v)$ can be computed in $O(n)$ time by scanning the points in P to see which fall in the slab, and thus for all pairs in R the sum slab cost can be computed in $O(m^2 n)$ time. As $sum_{cone}(u, v, w)$ is determined by three vertices in R, similarly computing these values would take $O(m^3 n)$ time. However, we now argue that they can be implicitly computed more efficiently as follows. First, fix any vertex $v \in R$. Recall the boundary of $cone(u, v, w)$ is determined by the rays $r_r(u, v)$ and $r_l(v, w)$ (defined above). So let U and W be the sets of all vertices that come before and after v in the clockwise sorted order about s, respectively, and let $R_r = \cup_{u \in U} r_r(u, v)$ and $R_l = \cup_{w \in W} r_l(v, w)$. Now sort all the vectors in $R_r \cup R_l \cup P$ in clockwise order around v, starting from the first vertex occurring after the negative y-axis direction. Now walk through the vertices in order maintaining a rolling sum, which initially is zero. If the next vertex w is in P then we add $g_v(||v - w||)$ to the sum, otherwise if $w \in R_r$ then we assign the current sum as $value_r(w)$ and if $w \in R_l$ we assign the current sum as $value_l(w)$. Observe that given vertices $u, w \in R$ where u comes before v and w comes after v in clockwise order about s, that $sum_{cone}(u, v, w) = value_l(w) - value_r(u)$. Thus while we do not explicitly compute $sum_{cone}(u, v, w)$ for all triples, by computing all of the $value_l$ and $value_r$ values, then by taking a difference of two such values in constant time we have access to $sum_{cone}(u, v, w)$. This takes $O((n+m)\log(n+m))$ time per vertex in R and thus for all vertices in R takes $O(m(n+m)\log(n+m))$ time. Thus precomputing all the slab sums and implicitly precomputing all the cone sums overall takes $O(m^2 n + m^2 \log(m) + mn \log(n))$ time. Thus the total running time of the entire algorithm is $O(m^3 k + m^2 n + mn \log(n))$. □

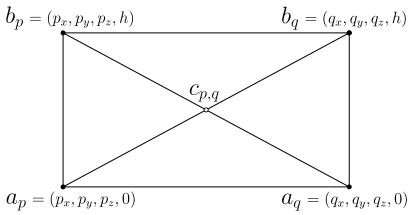

Fig. 3.1. The edge (a_p, a_q) with added points in the added dimension.

3. Hardness in higher dimensions

A convex polytope $T = (V, E)$ in $\mathbb{R}^3$, will be defined as a graph where the vertices V are a set of points in convex position in $\mathbb{R}^3$, and the edges E are the edges of $\mathcal{CH}(V)$. Das and Goodrich [10] proved the following variant of vertex cover is NP-hard.

Problem 3.1 *(Polytope vertex cover).* Given a convex polytope $T = (V, E)$ in $\mathbb{R}^3$ and an integer k, is there a subset $U \subseteq V$ of k vertices such that each edge in E is incident to a vertex in U?

The following is the decision version of our main problem, Problem 1.2.

Problem 3.2. Given a set $P \subset \mathbb{R}^d$ of n points, a set $R \subset \mathbb{R}^d$ of m points, a function $f \in \mathcal{F}_P$, and a parameter ε, is there a subset $Q \subseteq R$ of at most k points such that $f(Q, P) \leq \varepsilon$?

We now show Problem 3.2 is NP-hard for $d \geq 3$, where $f(Q, P) = \sum_{x \in P} g_x(||x - \mathcal{CH}(Q)||)$ is a natural and simple function. Namely, we set $g_x(||x - \mathcal{CH}(Q)||) = ||x - \mathcal{CH}(Q)||$ for all x. We denote this sum of distances function as $\mathsf{sd}(Q, P) = \sum_{x \in P} ||x - \mathcal{CH}(Q)||$.

