

A Combinatorial Proof of Universal Optimality for Computing a Planar Convex Hull

Ivor van der Hoog 

IT University of Copenhagen, Denmark

Eva Rotenberg 

IT University of Copenhagen, Denmark

Daniel Rutschmann 

IT University of Copenhagen, Denmark

Abstract

For a planar point set P , its convex hull is the smallest convex polygon that encloses all points in P . The construction of the convex hull from an array I_P containing P is a fundamental problem in computational geometry. By sorting I_P in lexicographical order, one can construct the convex hull of P in $O(n \log n)$ time which is worst-case optimal. Standard worst-case analysis, however, has been criticized as overly coarse or pessimistic, and researchers search for more refined analyses.

For an algorithm A , worst-case analysis fixes n , and considers the maximum running time of A across all size- n point sets P and permutations I_P of P . Output-sensitive analysis fixes n and k , and considers the maximum running time across all size- n point sets P with k hull points and permutations I_P of P . Universal analysis provides an even stronger guarantee. It fixes a point set P and considers the maximum running time across all permutations I_P of P .

Kirkpatrick, McQueen, and Seidel [SICOMP'86] consider output-sensitive analysis. If the convex hull of P contains k points, then their algorithm runs in $O(n \log k)$ time. Afshani, Barbay, Chan [FOCS'07] prove that the algorithm by Kirkpatrick, McQueen, and Seidel is also universally optimal. Their proof restricts the model of computation to any algebraic decision tree model where the test functions have at most constant degree and at most a constant number of arguments. They rely upon involved algebraic arguments to construct a lower bound for each point set P that matches the universal running time of [SICOMP'86].

We provide a different proof of universal optimality. Instead of restricting the computational model, we further specify the output. We require as output (1) the convex hull, and (2) for each internal point of P a witness for it being internal. Our argument is shorter, perhaps simpler, and applicable in more general models of computation.

2012 ACM Subject Classification Theory of computation \rightarrow Computational geometry; Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Convex hull, Combinatorial proofs, Universal optimality

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.102

Funding This work was supported by the Carlsberg Foundation Fellowship CF21-0302 “Graph Algorithms with Geometric Applications”, the VILLUM Foundation grant (VIL37507) “Efficient Recomputations for Changeful Problems”, and the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987.

1 Introduction

Convex hulls are a central topic in computational geometry. By sorting a point set P in lexicographical order, one can construct its convex hull in $O(n \log n)$ time, which is worst-case optimal. Traditional worst-case analysis often faces criticism for being overly coarse or pessimistic. To address these concerns, researchers introduced more nuanced complexity measures that account not only for the input size but also for additional parameters that



© Ivor van der Hoog, Eva Rotenberg, and Daniel Rutschmann;
licensed under Creative Commons License CC-BY 4.0

33rd Annual European Symposium on Algorithms (ESA 2025).

Editors: Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman; Article No. 102; pp. 102:1–102:13

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

capture the difficulty of the instance. A classical example is output-sensitive analysis, where the running time analysis depends on the output size [7]. Other algorithms measure their performance by the spread of the point set, defined as the ratio of the maximum to minimum pairwise distances among the points [4]. More examples include algorithmic complexity parametrized on various geometric characteristics of P , such as the ratio of circumradii to inradii or the number of reflex angles in an input polygon [3, 8].

This paper improves upon the work by Afshani, Barbay, and Chan [1] (see also their journal version [2]). They observe that, for geometric problems, the worst-case algorithmic analysis contains a double maximum. They only consider *correct* algorithms, which we formally define in the preliminaries. For now, denote by \mathbb{P}_n all point sets of size n and for any $P \in \mathbb{P}_n$ by \mathbb{I}_P all size- n arrays that contain the points of P in some order. For an algorithm A , denote by $\rho(A, I_P)$ its runtime when the input is I_P . Then, for a fixed correct algorithm A and input size n , the worst-case running time is:

$$\text{worst-case}(A, n) := \max_{P \in \mathbb{P}_n} \max_{I_P \in \mathbb{I}_P} \rho(A, I_P).$$

An algorithm A is worst-case optimal if there exists no algorithm A' whose worst-case running time is asymptotically smaller than that of A . Prior works perform a finer-grained analysis by restricting the first maximum. For example, output-sensitive running time is defined as:

$$\text{output-sensitive}(A, n, k) := \max_{P \in \mathbb{P}_n \text{ and } P \text{ has } k \text{ points on the convex hull}} \max_{I_P \in \mathbb{I}_P} \rho(A, I_P).$$

