Convex Hull Complexity of Uncertain Points

Hongyao Huang* Benjamin Raichel*

Abstract

An uncertain point set U is a collection of compact regions in the plane, and a realization of U is any point set determined by selecting one point from each set in U. Here we consider the problem of determining the realization whose convex hull has the minimum number of vertices possible. We prove that when U is a set of n parallel line segments then the problem can be solved in $O(n^3)$ time, but when the line segments can have arbitrary orientations then the problem is NP-Complete.

1 Introduction

Uncertainty in computational problems has received significant attention in recent years, as many real world inputs are inherently noisy. Such problems have been particularly well studied within computational geometry, as uncertainty naturally arises when for example collecting locational data from the physical world.

In this paper we consider the complexity of the convex hull, one of the most fundamental geometric structures, in the context of uncertainty. Specifically, given an uncertain point set $U = \{u_1, \ldots, u_n\}$, where each u_i is a compact region in the plane, a realization of U is any point set $P = \{p_1, \ldots, p_n\}$ such that $p_i \in u_i$ for all i. Here we consider finding the realization of U whose convex hull has the minimum number of vertices.

To the best of our knowledge, our paper is the first to consider the minimum complexity of the convex hull in such uncertain settings when measured by the number of vertices. Previous papers have considered the problem when complexity is measured by perimeter or area. Rappaport [20] computed the minimum perimeter convex hull for line segments with a constant number of orientations in near linear time. Mukhopadhyay *et al.* [19] computed the minimum area convex hull for parallel lines in near linear time. Subsequently, Löffler and van Kreveld [18] did an extensive study on finding the realization which either minimized or maximized the area or perimeter of the convex hull, where different algorithmic or hardness results were given depending on the shape of the uncertain regions.

Various other geometric structures have also been considered in uncertain settings, such as bounding boxes [17], Delaunay triangulations [3, 16], Voronoi diagrams [5, 7, 12, 15], terrains [6, 9], and more. When the uncertain points have associated probabilities, the expected complexity of the convex hull was previously studied (see [11] and references therein). Related questions concerning the convex hull have also been considered, such as computing the probability a given query point is contained in the convex hull [1], or computing the most likely convex hull of probabilistic points [21]. More generally, Jørgensen *et al.* [13] considered the distributions of various geometric quantities in probabilistic settings.

Our problem also relates to the traversal problem, where given a set of convex regions in the plane, one seeks a polygonal chain with some property which stabs all the regions. When the regions are disjoint and ordered, Guibas *et al.* [10] gave efficient algorithms to compute the minimum link polygonal chain stabbing the objects in order. When the objects are parallel line segments, Goodrich and Snoeyink [8] gave a near linear time algorithm to compute a convex stabber if it exists, that is a selection of a single point from each segment such that the resulting set is in convex position. For line segments with general orientations, Arkin *et al.* [2] proved that determining the existence of such a convex stabber is NP-hard. More recently, for the case when the line segments have a constant number of orientations, Keikha *et al.* [14] gave a polynomial time algorithm to determine if there is a convex stabber which stabs at least k segments.

Our Contribution. Our main result is an $O(n^3)$ time algorithm to compute the realization whose convex hull has the fewest vertices when the uncertain regions are parallel line segments. Without loss of generality, for this case we can assume the segments are all vertical. The behavior of our minimization problem differs from the previously studied minimization problems for perimeter or area [19, 20], and instead behaves more similarly to the problem of maximizing the area, for which Löffler and van Kreveld [18] gave an $O(n^3)$ time algorithm. There the authors argued one can assume each segment is realized either at its top or bottom endpoint. This is no longer true for our problem, however, we can argue that other than the leftmost and rightmost segment, one can assume all segments defining vertices of the convex hull are realized either at their top or bottom endpoint. The differences between these two state-

*Department of Computer Science, University of Texas at Dallas, {hhuang, benjamin.raichel}@utdallas.edu. Partially supported by a NSF CAREER Award 1750780.

ments, makes achieving the same $O(n^3)$ running time for our problem more challenging, particularly when it comes to determining the leftmost and rightmost points. Related challenges arise in the problem of maximizing the number of stabbed segments in a convex traversal, considered by Keikha *et al.* [14], for which the authors give an $O(n^6)$ time algorithm. The $O(n^3)$ running time of our algorithm thus compares favorably, though such a comparison is limited as the problems differ.

We complement our algorithmic result for parallel line segments, by proving that the problem is NP-Complete when the line segments can have arbitrary orientations. Our reduction is inspired by the NP-hardness proof in [18]. However, as our problem is a minimization problem and theirs a maximization problem, additional points must be added to keep the gadgets from collapsing inwards to a trivial solution.

1.1 Preliminaries

We follow the uncertain point model of previous papers such as [18], where an uncertain point is modeled by an uncertain region u, which is any compact subset of the plane. For a set of uncertain regions $U = \{u_1, u_2, ..., u_n\}$, a *realization* of U is any point set $P = \{p_1, \ldots, p_n\}$ such that $p_i \in u_i$ for all i. Let $Real(U)$ denote the set of all possible realizations of U.

Given a point set P, let $CH(P)$ denote the convex hull of P, and let $|CH(P)|$ denote the number of vertices of the convex hull, where a vertex of $CH(P)$ is any point $q \in P$ such that $q \notin CH(P \setminus \{q\})$.

Problem 1 *Given a set $U = \{u_1, u_2, ..., u_n\}$ of uncertain regions, compute* $\arg\min_{P \in Real(U)} |CH(P)|$.

Throughout we will use the following basic polygonal chain definitions.