Theorem 3.3. *Problem 3.2 is NP-hard for $d \geq 3$ and $f = \mathsf{sd}$.*

Proof. We give a polynomial time reduction from Problem 3.1. Let $T = (V, E)$ and k be an instance of Problem 3.1. We first define several quantities based on T. For any edge $e \in E$, define a vector $u_e = (n_1 + n_2)/2$, where n_1 and n_2 are the normals of the planes of the two faces adjacent to e. For any edge $e = (v_1, v_2)$, consider the plane z_e containing e and with normal u_e. Let h_e be the distance from z_e to the convex hull of V after removing the endpoints of e, i.e. $h_e = ||z_e - \mathcal{CH}(V \setminus \{v_1, v_2\})||$, and let $h = \min_{e \in E} h_e$. (Note that h is non-zero as V is in convex position.) Finally, let l_e be the length of the edge e, and let $l = \max_{e \in E} l_e$.

We construct our instance of Problem 3.2 as follows. We use the same value of k and set $R = V$. P will contain one point for each edge $e \in E$, denoted p_e. We place p_e outside $\mathcal{CH}(V)$ at a distance x in the direction of u_e from the midpoint of e, where x is a value to be determined shortly. Finally, we set $\varepsilon = n\sqrt{x^2 + (l/2)^2}$, and recall $f(Q, P) = \mathsf{sd}(Q, P) = \sum_{p \in P} ||p - \mathcal{CH}(Q)||$.

Observe that for any edge $e \in E$, if at least one of its endpoints is selected, then the distance from p_e to the hull of the selected vertices is at most $\sqrt{x^2 + (l_e/2)^2} \leq \sqrt{x^2 + (l/2)^2}$. Thus if $U \subseteq V$ is a vertex cover of V, then $\mathsf{sd}(U, P) \leq n\sqrt{x^2 + (l/2)^2} = \varepsilon$. On the other hand if U is not a vertex cover, then there is an edge e for which neither endpoint is selected, in which case the distance from p_e to the hull of the selected vertices is at least $x + h$. Thus the total distance of all points to the hull is at least $(n-1)x + (x + h) = nx + h$, as by construction for any $e' \in E$ we have $||p_{e'} - \mathcal{CH}(R)|| = x$. Thus if we select x such that $nx + h > \varepsilon$, then U is vertex cover if and only if $\mathsf{sd}(U, P) \leq \varepsilon$. To ensure this inequality holds, set $x = \frac{l^2 n}{8h}$. Then we have

$$\varepsilon = n \cdot \sqrt{x^2 + \frac{l^2}{4}} = n \cdot \sqrt{\left(\frac{l^2 n}{8h}\right)^2 + \frac{l^2}{4}} < n \cdot \sqrt{\left(\frac{l^2 n}{8h} + \frac{h}{n}\right)^2} = n \cdot \frac{l^2 n}{8h} + h = nx + h. \quad \square$$

By lifting to $\mathbb{R}^4$ we can argue that the problem remains NP-hard for the restricted variant where $P = R$, i.e. the case considered in Section 2.1. The proof is more technically challenging, though at a high level uses a similar approach.

Theorem 3.4. *Problem 3.2 is NP-hard for $d \geq 4$, $f = \mathsf{sd}$, and $P = R$.*

Proof. We give a polynomial time reduction from Problem 3.1. Let $T = (V, E)$ and k be an instance of Problem 3.1. For any edge $e \in E$, let ℓ_e denote its length and m_e its midpoint. Define the quantity $h = \min\{\ell, h_1, h_2\}$, where $\ell = \min_{e \in E} \ell_e$,