Afshani, Barbay, and Chan [1] obtain an even stronger quality guarantee by eliminating the first maximum all-together. They call their notion of optimality *instance-optimality in the order-oblivious setting*. Recently, this notion has been called *universal optimality* [5, 6, 9]. For a fixed point set P , we define the *universal* running time of an algorithm A as:

$$\text{universal}(A, P) := \max_{I_P \in \mathbb{I}_P} \rho(A, I_P).$$

An algorithm A is universally optimal if for all point sets P , there exists no algorithm A' whose universal running time is asymptotically smaller than that of A . Observe that any universally optimal algorithm is automatically output-sensitive.

Contribution. Afshani, Barbay, Chan prove that the algorithm by Kirkpatrick, McQueen, and Seidel [7] is universally optimal.¹ Their proof restricts the model of computation to any algebraic decision tree model where the test functions have at most constant degree and have at most a constant number of arguments. They present an involved algebraic argument to construct a lower bound for each point set P that matches the universal running time of [7].

Here, we give a different proof. Rather than restricting the model of computation, we give a more extensive but natural specification of the output. We call a point of P *internal* if it does not appear on the boundary of the convex hull. We require that any algorithm computes (1) the convex hull and (2) for each internal point a witness for its being internal. For a point set P , we apply a simple combinatorial counting argument over all outputs to obtain a lower bound that matches the universal running time of [7]. Our argument is shorter, arguably simpler, and holds in more general models of computation.

¹ We note that the paper has only Kirkpatrick and Seidel as authors. However, a footnote remarks that McQueen suggests an adaption to the algorithm. Only with this adaption, the algorithm becomes universally optimal.

2 Preliminaries

We follow [1] and assume that the input points lie in general position. I.e., P contains no duplicates and no collinear triples. For any integer n , we denote by \mathbb{P}_n all planar point sets of n points that lie in general position. We denote by \mathbb{I}_P all arrays of size n that store P .

► **Definition 1.** For a planar point set P , its convex hull $\text{CH}(P)$ is the minimum convex region that contains all points in P .

► **Definition 2.** For a planar point set P , its convex hull vertices $\text{ch}(P)$ are the points in P that lie on the boundary of $\text{CH}(P)$.

The Convex Hull Problem. Our input is an array $I_P \in \mathbb{I}_P$ of n points P in general position. Convex hull algorithms output $\text{ch}(P)$ in cyclical ordering. For our analysis, we further require that the output contains, for all points in $P - \text{ch}(P)$, a *witness* that they are internal:

► **Definition 3.** For any $p \in P$, a triangle t (whose corners lie in P) is a witness of p if p is contained in the interior of t .

► **Observation 4.** A point $p \in P$ does not lie on the boundary of $\text{CH}(P)$ if and only if there exists at least one witness of p .

Requiring the output to contain witnesses is a natural condition as it is effectively asking the algorithm A to verify its correctness. The concept of certification and verifiability is central to the design of algorithms. Without this requirement, there are oblivious decision trees (defined below) that compute the convex hull in $O(k \log n)$ time.

Formalising output. Our output consists of two lists: A *hull list* H encodes $\text{ch}(P)$ in cyclical order, and a *witness list* W defines for each point in $P - \text{ch}(P)$ a witness. Formally, a *hull list* $H = (h_1, \dots, h_k)$ of I_P is a sequence of integers such that the points $I_P[h_i]$, for $i \in [k]$, are precisely the k points on the boundary of $\text{CH}(P)$ in their cyclical ordering. We require that $I_P[h_1]$ is the leftmost point of $\text{ch}(P)$. A *witness list* (we refer ahead to Figure 1) of I_P is a sequence $W = (a_i, b_i, c_i)_{i=1}^n$ of integer triples such that:

- $a_i = b_i = c_i = -1$ if $I_P[i]$ lies on $\text{CH}(P)$, and otherwise
- the triangle $(I_P[a_i], I_P[b_i], I_P[c_i])$ is a witness of $I_P[i]$.

Algorithms. We define an *algorithm* for convex hull construction as any comparison-based algorithm that receives any ordered set of planar points I_P and outputs a hull list and witness list of I_P . Given a fixed input I_P and an algorithm A , the *running time* $\rho(A, I_P)$ is the number of instructions used by A to terminate on the input I_P .