Definition 2 *A* polygonal chain *is an ordered sequence of points in the plane $P = \{p_1, \ldots, p_n\}$. P is* monotone *(resp. reverse monotone) if for all $1 \leq i < n$, p_{i+1} has larger (resp. smaller) x-coordinate than p_i. P is* convex *if for all $1 < i < n$, p_{i-1}, p_i, p_{i+1} defines a right turn, that is p_{i+1} lies to the right of the line segment $\overline{p_{i-1}p_i}$ when directed from p_{i-1} to p_i. If P is convex and monotone (resp. reverse monotone) then it is called a* top chain *(resp.* bottom chain*). P is* simple *if for all $i < j$, $\overline{p_ip_{i+1}}$ and $\overline{p_jp_{j+1}}$ do not intersect, except at p_{i+1} when $j = i + 1$. (Bottom and top chains are simple.)*

Let Q be a point set, and let p_l and p_r respectively be the leftmost and rightmost points in Q. $CH(Q)$ is described by a simple closed convex polygonal chain of its vertices, which is composed of a top chain from p_l to p_r followed by a bottom chain from p_r to p_l.

For a point p in the plane, let $p.x$ and $p.y$ denote its x and y coordinate, respectively. Let $P = \{p_1, \ldots, p_n\}$ be a monotone polygonal chain, and let q be any point in the plane such that $p_1.x \leq q.x \leq p_n.x$. Let i be the index such that the vertical line through q intersects the segment $\overline{p_ip_{i+1}}$. Then we say q lies *below* (resp. *above*) the monotone chain P if it lies below (resp. above) or on the segment $\overline{p_ip_{i+1}}$.

2 Vertical Line Segments

In this section, we give a polynomial time algorithm for Problem 1, when U is a set of vertical line segments S. More generally, the algorithm works for any set of parallel line segments, as rotation does not change $|CH(P)|$. Throughout we let $S = \{s_1, \ldots, s_n\}$ denote a set of vertical line segments, where for simplicity we assume no two segments lie on the same vertical line, and the segments are ordered such that for $i < j$ segment s_i lies to the left of s_j. For a segment s_i, we use s_i^+ to denote its top endpoint, and s_i^- to denote its bottom endpoint.

Definition 3 *Call a monotone polygonal chain $P = \{p_1, \ldots, p_m\}$ a* positive chain *with respect to S if, $p_1 \in s_1$, $p_m \in s_n$, and for all $1 < i < m$, $p_i = s_j^+$ for some j. Similarly, define negative chains.*

Lemma 4 *When U is a set of vertical line segments S, there is an optimal solution to Problem 1, such that the top chain of the convex hull is a positive chain and the bottom chain is a negative chain.*

Proof. Consider any set $R \in Real(S)$, and let $T = \{t_1, \ldots, t_m\}$ be the top chain of $CH(R)$. Let $t_i \in T$ be any vertex of the top chain other than t_1 and t_m, and let t_i^+ be the upper endpoint of the segment which generated t_i. Let $H_{old} = CH(R)$ and $H_{new} = CH((R \setminus t_i) \cup t_i^+)$. We now argue that $|H_{new}| \leq |H_{old}|$. This will prove the lemma, as one can then iteratively move each vertex remaining on the top chain to its upper segment endpoint until all top chain vertices are at their upper segment endpoints, without ever increasing the number of top chain vertices. A symmetric argument applies to the bottom chain.

Observe that H_{old} and H_{new} are convex hulls of the same set of points except where t_i has been exchanged for t_i^+. This implies that if we can argue that $H_{old} \subseteq H_{new}$ then any vertex of H_{new} (other than t_i^+) must be a vertex of H_{old} and thus $|H_{new}| \leq |H_{old}|$ as desired. Note that $CH(R \setminus t_i) \subseteq H_{new}$, thus to argue $H_{old} \subseteq H_{new}$, it suffices to argue $t_i \in H_{new}$. To this end, note that t_{i-1} and t_{i+1} exist as $1 < i < m$. So let z be the point on $\overline{t_{i-1}t_{i+1}}$ lying directly below t_i, i.e. with the same x coordinate, and note that z is well defined as t_{i-1}, t_i, and t_{i+1} are consecutive on the upper chain T (which is convex monotone). Moreover, $z \in H_{new}$ as H_{new} contains both t_{i-1} and t_{i+1}. As t_i^+ lies directly above t_i and z, we have that $t_i \in \overline{z\, t_i^+} \subseteq H_{new}$, proving the lemma. □

The above lemma suggests a natural dynamic programming strategy. Process the segments in S from left to right, where at each segment if we decide it corresponds to a hull vertex then we take its top or bottom endpoint. If it does not correspond to a hull vertex, then by mainitaining appropriate information about the previously selected hull vertices, we will enforce that the segment intersects the final hull, implying its realization can be inside the hull.

Intuitively the structure we wish to maintain is the top and bottom chains of the optimal convex hull. First, observe that these chains cannot be computed independently as selecting a vertex for the top chain affects whether it can or needs to be selected for the bottom chain. Thus as we go from left to right we will remember the last vertex selected from both the top and the bottom chains. Enforcing convexity, however, would require remembering the previous edge not the previous vertex, which would be more expensive. Thus instead we look for positive and negative chains with the fewest vertices, which may not be convex but must satisfy certain properties implied by convexity, and then we use the lemma below to argue these properties are sufficient. (Ultimately one can argue minimal such chains are in fact top and bottom chains, though it is not necessary.)

Definition 5 *Call a pair P^+, P^- of positive and negative chains, a* valid chain pair *if the first and last vertices of P^+ are the same as those of P^-, and for all $1 < i < n$ (i) if $s_i^+ \in P^+$ then $s_i^- \notin P^-$ and if $s_i^- \in P^-$ then $s_i^+ \notin P^+$, (ii) s_i^- lies below P^+, and (iii) s_i^+ lies above P^-.*

The proof of the following is in Appendix A.2.