$h_1 = \min_{p\in V} ||p - \mathcal{CH}(V \setminus \{p\})||$, and $h_2 = \min_{(e_1,e_2)\in E} ||m_{(e_1,e_2)} - \mathcal{CH}(V \setminus \{e_1, e_2\})||$. Then for each point $p = (p_x, p_y, p_z) \in V$, define the points $a_p = (p_x, p_y, p_z, 0)$ and $b_p = (p_x, p_y, p_z, h)$, and for each edge $(p, q) \in E$ define the point $c_{p,q} = (\frac{p_x+q_x}{2}, \frac{p_y+q_y}{2}, \frac{p_z+q_z}{2}, \frac{h}{2})$. See Fig. 3.1. Define the sets $A = \{a_p \mid p \in V\}$, $B = \{b_p \mid p \in V\}$, and $C = \{c_{p,q} \mid (p, q) \in E\}$. Intuitively, we wish to give unit weight to all points in B, and give n^2 weight to all points in A and C. To accomplish this, let A' and C' be the multi-sets consisting of n^2 copies of all points in A and C, respectively. Our instance of Problem 3.2 in $\mathbb{R}^4$ is defined by $P = R = A' \cup B \cup C'$, $k_0 = k + n$, and $\varepsilon = nh$. Note that any solution to Problem 3.2 containing a point from A' (or C'), does not change in cost if we add one of its duplicates or exchange it for a duplicate. Thus we can assume the optimal solution does not select duplicates, and so below we write $A \subset P$ and refer to selecting points from A.

Let $W \subseteq V$ be a vertex cover of size k for the given instance of Problem 3.1, and let $B(W) = \{b = (p, h) \in B \mid p \in W\}$. The claim is that $B(W) \cup A$ is a solution to our instance of Problem 3.2 with cost $\leq \varepsilon$. First, observe that $|B(W) \cup A| = k + n = k_0$ as required. Next, observe that naturally this gives zero error to all points in A' and $B(W)$. The same is true for any point $c_{p,q} \in C'$. To see this, observe that since W is a vertex cover, it must contain at least one of p or q. Without loss of generality, suppose it contains q, in which case $b_q \in B(W)$. Thus $B(W) \cup A$ contains both b_q and a_p, and since $c_{p,q}$ is defined as the midpoint on the segment between b_q and a_p, it is in their convex hull, i.e. it is covered with zero error. Thus the error can only come from points in $B \setminus B(W)$, however, the error for these points is easily upper bounded by $\varepsilon = nh$, as $|B \setminus B(W)| \leq n$ and since for any point $b_p \in B$ we have $||b_p - a_p|| = h$ and a_p is in our solution.

Now let Q be a solution to Problem 3.2 with k_0 points and error $\leq \varepsilon$. We first argue that $A \subseteq Q$. Suppose otherwise that some point $a_0 \in A$ is not in Q. We now lower bound $||a_0 - \mathcal{CH}(Q)||$. Specifically, we will assume $Q = P \setminus \{a_0\}$, as this minimizes $||a_0 - \mathcal{CH}(Q)||$ over all possible Q. Observe, that $c_{p,q} \in \mathcal{CH}(Q)$ for any point $c_{p,q} \in C$, since $b_p, b_q \in Q$, and at least one of a_p or a_q is in Q. Thus every point in $\mathcal{CH}(Q)$ either lies in $\mathcal{CH}(A \setminus \{a_0\})$, in $\mathcal{CH}(B)$, or on a segment between a point of $\mathcal{CH}(A \setminus \{a_0\})$ and $\mathcal{CH}(B)$. Let α, β be the closest points to a_0 in $\mathcal{CH}(A \setminus \{a_0\})$ and $\mathcal{CH}(B)$, respectively. Then by the definition of h, $||a_0 - \alpha|| \geq h$, and $||a_0 - \mathcal{CH}(B)|| = h$. Moreover, it is not hard to see that among all segments between a point of $\mathcal{CH}(A \setminus \{a_0\})$ and $\mathcal{CH}(B)$, the closest segment to a_0 is the segment between α and β. Thus the distance from a_0 to $\mathcal{CH}(Q)$ is at least $(\sqrt{h^2 + h^2})/2 = h/\sqrt{2}$. Since A' contains n^2 copies of a_0, the error of Q for Problem 3.2 is at least $n^2 \frac{h}{\sqrt{2}} > \varepsilon$ (for $n \geq 2$). Thus all points in A must have been selected.