Decision trees. For $n \in \mathbb{N}$, we define an *abstract decision tree* T as a tree whose inner nodes have two children. The inner nodes contain no additional information. Its leaves contain two ordered lists. The first is an arbitrary list of integers and the second is a list of n integer triples. We define an *oblivious decision tree* T as any abstract decision tree such that for any point set $P \in \mathbb{P}_n$, and any $I_P \in \mathbb{I}_P$, there exists a root-to-leaf path in T where the two lists at the leaf are a hull list and witness list of I_P , respectively. We denote by \mathcal{T}_n^O the set of all oblivious decision trees for $n \in \mathbb{N}$.

► **Definition 5.** A comparison-based algorithm A is correct if for any input I_P it outputs a correct hull and witness list. Observe that any correct comparison-based algorithm A for convex hull construction has, for each n , a corresponding oblivious decision tree in \mathcal{T}_n^O .

Worst-case optimality. Let \mathcal{A} denote the set of all correct algorithms. For an algorithm $A \in \mathcal{A}$ the *worst-case running time* is defined as

$$\text{worst-case}(A, n) := \max_{P \in \mathbb{P}_n} \max_{I_P \in \mathbb{I}_P} \rho(A, I_P).$$

An algorithm is worst-case optimal if there exists a constant c such that, for all n large enough, $\text{worst-case}(A, n) \leq c \cdot \min_{A' \in \mathcal{A}} \text{worst-case}(A', n)$.

► **Observation 6.** For any algorithm A and n , the worst-case running time of A is lower bounded by the minimum height among all decision trees in \mathcal{T}_n^O . Formally, $\forall A \in \mathcal{A}, \forall n \in \mathbb{N}$,

$$\text{worst-case}(A, n) = \max_{P \in \mathbb{P}_n} \max_{I_P \in \mathbb{I}_I} \rho(A, I_P) \geq \min_{T \in \mathcal{T}_n^O} \text{Height}(T).$$

Universal optimality. Afshani, Barbay and Chan [1] introduce a strictly stronger notion of algorithmic optimality which they call *instance optimality in the order-oblivious setting*. We will instead use the equivalent modern term *universal optimality* [5, 6, 9]. Intuitively, an algorithm is *universally optimal* if it is worst-case optimal for every fixed point set P . Formally, we define the *universal running time* of an algorithm A and point set P as

$$\text{universal}(A, P) := \max_{I_P \in \mathbb{I}_P} \rho(A, I_P).$$

An algorithm is universally optimal if there exists a constant c such that for all P , $\text{universal}(A, P) \leq c \cdot \min_{A' \in \mathcal{A}} \text{universal}(A', P)$. To show universal optimality, we introduce the concept of *clairvoyant decision trees*. For a fixed point set P , a *clairvoyant decision tree* T is an abstract decision tree such that for any input $I_P \in \mathbb{I}_P$, there exists a root-to-leaf path in T such that the two lists at the leaf are a hull list and witness list of I_P , respectively. We denote by \mathcal{T}_P^C the set of all clairvoyant decision trees for a fixed planar point set P .

► **Observation 7.** For any planar point set P in general position, for any algorithm A , the universal running time of A is lower bounded by the minimum height amongst all clairvoyant decision trees in \mathcal{T}_P^C . Formally, for all planar point set P in general position, for all $A \in \mathcal{A}$,

$$\text{universal}(A, P) = \max_{I_P \in \mathbb{I}_I} \rho(A, I_P) \geq \min_{T \in \mathcal{T}_P^C} \text{Height}(T).$$

Prior work. Kirkpatrick, McQueen, and Seidel [7] give an efficient algorithm to compute for any point set P its convex hull. We present this algorithm in Section 4 and observe that it naturally produces a hull list and a witness list. Afshani, Barbay, and Chan [1] show that the algorithm in [7] is universally optimal in a somewhat restricted model of computation which we define below. We note for the reader that this definition is not vital to our analysis.

► **Definition 8** (Definition 3.1 [1]). A function $f : (\mathbb{R}^d)^c \mapsto \mathbb{R}$ is *multilinear* if the restriction of f is a linear function from \mathbb{R}^d to \mathbb{R} when any $(c-1)$ of the c arguments are fixed. Equivalently, f is *multilinear* if $f((x_{11}, \dots, x_{1d}), \dots, (x_{c1}, \dots, x_{cd}))$ is a multivariate polynomial function that never multiplies coordinates of the same point.