Lemma 6 *Let P^+, P^- be a valid chain pair. Then $\mathcal{CH}(P^+ \cup P^-)$ intersects all segments in S.*

By Lemma 4 we know that there is an optimal solution to Problem 1 such that the top chain of the convex hull is a positive chain $\hat{P}^+$ and the bottom chain is a negative chain $\hat{P}^-$. Note that all points in this optimal realization of S lie below the top chain and above the bottom chain of the convex hull. This implies that for all $1 < i < n$, s_i^- lies below $\hat{P}^+$ and s_i^+ lies above $\hat{P}^-$, and therefore $\hat{P}^+, \hat{P}^-$ is a valid chain pair. So let P^+, P^- be a valid chain pair minimizing $|P^+|+|P^-|$. By Lemma 6 $\mathcal{CH}(P^+ \cup P^-)$ intersects all segments in S, and thus there is a realization of S whose convex hull vertices all lie in $P^+ \cup P^-$. As clearly $|P^+| + |P^-| \leq |\hat{P}^+| + |\hat{P}^-|$, we have the following.

Corollary 7 *Let P^+, P^- be a valid chain pair which minimizes $|P^+| + |P^-|$ over all valid chain pairs. Then $P^+ \cup P^-$ are the vertices of an optimal solution to Problem 1 on S.*

By the above it thus suffices to compute the minimum sized valid chain pair, which can easily be accomplished using a standard dynamic programming approach. Specifically, the recursive Algorithm 1 maintains the previous vertex selected on the positive chain, s_i^+, and the previous vertex selected on the negative chain, s_j^-, and then tries all possible choices for the next vertex to the right (of both s_i and s_j), which if on segment s_k could be either s_k^+ or s_k^-. Specifically, in order for s_k^+ to be considered as a possible next vertex on the positive chain, by Definition 5, we must require that for all $i < x < k$ that s_x^- lies below the segment $\overline{s_i^+ s_k^+}$. Assume we have a function $\textsc{Positive}(i)$ that computes all such indices. For k to be a valid next index we also require $k > \max\{i, j\}$. Thus if P denotes the set of all possible next positive chain vertex indices, then $P = \textsc{Positive}(i) \cap \{\max\{i, j\} < k < n\}$. Similarly define the set of possible next negative chain vertices $N = \textsc{Negative}(j) \cap \{\max\{i, j\} < k < n\}$, where $\textsc{Negative}(j)$ is defined analogously to $\textsc{Positive}(i)$. Finally, define $\textsc{EndRight}(i, j)$ as the function which returns true if we can extend the current chains directly to the rightmost segment s_n, namely does there exist a point $r \in s_n$ such that s_x^- lies below the segment $\overline{s_i^+ r}$ for $i < x < n$ and s_x^+ lies above the segment $\overline{s_j^- r}$ for $j < x < n$.

Algorithm 1 Recursive Algorithm for Problem 1

Output: Min number of remaining valid chain pair vertices or ∞ if no solution, given the previous positive and negative chain vertices were s_i^+ and s_j^-.

```
1:  function MinCH(i, j)
2:      P ← Positive(i) ∩ {max{i, j} < k < n}
3:      N ← Negative(j) ∩ {max{i, j} < k < n}
4:      value ← ∞
5:      for k ∈ P do
6:          value ← min{value, 1 + MinCH(k, j)}
7:      for k ∈ N do
8:          value ← min{value, 1 + MinCH(i, k)}
9:      if EndRight(i, j) then
10:         value = 1
11:     return value
```

First we argue that when the leftmost segment s_1 is a single point (or equivalently we know the point to select on segment s_1), then Algorithm 1 can be used to to solve Problem 1 in cubic time. Afterwards, we argue how to remove this assumption on s_1 while maintaining the same running time.

Theorem 8 *For a set $S = \{s_1, \ldots, s_n\}$ of vertical segments, where the leftmost segment s_1 is a single point, Problem 1 can be solved in $O(n^3)$ time.*

Proof. Assuming that $\textsc{Positive}(i)$, $\textsc{Negative}(j)$, and $\textsc{EndRight}(i, j)$ all work as described

above then Algorithm 1 sets $\textsc{MinCH}(i,j) = 1 + \min\{\min_{k\in P} \textsc{MinCH}(k,j), \min_{k\in N} \textsc{MinCH}(i,k)\}$ or 1 if we can connect directly to the rightmost segment, where P and N are respectively the sets of all possible next positive and negative chain vertices. Thus $\textsc{MinCH}(1,1)+1$ computes the size of a minimum cardinality valid chain pair, which by Corollary 7 corresponds to an optimal solution to Problem 1. Note because s_1 is a single point, $s_1 = s_1^+ = s_1^-$, thus all subroutine calls are well defined, and $\textsc{MinCH}(1,1)$ will start the valid chain pairs on the same point as required, where this point is counted by the $+1$.

As i and j both range over $O(n)$ possible values, this recursive algorithm can be turned into a dynamic program with a table of total size $O(n^2)$. Assuming $\textsc{Positive}(i)$, $\textsc{Negative}(j)$, and $\textsc{EndRight}(i,j)$ all run in $O(n)$ time, then each table entry takes $O(n)$ time to compute as outside those subroutines the code consists of constant time operations and two disjoint for loops going over P and N. This then gives an $O(n^3)$ running time overall as claimed. Thus what remains is to describe how to implement $\textsc{Positive}(i)$, $\textsc{Negative}(j)$, and $\textsc{EndRight}(i,j)$ in linear time.