Observe that for any point $c_{p,q} \in C$, that $c_{p,q} \in \mathcal{CH}(a_p, a_q, b_p)$ and $c_{p,q} \in \mathcal{CH}(a_p, a_q, b_q)$, see Fig. 3.1. That is, since $A \subseteq Q$, if a point $c_{p,q}$ is in Q, exchanging it for either b_p or b_q can only enlarge $\mathcal{CH}(Q)$. Thus without loss of generality we assume Q contains no points from C. Moreover, for any $c_{p,q} \in C$, at least one of b_p or b_q is in Q, since otherwise $c_{p,q} \notin \mathcal{CH}(Q)$, in which case by the same argument as above for a_0, we have $||c_{p,q} - \mathcal{CH}(Q)|| \geq (\sqrt{(h/2)^2 + (h/2)^2})/2 = h/\sqrt{8}$. Since C' contains n^2 copies of $c_{p,q}$, the total error is then at least $n^2 \frac{h}{\sqrt{8}} > \varepsilon$ (for $n \geq 3$), a contradiction. Thus for every point $c_{p,q}$ at least one of b_p or b_q is in Q, or equivalently $W = \{p \mid b_p \in Q\}$ is a vertex cover of E. Moreover, it must be that $|W| = k$, as $k_0 = n + k$ and all n points of A were selected. Thus all that remains is to argue that the error due to $B \setminus W$ is less than ε (as all other points are in $\mathcal{CH}(Q)$). However, since all points in A are in Q, this error is as most $(n - k)h < nh = \varepsilon$. □

4. Approximation in higher constant dimensions

Given the hardness of our problem when $d \geq 3$, it is natural to consider approximations. For the Set Cover problem, it is well known that if k sets cover the ground set, then the greedy algorithm covers the ground set with $O(k \log n)$ sets. Our hull problem is also a coverage problem, though it is more challenging as the points in P are not covered by the individual points we select but rather convex combinations of them. Despite this, we argue a similar greedy approach works, though it depends on the number of facets of the convex hull of the optimal k point solution. In $\mathbb{R}^3$ the number of facets is $O(k)$, yielding a $(1 + \varepsilon)$ approximation to the error with only $O(k \log(n/\varepsilon))$ points, similar to Set Cover. In higher constant dimensions, however, the worst case facet complexity is $O(k^{\lfloor d/2 \rfloor})$. On real world inputs the complexity may be significantly lower (see [15] for the facet complexity of randomly sampled points), thus our analysis suggests that greedily selecting roughly a logarithmic factor more points may be a reasonable heuristic in practice for small constant dimensions.

In this section we assume P and R are contained in the unit ball, which as remarked in the introduction is equivalent to measuring the error relative to the diameter, as is standard.

Previously we considered the sum of distances function $\mathsf{sd}(Q, P) = \sum_{x\in P} ||x - \mathcal{CH}(Q)||$. Similarly, we can define the maximum distance function $\mathsf{md}(Q, P) = \max_{x\in P} ||x - \mathcal{CH}(Q)||$. We have the following corresponding optimization problem, considered by Blum *et al.* [4].

Problem 4.1. Given a set $P \subset \mathbb{R}^d$ of n points and a set $R \subset \mathbb{R}^d$ of m points, select a subset $Q \subseteq R$ of at most k points which minimizes $\mathsf{md}(Q, P)$. That is, $Q = \arg\min_{Q\subseteq R, |Q|\leq k} \mathsf{md}(Q, P)$.