In the *multilinear decision tree model* [1], algorithms can access the input points only through tests of the form $f(p_1, \dots, p_c) > 0$ for a multilinear function f with c constant.

► **Theorem 9** (Theorem 3.5+3.6 in [1]). The algorithm in [7] is universally optimal for dimension $d = 2$ in the multilinear decision tree model.

We instead show universal optimality by a counting all possible hull and witness lists:

► **Theorem 10.** *If we define \mathcal{A} as the set of all correct comparison-based convex hull algorithms (Definition 5) then the algorithm in [7] for $d = 2$ is universally optimal.*

3 Creating a universal lower bound

Let P be a point set of n points in general position. We use our definition of hull lists and witness lists to show a universal lower bound. That is, we define a quantity Q such that for all algorithms $A \in \mathcal{A}$, $\text{Universal}(A, P) \geq Q$. To this end, we explore the definition of a witness list. Recall that \mathbb{I}_P is the set of all $n!$ arrays that contain P . Consider any list $W = (a_i, b_i, c_i)_{i=1}^n$ of integer triples.

For a fixed W , there can be many $I_P \in \mathbb{I}_P$ for which W is a witness list. For example, let P be a point set containing $n - 3$ points in a small ball around $(0, 0)$, and the three corners (α, β, γ) of a unit equilateral triangle centred at $(0, 0)$ (we refer to Figure 1). Let:

$$W = ((-1, -1, -1), (-1, -1, -1), (-1, -1, -1), (0, 1, 2), (0, 1, 2), \dots, (0, 1, 2)).$$

Any $I_P \in \mathbb{I}_P$ which contains (α, β, γ) in the first three positions has W as a witness list. Specifically, there are $3! \cdot (n - 3)!$ arrays $I_P \in \mathbb{I}_P$ which have W as a witness list. There does not exist any other witness list W^* where more than $3!(n - 3)!$ arrays in \mathbb{I}_P have W^* as a witness list. We use this maximum quantity for our lower bound:

► **Definition 11.** *Let P be a point set and $W = (a_i, b_i, c_i)_{i=1}^n \subset \mathbb{Z}^{n \times 3}$. We define*

$$V(P, W) := \left| \left\{ I_P \in \mathbb{I}_P \mid W \text{ is a witness list of } I_P \right\} \right|.$$

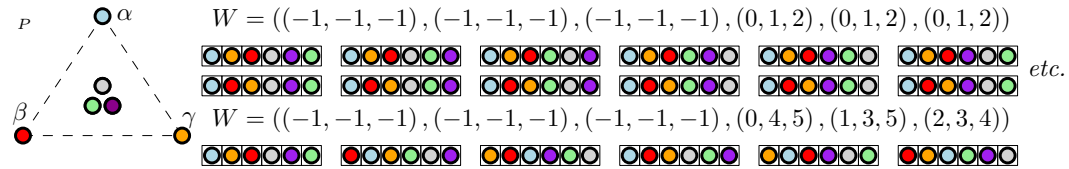
We consider the maximal number of input arrays that can have the same witness list:

$$V_{\max}(P) := \max_{W \in \mathbb{Z}^{n \times 3}} |V(P, W)|.$$

► **Theorem 12.** *Let P be a point set of n points and A be any algorithm in \mathcal{A} , then*

$$\text{universal}(A, P) \in \Omega\left(n + |\text{ch}(P)| \cdot \log n + \log \frac{n!}{V_{\max}(P)}\right).$$

Proof. Any algorithm must spend $\Omega(n)$ time to read the input. Denote by ℓ the minimum height across all clairvoyant decision trees in \mathcal{T}_P^C . By Observation 7, $\Omega(\ell)$ is a universal lower bound. What remains is to bound ℓ . Let $k = |\text{ch}(P)|$. Let T be an arbitrary clairvoyant decision tree in \mathcal{T}_P^C . Recall that each leaf of T stores a hull list and a witness list.



■ **Figure 1** For $W = ((-1, -1, -1), (-1, -1, -1), (-1, -1, -1), (0, 1, 2), (0, 1, 2), (0, 1, 2))$ there are 36 $I_P \in \mathbb{I}_P$ for which W is a witness list (any array that assigns (α, β, γ) to the first three cells). For $W = ((-1, -1, -1), (-1, -1, -1), (-1, -1, -1), (0, 4, 5), (1, 3, 5), (2, 3, 4))$ there are only 6.