First, we describe how to compute $P' = \textsc{Positive}(i)$ in linear time, from which one can then easily compute $P = P' \cap \{\max\{i,j\} \le k < n\}$. Fix an index $k > i$, and let $X = \{x \mid i < x < k\}$. Consider the ray from s_i^+ pointing vertically downwards. Each point s_x^- for $x \in X$ determines an angle with this ray, when rotating the ray counterclockwise. Let s_{max}^- be the point with the largest such angle from the index set X. If s_k^+ lies above the line supporting the segment $\overline{s_i^+ s_{max}^-}$, then s_{max}^- and hence all s_x^- for $x \in X$ lie below $\overline{s_i^+ s_k^+}$ as required for k to be in P'. Conversely, if s_k^+ lies below the line supporting the segment $\overline{s_i^+ s_{max}^-}$, then s_{max}^- would not lie below the line $\overline{s_i^+ s_k^+}$ and so $k \notin P'$. Thus if our algorithm maintains s_{max}^- as we increment k then in constant time we can check if $k \in P'$, an moreover s_{max}^- can be updated in constant time per iteration by comparing the new bottom endpoint with the previous s_{max}^-. Thus $P' = \textsc{Positive}(i)$ can be computed in linear time, as shown in Algorithm 2, in Appendix A.1. A similar argument allows us to compute $N' = \textsc{Negative}(j)$ in linear time as is also shown in Algorithm 2.

Now we describe how to compute $\textsc{EndRight}(i,j)$ in linear time. Specifically, we seek to determine if there exists a point $r = (r.x, r.y)$ on s_n such that for all $i < x < n$, s_x^- lies below $\overline{s_i^+ r}$, and for all $j < x < n$, s_x^+ lies above $\overline{s_j^- r}$. Note that since s_n is a vertical segment, $r.x$ is fixed, and thus all of these constraints can be written as linear constraints in the one variable $r.y$. In particular, restricting r to lie on s_n means that $s_n^-.y \le r.y \le s_n^+.y$. All other constraints can be written as satisfying a right or a left turn check, each of which is expressible by checking the sign of the determinant of a matrix whose three rows are of the form $(1, s_i^+)$, $(1, r)$, and $(1, s_x^-)$. (Note the cross terms in the determinant are linear in the only variable $r.y$.) Thus we are doing a feasibility check of a linear program with $O(n)$ constraints and one variable. This is easily solved in $O(n)$ time by checking whether the tightest lower bound constraint on $r.y$ lies to the left on the real line of the tightest upper bound constraint on $r.y$. □

Now we remove the assumption that s_1 is a single point. In Appendix A.3 we remark how the optimal starting point must lie in a set of $O(n^2)$ canonical points on s_1, thus leading to an easy $O(n^5)$ time solution by trying our above $O(n^3)$ algorithm on all such points.

Instead of reducing to the single point case, we describe an alternative approach which still runs in $O(n^3)$ time. First, for now assume that the top and bottom chains both have at least one interior vertex (i.e. a vertex not on s_1 or s_n). While we cannot compute $\textsc{MinCH}(i,j)$ if either $i = 1$ or $j = 1$, we can compute $\textsc{MinCH}(i,j)$ for all $1 < i,j < n$ in $O(n^3)$ time by the approach of Theorem 8. Let $\textsc{StartLeft}(i,j)$ be defined similarly to $\textsc{EndRight}(i,j)$ above, except that it checks in linear time if there is a point l on s_1 such that s_x^- lies below the segment $\overline{ls_i^+}$ for $1 < x < i$ and s_x^+ lies above the segment $\overline{ls_j^-}$ for $1 < x < j$. Let $T = \{1 < i,j < n \mid \textsc{StartLeft}(i,j) = True\}$, then $3 + \min_{(i,j)\in T} \textsc{MinCH}(i,j)$ would find the minimum solution over all $1 < i,j < n$ pairs that can connect directly to the leftmost edge s_1 (where the $+3$ counts s_i^+, s_j^-, and the vertex on s_1). Unfortunately, this does not count all possible cases as initially there may be several hull vertices on the top chain interior before the first bottom chain interior vertex, and $\textsc{MinCH}(i,j)$ assumes s_i^+ and s_j^- are consecutive in the left to right order of vertices on the hull (i.e. we miss cases of the form $\textsc{MinCH}(1,j)$ and $\textsc{MinCH}(i,1)$). However, there is a simple way to overcome this issue. Rather than trying to directly connect to the left edge, just compute the minimal chains to the left and then append them to the minimal chains we computed on the right. Specifically, let $\textsc{MinCHLeft}(i,j)$ be the same as $\textsc{MinCH}(i,j)$ except that it computes the minimal valid chain pairs to the left (instead of the right) when the previous vertex on the positive chain was s_i^+ and the previous on the negative chain was s_j^-. Note $\textsc{MinCHLeft}(i,j)$ uses $\textsc{StartLeft}(i,j)$ instead of $\textsc{EndRight}(i,j)$, and similarly modifies $\textsc{Positive}(i)$ and $\textsc{Negative}(j)$. Therefore we return

$$2 + \min_{1<i,j<n} \{\textsc{MinCHLeft}(i,j) + \textsc{MinCH}(i,j)\},$$

where the $+2$ counts the vertices s_i^+ and s_j^-. It is important to note here that $\textsc{MinCHLeft}(i,j)$ only selects vertices to the left of $\min\{i,j\}$ and $\textsc{MinCH}(i,j)$ to the

right of $\max\{i,j\}$. That is, both assume there are no hull vertices on segments with indices between $\min\{i,j\}$ and $\max\{i,j\}$, and thus the above approach only works if there exists an index pair i,j from the optimal solution such that s_i^+ and s_j^- are consecutive in the left to right order of vertices on the hull, i.e. there are no hull vertices on segments with indices between $\min\{i,j\}$ and $\max\{i,j\}$. However, it is easy to see this holds by our assumption that there is at least one interior vertex on both the top and bottom chains. As for the running time, observe we can independently precompute all MinCHLeft(i,j) values in $O(n^3)$ time and all MinCH(i,j) values in $O(n^3)$ time, and this dominates the time to compute the above minimum over all index pairs.