For an instance $P, R \subset \mathbb{R}^d$ and k of Problem 4.1, define

$$\mathsf{opt}_{\mathsf{md}} := \mathsf{opt}_{\mathsf{md}}(P, R, k) = \arg\min_{Q\subseteq R, |Q|\leq k} \mathsf{md}(Q, P), \qquad \text{and} \qquad \overline{\mathsf{opt}_{\mathsf{md}}} := \mathsf{md}(\mathsf{opt}_{\mathsf{md}}, P).$$

Similarly define

$$\mathsf{opt}_{\mathsf{sd}} := \mathsf{opt}_{\mathsf{sd}}(P, R, k) = \arg \min_{Q \subseteq R, |Q| \leq k} \mathsf{sd}(Q, P), \qquad \text{and} \qquad \overline{\mathsf{opt}_{\mathsf{sd}}} := \mathsf{sd}(\mathsf{opt}_{\mathsf{sd}}, P).$$

Lemma 4.2 *([4]). Let $P, R \subset \mathbb{R}^d$ and k be an instance of Problem 4.1, where d is a constant. Then in polynomial time one can compute a set Q_0 of $O(k \log k)$ points such that $\mathsf{md}(Q_0, P) \leq \overline{\mathsf{opt}_{\mathsf{md}}}(P, R, k)$.*

Let Q_0 be the set described in the above lemma. Observe that

$$\begin{aligned}\frac{\mathsf{sd}(Q_0, P)}{n} &\leq \mathsf{md}(Q_0, P) \leq \overline{\mathsf{opt}_{\mathsf{md}}} = \max_{p \in P} ||p - \mathcal{CH}(\mathsf{opt}_{\mathsf{md}})|| \\ &\leq \max_{p \in P} ||p - \mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})|| \leq \sum_{p \in P} ||p - \mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})|| = \overline{\mathsf{opt}_{\mathsf{sd}}},\end{aligned}$$

that is Q_0 achieves an n-approximation to the optimal sum cost $\overline{\mathsf{opt}_{\mathsf{sd}}}$.

For any subsets $Q \subseteq R$ and $P' \subseteq P$, let $Z(Q, P') = \mathsf{sd}(Q, P') - \mathsf{sd}(\mathsf{opt}_{\mathsf{sd}}, P') = \mathsf{sd}(Q, P') - \overline{\mathsf{opt}_{\mathsf{sd}}}$. For convenience $Z(Q)$ will denote $Z(Q, P)$ when P is the full point set.

Lemma 4.3. *Given an instance $P, R \subset \mathbb{R}^d$ and k of Problem 1.2, where $f = \mathsf{sd}$ and d is a constant, for any subset $Q \subseteq R$ such that $Z(Q) \geq 0$, there exists a d-simplex Δ such that $Z(Q \cup \Delta) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right) Z(Q)$, where c is a constant.*

Proof. Let $\{\Delta_1, \Delta_2, \ldots, \Delta_\ell\}$ be the d-simplices of the d-dimensional triangulation of $\mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})$ with the minimum number of d-simplices. It is known that $\ell \leq c \cdot k^{\lfloor d/2 \rfloor}$, where c is a constant (using for example the bottom vertex triangulation of [7]). Let $\{P_1, P_2, \ldots, P_\ell\}$ be the partition of P where $p \in P_i$ if and only if $||p - \Delta_i|| = ||p - \mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})||$. (If the projection is on a common point of more than one simplex, assign one arbitrarily.) Now, rewrite $Z(Q)$ as

$$Z(Q) = \sum_{i=1}^{\ell} \sum_{x \in P_i} \left(||x - \mathcal{CH}(Q)|| - ||x - \mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})|| \right).$$

Let $\mathsf{Avg} := \frac{Z(Q)}{\ell}$ denote the average of $Z(Q)$ over the partitions P_i. Hence, there exists a simplex Δ_j with corresponding partition P_j such that

$$Z(Q, P_j) = \sum_{x \in P_j} \left(||x - \mathcal{CH}(Q)|| - ||x - \mathcal{CH}(\mathsf{opt}_{\mathsf{sd}})|| \right) \geq \mathsf{Avg} \geq \frac{Z(Q)}{c \cdot k^{\lfloor d/2 \rfloor}},$$

where the last inequality is where we used $Z(Q) \geq 0$. Finally, we have $Z(Q \cup \Delta_j, P_j) = \mathsf{sd}(Q \cup \Delta_j, P_j) - \mathsf{sd}(\mathsf{opt}_{\mathsf{sd}}, P_j) = \mathsf{sd}(Q \cup \Delta_j, P_j) - \mathsf{sd}(\Delta_j, P_j) \leq 0$. Thus,

$$\begin{aligned}Z(Q \cup \Delta_j) &= \left(\sum_{i \in [\ell], i \neq j} Z(Q \cup \Delta_j, P_i) \right) + Z(Q \cup \Delta_j, P_j) \leq \sum_{i \in [\ell], i \neq j} Z(Q, P_i) \\ &= \left(\sum_{i \in [\ell]} Z(Q, P_i) \right) - Z(Q, P_j) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right) Z(Q) \quad \square\end{aligned}$$