For any input $I_P \in \mathbb{I}_P$, there is a unique hull list of I_P . It follows that the number of distinct hull lists, and therefore the number of distinct leaves of T , across all I_P is at least $\frac{n!}{(n-k)!}$. Hence, $\ell \geq \log(\frac{n!}{(n-k)!}) \in \Omega(k \log n)$. For any input $I \in \mathbb{I}_P$, there is at least one leaf in T containing a witness list W of I . Therefore,

$$n! = |\mathbb{I}_P| \leq \sum_{\ell \in L} |V(P, W)| \leq (\# \text{ leaves of } T) \cdot V_{\max}(P) \Rightarrow \ell \geq \log \frac{n!}{V_{\max}(P)}. \quad \blacktriangleleft$$

4 Analysing Kirkpatrick-McQueen-Seidel

Kirkpatrick and Seidel present an output-sensitive algorithm for computing the convex hull for a planar point set [7]. McQueen suggests a modification to the algorithm by Kirkpatrick and Seidel in Footnote 2 of [7]. The modified algorithm, depicted in Algorithm 1, is universally optimal. We first describe their algorithm. Next, we construct a geometric object that we will call a *quadrangle tree* and use it to analyse the running time of Algorithm 1.

High-level overview. Their algorithm is recursive and it constructs the *upper convex hull* of P . The original input consists of an unordered array I_P , from which they find the leftmost and rightmost vertex of P in linear time. Each recursive call has as input some set $S \subset P$ and two points $p_\ell, p_r \in S$ such that (p_ℓ, p_r) is an edge of $\text{CH}(S)$ (Figure 2) – we assume that the line segment between the leftmost and rightmost vertex lies on $\text{CH}(P)$. Otherwise, this line segment splits the problem into two independent convex hull construction problems.

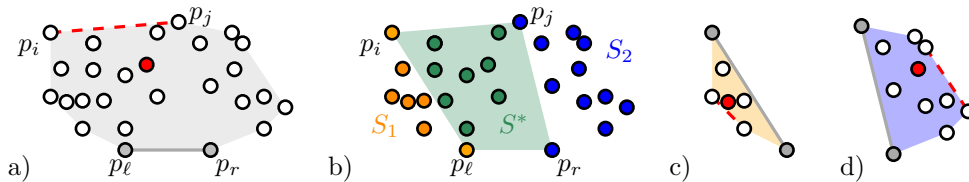
Given (S, p_ℓ, p_r) , the algorithm first finds a point $m \in S$ with the median x -coordinate in linear time. It then finds in linear time an edge (p_i, p_j) of $\text{CH}(P) \cap S$ with $(p_i, p_j) \neq (p_\ell, p_r)$ such that m lies inside or on the quadrangle $Q = (p_\ell, p_r, p_j, p_i)$ (Q may also be a triangle, for example when $p_i = p_\ell$). The algorithm partitions S in linear time into three sets:

- S_1 consists of all points of S that lie above or on the line $p_\ell p_i$,
- S_2 consists of all points of S that lie above or on the line $p_r p_j$, and
- $S^* = S - S_1 - S_2$. S^* lies within Q and, therefore, its points are not on the convex hull.

Through Q we assign these points a witness in $O(|S^*|)$ time at no asymptotic overhead. The algorithm recurses on S_1 and S_2 , using the segments (p_ℓ, p_i) and (p_r, p_j) , respectively.

4.1 Runtime analysis

We propose a novel fine-grained runtime analysis of this algorithm. To this end, we define what we will call the *quadrangle tree* (we refer to Figure 3 (a)).



■ **Figure 2** (a) The input are the points and the grey segment. We compute the point m with median x -coordinate and the edge (p_i, p_j) . (b) Given $Q = (p_\ell, p_r, p_j, p_i)$, we partition the points into three sets: S_1, S^*, S_2 . (c + d) We recurse on S_1 and S_2 using their respective edges of Q .

■ **Algorithm 1** Kirkpatrick-McQueen-Seidel(unordered point set S , edge (p_ℓ, p_r) of $\text{CH}(S)$) [7].

Require: $S = \{p \in P \mid p \text{ lies on or above the line } p_\ell p_r\} \text{ and } p_\ell, p_r \in \text{ch}(P)$.