So what remains is to handle the case when the optimal solution may not have an interior vertex on either the top or bottom chain. We have the following lemma for the case when the top chain has no interior vertex, the bottom chain case is handled symmetrically. Due to space the proof is in Appendix A.2.

Lemma 9 *For a set S of n vertical segments, the optimal solution to Problem 1 where the top chain of the convex hull is not allowed to have interior vertices can be solved in $O(n^3)$ time.*

By running all cases for whether the bottom or top chain interiors are empty and taking the minimum we thus have the following.

Theorem 10 *For a set S of n vertical segments, Problem 1 can be solved in $O(n^3)$ time.*

In Appendix A.3 we briefly remark how our approach can be extended to ordered axis-aligned rectangles.

3 NP-Hardness for General Segments

We now argue that when the segments in S are not required to be vertical, then Problem 1 is NP-hard. The proof is by reduction from the standard NP-hard problem CNF-SAT. Our reduction closely follows the approach of the NP-hardness proof in [18] for maximizing the area of the convex hull for uncertain line segments. However, our construction requires additional points in the clause and variable gadgets, in large part since our problem involves minimization and theirs maximization.

Let the given instance of CNF-SAT have n variables and m clauses. Call an uncertain segment a certain point if its two endpoints are the same. Consider a circle in the plane, and evenly place a set of certain points, $B = \{b_1, \ldots, b_{n+m}\}$, along this circle. Call these our base points. Observe that if we conceptually remove $\mathcal{CH}(B)$ then we are left with a set of disjoint circular caps, $c_1, \ldots, c_{n+m}$, each bounded some segment $\overline{b_i b_{i+1}}$ and corresponding circular arc from b_i to b_{i+1}, see Figure B.1 in Appendix B. We have one cap for each variable and one for each clause. All remaining uncertain segments we construct will have both their endpoints in the same cap, or in two different caps when those caps correspond to a variable and a clause that contains it. Note that since all segment endpoints will lie in the circle, all base points are always vertices of the convex hull in any realization. This conceptually separates the caps, in the sense if you added a point in one of the caps, the area it adds to the convex hull is confined to that cap. The way we then connect a variable and clause cap is by adding an uncertain segment between them.

First, consider the cap for a given clause L. We create one uncertain segment for each literal in L. All these segments share a common endpoint at the center of the clause cap, the other endpoints are in the caps of the respective variables. Let this common endpoint be denoted e and let b and b' be the base points of the clause cap for L. We add a convex chain of z certain points from b to b' such that all these points are contained in $\mathcal{CH}(\{e, b, b'\})$. Here z an integer value, to be determined shortly, but intuitively we require z be set large enough so that one of the segments adjacent to e, must select e as its realization to cover these z points. See Figure 3.1.

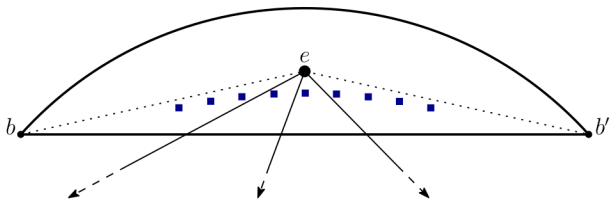

Figure 3.1: Clause cap with common endpoint e. Convex chain of z certain points shown as squares.

Now consider the cap c for some variable x, with corresponding base points b and b'. Within c we add a segment $\overline{tf}$ above and parallel to the segment $\overline{bb'}$, where ultimately selecting t or f will correspond to setting the variable to True or False, respectively. Let l be the maximum over all variables of the maximum of the number of times that variable appears as a positive literal or appears as a negative literal. For the variable x we create a convex chain of l "positive" vertices, P, and a convex chain of l "negative" vertices, N. Specifically, we require (i) every point of P is a vertex of $\mathcal{CH}(P \cup \{b, b', f\})$ (ii) every point of N is a vertex of $\mathcal{CH}(N \cup \{b, b', t\})$ (iii) $N \subset \mathcal{CH}(\{b, b', f\})$, (iv) $P \subset \mathcal{CH}(\{b, b', t\})$, and (v) for any point $v \in \overline{tf}$ if $\mathcal{CH}(\{b, b', v\})$ contains a point of P (resp. N) it does not contain a point of N (resp. P). See Figure 3.2. Recall that for each literal occurrence of x we created an uncertain segment with one end fixed at the corresponding clause. We now make the other end of the uncertain segment a unique point in P or N, depending on whether it appeared as a positive or

negative literal in the clause. We place a certain point at any unused points in P or N. Finally, for each point u in either P or N, we create a small convex chain of z certain points R_u just below it, such that all the points in R_u are contained in $\mathcal{CH}(\{b, b', u\})$ and none are contained $\mathcal{CH}(\{b, b', u'\})$ for any other point u' in either P or N.

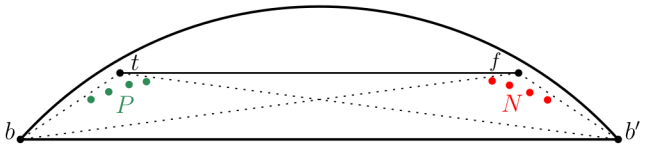

Figure 3.2: Variable cap. Convex chains of z certain points below points in P and N, as well as uncertain segments adjacent to P and N, are not shown.