We remark that the running time of Lemma 4.2 from [4] depends exponentially on d, and thus the same is true for the following theorem which makes use of it.

Theorem 4.4. *Given an instance $P, R \subset \mathbb{R}^d$ and k of Problem 1.2, where $f = \mathsf{sd}$ and d is a constant, in polynomial time one can compute a set $Q \subseteq R$ of $O(k^{\lfloor d/2 \rfloor} \log(n/\varepsilon))$ points such that $\mathsf{sd}(Q, P) \leq (1 + \varepsilon) \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}(P, R, k)$.*

Proof. Use Lemma 4.2 to compute a set $Q_0 \subseteq R$ of $O(k \log k)$ points such that $\mathsf{sd}(Q_0, P) \leq n \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}(P, R, k)$. We will iteratively add subsets of $d + 1$ points to Q_i for $i = \{0, 1, \ldots, m - 1\}$ where m is the total number of iterations. Let $A_i := \arg\min_{\Delta \subseteq R, |\Delta| = d+1} \mathsf{sd}(Q_i \cup \Delta, P)$ that is, A_i is the d-simplex whose addition to the current hull minimizes the sum of distances. In the i-th iteration we add A_i to Q_i to obtain $Q_{i+1} := Q_i \cup A_i$.

Recall that $Z(Q_m) = \mathsf{sd}(Q_m, P) - \overline{\mathsf{opt}_{\mathsf{sd}}}$. Thus if $Z(Q_m) \leq \varepsilon \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}$ then $\mathsf{sd}(Q_m, P) \leq (1 + \varepsilon)\overline{\mathsf{opt}_{\mathsf{sd}}}$ as desired. If at any iteration $Z(Q_i) \leq 0$, then $Z(Q_m) \leq 0 \leq \varepsilon \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}$, since adding more points in later iterations can only further decrease the error. So assume that $Z(Q_i) > 0$, then by Lemma 4.3, there exists a simplex Δ such that $Z(Q_i \cup \Delta) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right) Z(Q_i)$.

Note that since $Z(Q_i \cup \Delta) = \mathsf{sd}(Q_i \cup \Delta, P) - \overline{\mathsf{opt}_{\mathsf{sd}}}$, we have $Z(Q_{i+1}) = Z(Q_i \cup A_i) \leq Z(Q_i \cup \Delta)$ since we chose A_i to minimize $\mathsf{sd}(Q_i \cup A_i, P)$ and $\overline{\mathsf{opt}_{\mathsf{sd}}}$ is fixed. Thus we have $Z(Q_{i+1}) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right) Z(Q_i)$, and inductively

$$Z(Q_m) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right)^m Z(Q_0) \leq \left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right)^m n \cdot \overline{\mathsf{opt}_{\mathsf{sd}}},$$

where the second inequality follows as $\mathsf{sd}(Q_0, P) \leq n \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}$. Thus if we select m such that $\left(1 - \frac{1}{c \cdot k^{\lfloor d/2 \rfloor}}\right)^m \leq (\varepsilon/n)$, then $Z(Q_m) \leq \varepsilon \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}$ as desired. Note that $(1 - \frac{1}{ck^{\lfloor d/2 \rfloor}})^m \leq exp(-m/ck^{\lfloor d/2 \rfloor})$ and rearranging the equation $exp(-m/ck^{\lfloor d/2 \rfloor}) = \varepsilon/n$ gives $m = ck^{\lfloor d/2 \rfloor} \log(n/\varepsilon)$. As we are adding $d+1$ points in each round, and d is a constant, we thus get $O(k^{\lfloor d/2 \rfloor} \log(n/\varepsilon))$ points in total. □