- 1: **if** $p_\ell = p_r$ **then**
- 2: Append p_ℓ to the hull list
- 3: **return**
- 4: **end if**
- 5: $m \leftarrow \arg \text{median} \{x(p) \mid p \in S\}$
- 6: $(p_i, p_j) \leftarrow \text{edge of } \text{CH}(S) \text{ such that } m \in Q = (p_\ell, p_r, p_j, p_i)$ *bridge-finding* [7].
- 7: $S_1 \leftarrow \{p \in S \mid p \text{ lies on or above the line } p_\ell p_i\}$
- 8: $S_2 \leftarrow \{p \in S \mid p \text{ lies on or above the line } p_j p_r\}$
- 9: $S^* \leftarrow S - S_1 - S_2$
- 10: Kirkpatrick-McQueen-Seidel($S_1, (p_\ell, p_i)$)
- 11: Kirkpatrick-McQueen-Seidel($S_2, (p_j, p_r)$)
- 12: GiveWitness($S^*, p_\ell, p_r, p_j, p_i$) *Note that this last step is not explicit in [7].*

► **Definition 13.** A rooted convex polygon (C, p, q) is a convex polygon C together with an edge pq of C . Given distinct vertices r, s of C the line rs splits C into two pieces (one piece is empty if rs is an edge of C). Let C^{rs} denote the piece that does not contain pq .

► **Definition 14.** A quadrangle tree \mathcal{Q} of a rooted convex polygon (C, p, q) is a binary tree in which every node stores a quadrangle spanned by (up to) four vertices of C , such that:

- The quadrangle at the root node is spanned by p, q, s, r where rs is an edge of C .
We allow for $p = r$ and/or $q = s$.
- If $q \neq r$, then the root has a child node whose subtree is a quadrangle tree of (C^{pr}, p, r) .
- If $s \neq p$, then the root has a child node whose subtree is a quadrangle tree of (C^{qs}, q, s) .

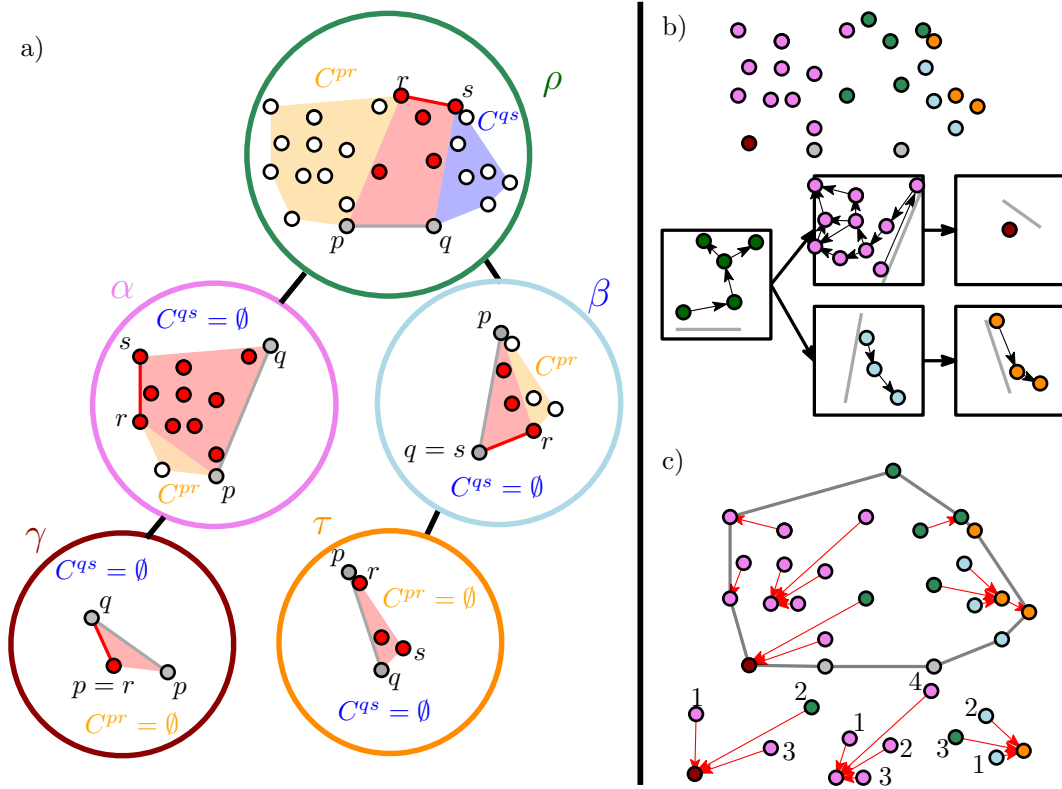
Given a point set P and a quadrangle tree \mathcal{Q} , the *population* of the root node are all points in P that lie on or inside the quadrangle $pqrs$, except for those that lie on the line pq . Observe that, since we assume that P lies in general position, the population of the root node equals all points in the interior of $pqrs$ plus: r if $p \neq r$ and s if $q \neq s$.