To argue correctness of the reduction, first suppose there is satisfying assignment to the given CNF-SAT instance. In this case we argue there is a realization of our uncertain segments with $\leq 2m+(l+2)n$ vertices on the convex hull. Specifically, for each variable x, if $x = True$ then in the cap for x we select t for the segment $\overline{tf}$, for each segment adjacent to a point $u \in N$ we select u, and for each segment adjacent to a point in P we select the opposite (i.e. clause) endpoint of the segment. Note that by construction $\mathcal{CH}(N \cup \{b, b', t\})$ contains both P and all the convex chains R_u that we added for each u in P or N, and thus in this case only t and the l points in N are vertices of the convex hull within this cap. Similarly, if $x = False$, we select f for the segment $\overline{tf}$, for each segment adjacent to a point $u \in P$ we select u, and for each segment adjacent to a point in N we select the opposite (i.e. clause) endpoint of the segment. Again, in this case only f and the l points in P are vertices of the convex hull within this cap. On the other hand, for the clause caps, observe that because this was a satisfying assignment for the CNF-SAT instance, we must have selected the common end point in each clause cap for at least one of its adjacent segments. Since for each clause, with common endpoint e and base points b and b', the convex chain of certain points in its cap is contained in $\mathcal{CH}(\{e, b, b'\})$, the number of vertices on the convex hull from this cap is just 1. Thus the total contribution from all clause and variable caps is $m + (l + 1)n$, and since the $n + m$ base points are always on the hull, we thus have $\leq 2m + (l + 2)n$ vertices as claimed.

Now suppose there is no satisfying assignment for the given CNF-SAT instance. In this case we argue that by setting the parameter z to be large enough,the convex hull of any realization has $> 2m + (l + 2)n$ vertices. Specifically, if $z = 2m + (l + 2)n + 1$, then clearly if any one of the chains with z certain points is entirely on the hull then the realization has $> 2m + (l + 2)n$ vertices. Define E as the set of all uncertain segments adjacent to points in P or N from any variable cap, but realized outside the corresponding variable cap. Consider one of the chains with z certain points in some clause cap with common end point e. In order for this chain to not entirely appear on the hull, at least one of the uncertain segments adjacent to e must be in E, and in particular must have its realization somewhere on the e side of the chain. In a minimal solution it can be assumed to be at the point e itself, since as discussed above placing it at e means this clause cap only contributes one vertex, and clearly this cap must contribute at least one vertex in any realization. Now consider a variable cap with base points b and b'. By condition (v) from above, for any point $v \in \overline{tf}$ if $\mathcal{CH}(\{b, b', v\})$ contains a point of P (resp. N) it does not contain a point of N (resp. P). Thus in a minimal solution we can assume $\overline{tf}$ is realized at either t or f. Suppose it is realized at t (the f case is symmetric), and recall that N is not in $\mathcal{CH}(\{t, b, b'\})$. Thus by the same argument as for the clause caps, for every uncertain segment adjacent to a point $u \in N$, a minimal solution can be assumed to select u as the realization, as the chain of z points R_u must be covered. (Recall if u has no adjacent segment we already placed a certain point there.) More generally, in order for a solution to have $< z$ vertices on the convex hull, for any variable v, all uncertain segments that it contributes to E are either all adjacent to points in P (when t is selected) or all adjacent to points in N (when f is selected). So consider the collection of all t and f endpoints chosen for all variable caps in a minimal solution, which thus determines which uncertain segments can fall in E. This collection can be viewed as a variable assignment for the given CNF-SAT instance, and as this instance is not satisfyable, some clause in this assignment evaluates to false. However, this implies that for some common endpoint e in some clause cap, there are no segments adjacent to e that are in E, and hence the number of vertices on the hull is at least $z = 2m + (l + 2)n + 1$.

Thus if we can decide whether there is a realization with $\leq 2m + (l + 2)n$ convex hull vertices, then we can decide the corresponding CNF-SAT instance. Also, it is easy to see that the above uncertain segments can be constructed such that all endpoints are rational points of polynomial complexity (see [18]). Thus we have the following theorem for the decision version of Problem 1.

Theorem 11 *Given a set S of n uncertain segments and an integer k, the problem of determining whether there is a realization of S with $\leq k$ vertices on the convex hull is NP-Complete.*

References

[1] P. K. Agarwal, S. Har-Peled, S. Suri, H. Yildiz, and W. Zhang. Convex hulls under uncertainty. *Algorithmica*, 79(2):340–367, 2017.

[2] E. M. Arkin, C. Dieckmann, C. Knauer, J. S. B. Mitchell, V. Polishchuk, L. Schlipf, and S. Yang. Convex transversals. *Comput. Geom.*, 47(2):224–239, 2014.

[3] K. Buchin, M. Löffler, P. Morin, and W. Mulzer. Preprocessing imprecise points for delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011.

[4] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

[5] A. Driemel, S. Har-Peled, and B. Raichel. On the expected complexity of voronoi diagrams on terrains. *ACM Trans. Algorithms*, 12(3):37:1–37:20, 2016.

[6] A. Driemel, H. Haverkort, M. Löffler, and R. Silveira. Flow computations on imprecise terrains. *Journal of Computation Geometry (JoCG)*, 4(1):38–78, 2013.

[7] R. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. pages 326–333, 1989.

[8] M. T. Goodrich and J. Snoeyink. Stabbing parallel segments with a convex polygon. *Computer Vision, Graphics, and Image Processing*, 49(2):152–170, 1990.

[9] C. Gray, F. Kammer, M. Löffler, and R. Silveira. Removing local extrema from imprecise terrains. *Comput. Geom.*, 45(7):334–349, 2012.

[10] L. J. Guibas, J. Hershberger, J. S. B. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Int. J. Comput. Geometry Appl.*, 3(4):383–415, 1993.

[11] S. Har-Peled. On the expected complexity of random convex hulls. *CoRR*, abs/1111.5340, 2011.

[12] S. Har-Peled and B. Raichel. On the complexity of randomly weighted multiplicative voronoi diagrams. *Discret. Comput. Geom.*, 53(3):547–568, 2015.

[13] A. Jørgensen, M. Löffler, and J. M. Phillips. Geometric computations on indecisive points. In *Workshop on Algorithms and Data Structures (WADS)*, pages 536–547, 2011.