Corollary 4.5. *Given an instance $P, R \subset \mathbb{R}^3$ and k of Problem 1.2, where $f = \mathsf{sd}$, in polynomial time one can compute a set $Q \subseteq R$ of $O(k \log(n/\varepsilon))$ points such that $\mathsf{sd}(Q, P) \leq (1+\varepsilon) \cdot \overline{\mathsf{opt}_{\mathsf{sd}}}(P, R, k)$.*

Conclusion

This paper initiated the rigorous study of the convex hull coverage problem. In the plane, we gave polynomial time exact algorithms and then we complemented these results by giving hardness results for higher dimensions. Further, we gave a bicriteria approximation algorithm for any fixed dimension $d \geq 3$. Possible future work includes reducing the running time of the planar algorithms, proving the special case when $P = R$ is still NP-hard when $d = 3$, and improving the approximation quality in higher dimensions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, C. Yap, Finding minimal convex nested polygons, Inf. Comput. 83 (1) (1989) 98–110.
[2] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, in: ACM-SIAM Symposium on Discrete Algorithms (SODA), 2007, pp. 1027–1035.
[3] S. Barman, Approximating Nash equilibria and dense subgraphs via an approximate version of Carathéodory's theorem, SIAM J. Comput. 47 (3) (2018) 960–981.
[4] A. Blum, S. Har-Peled, B. Raichel, Sparse approximation via generating point sets, ACM Trans. Algorithms 15 (3) (2019) 32.
[5] C. Boutsidis, D. Woodruff, Optimal CUR matrix decompositions, SIAM J. Comput. 46 (2) (2017) 543–589.
[6] E. Bronstein, Approximation of convex sets by polytopes, J. Math. Sci. 153 (6) (2008) 727–762.
[7] K. Clarkson, A randomized algorithm for closest-point queries, SIAM J. Comput. 17 (4) (1988) 830–847.
[8] K. Clarkson, Algorithms for polytope covering and approximation, in: Workshop on Algorithms and Data Structures (WADS), in: Lecture Notes in Computer Science, vol. 709, Springer, 1993, pp. 246–252.
[9] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, 3rd edition, The MIT Press, 2009.
[10] G. Das, M. Goodrich, On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees, Comput. Geom. 8 (1997) 123–137.
[11] S. Har-peled, Geometric Approximation Algorithms, American Mathematical Society, 2011.
[12] J. Hershberger, J. Snoeyink, An O(n log n) implementation of the Douglas-Peucker algorithm for line simplification, in: Symposium on Computational Geometry (SOCG), ACM, 1994, pp. 383–384.
[13] A. Kumar, V. Sindhwani, P. Kambadur, Fast conical hull algorithms for near-separable non-negative matrix factorization, in: Int. Conf. on Machine Learning, vol. 28, 2013, pp. 231–239.
[14] M. Mahajan, P. Nimbhorkar, K. Varadarajan, The planar k-means problem is np-hard, Theor. Comput. Sci. 442 (2012) 13–21.
[15] M. Reitzner, The combinatorial structure of random polytopes, Adv. Math. 191 (1) (2005) 178–208.
[16] C. Tóth, J. O'Rourke, J. Goodman, Handbook of Discrete and Computational Geometry, Discrete Mathematics and Its Applications, CRC Press, 2017.
[17] G. Van Buskirk, B. Raichel, N. Ruozzi, Sparse approximate conic hulls, in: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 2534–2544.
[18] M. van Kreveld, M. Löffler, L. Wiratma, On optimal polyline simplification using the Hausdorff and Fréchet distance, J. Comput. Geom. 11 (1) (2020) 1–25.
[19] S. Vavasis, On the complexity of nonnegative matrix factorization, SIAM J. Optim. 20 (3) (2010) 1364–1377.