► **Observation 15.** Given the input array I_P , the algorithm by Kirkpatrick, McQueen and Seidel finds the leftmost and rightmost vertices of P in linear time. For each point set P Algorithm 1 constructs, independent of the input ordering $I_P \in \mathbb{I}_P$, a quadrangle tree \mathcal{Q} where each call of Kirkpatrick-McQueen-Seidel($S, (p_\ell, p_r)$) uniquely corresponds to a node (C, p, q) in \mathcal{Q} with $C = \text{CH}(S)$, $p = p_\ell$ and $q = p_r$.

Ordered downdrafts. For a point set P and a quadrangle tree \mathcal{Q} of $\text{CH}(P)$, we create an *ordered downdraft*. Intuitively, this is a mapping $\varphi : P - \text{ch}(P) \rightarrow P$ such that for each $p \in P - \text{ch}(P)$, the point $\varphi(p)$ lies deeper in the quadrangle tree, or, in the same node as p but farther from the corresponding root edge. This is our most technically involved definition.

► **Definition 16** (Figure 3(b)). We define for any quadrangle tree \mathcal{Q} of $\text{CH}(P)$ a partial order $\prec_{\mathcal{Q}}$ on P . For $p \in P$, let $r(p)$ be the node in \mathcal{Q} within whose population p lies. Let $H_{r(p)}$ be the halfplane defined by the rooted edge of $r(p)$. For $p, q \in P$, we say that $p \prec_{\mathcal{Q}} q$ if

- $r(p)$ is a strict ancestor of $r(q)$, or
- $r(p) = r(q)$, and q lies deeper inside the halfspace $H_{r(p)}$ than p .



■ **Figure 3** (a) From a rooted polygon (C, p, q) we construct a quadrangle tree \mathcal{Q} with six nodes: $\rho, \alpha, \beta, \gamma, \tau$. For each node in the tree, the red coloured vertices indicate its population.
 (b) We assign all points $p \in P$ a colour based on $r(p)$. For example, for all pink points p the node $r(p)$ equals $\alpha \in \mathcal{Q}$. We illustrate a partial order over the points in P where for all p, q with $r(p) = r(q)$ the point $p \prec_{\mathcal{Q}} q$ if and only if p lies further from the line supporting the grey segment.
 (c) We create an ordered downdraft φ by mapping each point in $P - \text{ch}(P)$ to a point in P . For all points q where multiple points in P map to q , we create a linear order over $\varphi^{-1}(\{q\})$.

Given a quadrangle tree \mathcal{Q} , we define a *downdraft* φ as a map that maps each point in $P - \text{ch}(P)$ to a point in P . Specifically, it maps the point either to a quadrangle lower in the tree, or to a point in the same quadrangle that is strictly further away from the root edge.

► **Definition 17.** Let P be a point set and \mathcal{Q} be a quadrangle tree for $\text{ch}(P)$. A downdraft for (P, \mathcal{Q}) is a map $\varphi : (P - \text{ch}(P)) \rightarrow P$ such that $\forall p \in P$, we have $p \prec_{\mathcal{Q}} \varphi(p)$.

Given any downdraft φ and point $q \in P$, the fiber $\varphi^{-1}(\{q\})$ is the set of all points $p \in P$ that get mapped to q .

► **Definition 18** (Figure 3(c)). An ordered downdraft $\bar{\varphi} = (\varphi, \prec_{\bar{\varphi}})$ of (P, \mathcal{Q}) is a downdraft φ of (P, \mathcal{Q}) plus a total order $\prec_{\bar{\varphi}}$ on each fiber $\varphi^{-1}(\{p\})$ for $p \in P$. We define:

$\text{OD}(P, \mathcal{Q})$ to be the set of all ordered downdrafts of (P, \mathcal{Q}) .