[14] V. Keikha, M. van de Kerkhof, M. J. van Kreveld, I. Kostitsyna, M. Löffler, F. Staals, J. Urhausen, J. L. Vermeulen, and L. Wiratma. Convex partial transversals of planar regions. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 52:1–52:12, 2018.

[15] N. Kumar, B. Raichel, S. Suri, and K. Verbeek. Most likely voronoi diagrams in higher dimensions. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 65 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[16] M. Löffler and J. Snoeyink. Delaunay triangulations of imprecise points in linear time after preprocessing. In *Proc. of 24th ACM Symp. on Comp. Geom. (SoCG)*, pages 298–304, 2008.

[17] M. Löffler and M. J. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. In *Workshop on Algorithms and Data Structures (WADS)*, pages 446–457, 2007.

[18] M. Löffler and M. J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

[19] A. Mukhopadhyay, C. Kumar, E. Greene, and B. K. Bhattacharya. On intersecting a set of parallel line segments with a convex polygon of minimum area. *Inf. Process. Lett.*, 105(2):58–64, 2008.

[20] D. Rappaport. Minimum polygon transversals of line segments. *Int. J. Comput. Geometry Appl.*, 5(3):243–256, 1995.

[21] S. Suri, K. Verbeek, and H. Yildiz. On the most likely convex hull of uncertain points. In *European Symposium on Algorithms (ESA)*, pages 791–802, 2013.

A Vertical Line Segments

A.1 Missing Algorithm

The following algorithm is used as a subroutine in Algorithm 1, and is described in detail in the proof of Theorem 8.

Algorithm 2 Computes the set of valid positive or negative chain vertices

```
1:  function POSITIVE(i)
2:      P ← ∅
3:      s⁻_max ← s⁻_{i+1}
4:      for k ← i + 1 to n − 1 do
5:          if s⁺_k above line through s⁺_i and s⁻_max
                then P ← P ∪ {k}
6:          if s⁻_k above line through s⁺_i and s⁻_max
                then s⁻_max ← s⁻_k
7:      return P
8:  function NEGATIVE(j)
9:      N ← ∅
10:     s⁺_min ← s⁺_{j+1}
11:     for k ← j + 1 to n − 1 do
12:         if s⁻_k below line through s⁻_j and s⁺_min
                then N ← N ∪ {k}
13:         if s⁺_k below line through s⁻_j and s⁺_min
                then s⁺_min ← s⁺_k
14:     return N
```

A.2 Missing Proofs

Lemma 6. *Let P^+, P^- be a valid chain pair. Then $\mathcal{CH}(P^+ \cup P^-)$ intersects all segments in S.*

Proof. Note that P^+ and P^- both start on the same point on s_1 and end on the same point on s_n, and thus clearly $\mathcal{CH}(P^+ \cup P^-)$ intersects s_1 and s_n. So fix some segment $s = s_i$, for $1 < i < n$. From the lemma statement, there exists a point p from the chain P^+ which lies directly above s^- (p may be a vertex or an interior edge point). Similarly, define q as the point from P^- which lies directly below s^+. Thus we have $q.y \leq s^+.y$ and $s^-.y \leq p.y$. If $s^+.y \leq p.y$ then $q.y \leq s^+.y \leq p.y$ and hence $s^+ \in \overline{pq} = \mathcal{CH}(\{p, q\}) \subseteq \mathcal{CH}(P^+ \cup P^-)$. So assume otherwise, that $s^+.y > p.y$, which combined with our known inequality we have $s^+.y > p.y \geq s^-.y$. That is, p lies on the segment s, and hence $p \cap s = p \in \mathcal{CH}(P^+ \cup P^-)$. □

Lemma 9. *For a set S of n vertical segments, the optimal solution to Problem 1 where the top chain of the convex hull is not allowed to have interior vertices can be solved in $O(n^3)$ time.*

Proof. First consider the case when the bottom chain also has no interior vertices. Then we are looking for a single segment $\overline{lr}$ such that $l \in s_1$, $r \in s_2$, and which intersects all segments in S. Thus we are checking the feasibility of a linear program with $O(n)$ constraints in two variables, $l.y$ and $r.y$, which can be solved in $O(n^2)$ time by standard techniques (see [4]).

So now suppose the bottom chain has at least one interior vertex. First we precompute for every possible starting and ending index pair the minimal length negative subchain which could be in a valid chain pair. Specifically, for any pair of indices $1 < i \leq j < n$, let MINNEG(i, j) be the minimum number of vertices of a negative chain from s_i^- to s_j^- such that s_x^+ lies above the chain for all $i < x < j$. Observe that we can easily compute MINNEG(i, j) for all pairs $1 < i \leq j < n$ in $O(n^3)$ time using a similar but simpler dynamic programming approach as was done for MINCH(i, j) in Algorithm 1. Namely, the algorithm follows by the recursive relation MINNEG$(i, j) = 1 + \min_{k \in N}$ MINNEG(k, j) where N is the set of indices $i < k \leq j$, such that for all $i < x < k$, s_x^+ lies above the segment $\overline{s_i^- s_k^-}$. (N can be computed in linear time, similar to NEGATIVE(j) in Algorithm 2.)

By Corollary 7, we know the optimal solution is $2 + \min_{(i,j) \in V}$ MINNEG(i, j), where V is the set of all index pairs such that the minimal subchain computed by MINNEG(i, j) can be extended into a valid chain pair (such that the positive chain has no interior vertices). Specifically, V is the set of index pairs $1 < i \leq j < n$ where there exists points $l \in s_1$ and $r \in s_n$ such that 1) s_x^+ lies above $\overline{ls_i^-}$ for all $1 < x < i$, 2) s_x^+ lies above $\overline{s_j^- r}$ for all $j < x < n$, and 3) s_x^- lies below $\overline{lr}$ for all $1 < x < n$. Consider the first condition. Let $top = \min_{1<x<i} Int(s_x^+, s_i^-).y$, where $Int(s_x^+, s_i^-)$ denotes the point of intersection of the line supporting $\overline{s_x^+ s_i^-}$ with the vertical line supporting s_1. Then condition 1) is equivalent to requiring that $l.y \leq top$, and so this condition can be encoded by simply replacing the upper endpoint of s_1 with the point $(s_1^+.x, top)$ (if $top < s_1^-.y$ then $(i, j) \notin V$). Similarly, we can update the lower endpoint of s_n to handle condition 2) from above. Updating the endpoints in this manner takes $O(n)$ time for any given pair (i, j).

Thus all that remains is to handle condition 3). Here we require $\overline{lr}$ lies above s_x^- for all $1 < x < n$, where $l \in s_1$ and $r \in s_n$. Let E denote the set of all relevant endpoints, i.e. s_1^+, s_1^-, s_n^+, s_n^- and all s_x^- for $1 < x < n$. If such a segment $\overline{lr}$ exists, then we can translate it vertically downwards until it hits a point in E, and then rotate about that point until it hits a second endpoint in E, and it will still be a valid solution. Thus it suffices to limit our search to the set of all segments passing through two points in E. Now there are a few cases. First, suppose one of these two points is s_1^-. Consider the ray with base point s_1^- and pointing vertically up-

ward. Let s_k^- be the first point hit in the set of all s_x^- for $1 < x < n$, when rotating this ray clockwise. Clearly $\overline{lr}$ must pass above s_k^- and if it does then it passes above all s_x^- for $1 < x < n$. Thus in this case a valid $\overline{lr}$ exists if and only if the line supporting $\overline{s_1^- s_k^-}$ passes below s_n^+. Thus we can check all cases when one of the two points is s_1^- in $O(n)$ time, as this is how long it takes to compute s_k^-. A similar argument works for the cases when one of the two points is s_1^+, s_n^+, or s_n^-. So now suppose $\overline{lr}$ passes through two points s_g^- and s_h^-, such that $1 < g < h < n$. Observe that for $\overline{lr}$ to lie above s_x^- for all $1 < x < n$, this is equivalent to requiring $\overline{lr}$ to lie above the top chain of the convex hull of all such s_x^-. In other words, $\overline{s_g^- s_h^-}$ must define an edge of the top chain of the convex hull. There are only $O(n)$ such edges, all of which can be computed globally once in $O(n \log n)$ time (i.e. they do not need to be recomputed for each MinNeg(i, j)). For each such edge in constant time we can check whether the line supporting it intersects s_1 and s_n. Thus in $O(n)$ time (ignoring the global $O(n \log n)$ top chain computation) we can check all cases where $\overline{lr}$ passes through two points s_g^- and s_h^-, such that $1 < g < h < n$. So overall, for any pair (i, j) we can check in $O(n)$ time whether it lies in V, and thus by checking all pairs we can compute V in $O(n^3)$ time. □

A.3 Missing Remarks

Remark 12 *The case when s_1 is a vertical segment can be directly reduced to the single point case in Theorem 8, but the run time degrades. Imagine sliding a point p down the segment s_1. As we slide this point the behavior of* MinCH$(1, 1)$ *from Algorithm 1 only changes when either the set P or N change, and specifically as we slide p downwards the set P gets smaller and N larger. So fix an index k which initially is in P, and consider the moment when k leaves the set P. At this moment, for some $1 < x < k$, p must be the intersection of s_1 with the line supporting $\overline{s_x^- s_k^+}$, namely if p went any lower on s_1 then s_x^- would lie above $\overline{ps_k^+}$. A similar statement holds for changes in the set N. Thus consider the set of all $O(n^2)$ intersection points of the segment s_1 with lines supporting segments of the form $s_i^+ s_j^-$ for all pairs i, j. As all possible values for P, N are realizable by starting from some point in this canonical set of points, we can obtain the optimal solution to Problem 1 by calling the algorithm of Theorem 8 for each one of these points. As the running time of each call is $O(n^3)$, this would give an $O(n^5)$ time solution.*

Remark 13 *It is not hard to see that the approach in Section 2 also gives a polynomial time algorithm for Problem 1 when U is a set of axis-aligned rectangles that can be appropriately ordered. Specifically, suppose you are given the points l, r, t, b representing the leftmost, rightmost, topmost, and bottommost vertices of the optimal convex hull. Similar to Lemma 4, one can argue that there is an optimal solution where all the vertices on the top chain between l and t are realized at the upper left corner of their rectangle, and similar statements hold for the other corners. To use dynamic programming, however, we need to be given an ordering, such as the left to right order of the realizations of the rectangles. This would occur if, for example, the rectangles are separated by vertical lines, i.e. rectangles $R_1, \ldots R_n$ such that for any $i < j$, R_i lies entirely to the left of R_j. Note there are only a polynomial number of possibilities for l, r, t, b as their rectangles can be guessed, and there are only a polynomial number of canonical positions that to need be considered for their realization in each rectangle (similar to Remark 12). Thus this gives a polynomial time algorithm when we have such an ordering, though the constant would be high without similar optimizations as in the vertical segment case.*

B NP-Hardness for General Segments

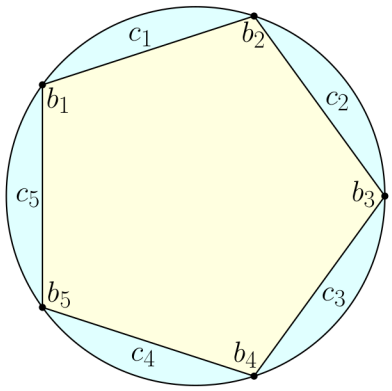

Figure B.1: Certain points $b_1, \ldots b_5$, and caps $c_1, \ldots, c_5$.