4.2 A universal upper bound

Let A denote the Kirkpatrick-McQueen-Seidel algorithm. Recall that Algorithm 1 naturally creates a quadrangle tree \mathcal{Q} of $\text{CH}(P)$, where each call of $A(S, (p_\ell, p_r))$ uniquely corresponds to a node (C, p, q) in \mathcal{Q} with $C = \text{CH}(S)$, $p = p_\ell$ and $q = p_r$. We show a universal upper bound for the running time of A by counting the number of ordered downdrafts of \mathcal{Q} .

► **Theorem 19.** *Let P be a planar point set in general position with n points. Let \mathcal{Q} be the corresponding quadrangle tree from Observation 15 and let A denote the algorithm by Kirkpatrick, McQueen and Seidel which as input receives some $I_P \in \mathbb{I}_P$ (and the leftmost and rightmost point in P). Then there exists a constant c such that*

$$\forall I_P \in \mathbb{I}_P, \quad \rho(A, I_P) \leq c \cdot \left(n + \log \frac{n^n}{|\text{OD}(P, \mathcal{Q})|} \right).$$

In Section 5, we show that Algorithm 1 is universally optimal by combining the upper bound from Theorem 19 with the lower bound from Theorem 12.

The proof. Every node in \mathcal{Q} corresponds to a call $A(S, (p_\ell, p_r))$ with $p_\ell \neq p_r$. The quadrangle at this node is $p_\ell p_r p_j p_i$, and its child nodes correspond to the calls $A(S_1, (p_\ell, p_i))$ and $A(S_2, (p_j, p_r))$. We denote the subtree of this node by $\mathcal{Q}(S)$. We put $\mathcal{Q} = \mathcal{Q}(P)$.

► **Lemma 20.** *Let S, S_1, S_2 be the sets in some recursive call of Algorithm 1. Then*

$$|\text{OD}(S, \mathcal{Q}(S))| \leq |S|^{|S \setminus (S_1 \cup S_2)|} \cdot |\text{OD}(S_1, \mathcal{Q}(S_1))| \cdot |\text{OD}(S_2, \mathcal{Q}(S_2))|.$$

Proof. First, we show the following upper bound:

$$|\text{OD}(S, \mathcal{Q}(S))| \leq |S|^{|S \setminus (S_1 \cup S_2)|} \cdot |\text{OD}(S_1 \cup S_2, \mathcal{Q}(S))|.$$

Indeed, any ordered downdraft in $\text{OD}(S, \mathcal{Q}(S))$ can be obtained from an ordered downdraft in $\text{OD}(S_1 \cup S_2, \mathcal{Q}(S))$ by placing each element of $S \setminus (S_1 \cup S_2)$ in some fiber, and by placing it at some point in the ordering of the fiber. Within a fiber, an element of S can be placed in at most $|S|$ different ways.

Finally, note that all points in $S_1 - CH(P)$ lie in the left child of $\mathcal{Q}(S)$ and all points in $S_2 - CH(P)$ lie in its right child. Thus, any downdraft over $(S_1 \cup S_2, \mathcal{Q}(S))$ may be obtained by constructing downdrafts over $(S_1, \mathcal{Q}(S_1))$ and $(S_2, \mathcal{Q}(S_2))$ independently, and thus

$$|\text{OD}(S_1 \cup S_2, \mathcal{Q}(S))| = |\text{OD}(S_1, \mathcal{Q}(S_1))| \cdot |\text{OD}(S_2, \mathcal{Q}(S_2))|. \quad \blacktriangleleft$$

For any call $A(S, (p_\ell, p_r))$ made during the execution of the Kirkpatrick-McQueen-Seidel algorithm, let $T(S)$ denote total running time spent on this call (including recursion). Since both median finding and bridge finding take $O(|S|)$ time, there is an absolute constant c such that:

$$T(S) \leq T(S_1) + T(S_2) + c \cdot |S|. \quad \text{For this constant } c, \text{ we show:}$$

► **Lemma 21.** *Denote for any call $A(S, (p_\ell, p_r))$ by $T(S)$ the time spent on executing the call (including all time spent during recursion). Then*

$$T(S) \leq c \cdot \left(|S| + |S| \log |S| - \log |\text{OD}(S, \mathcal{Q}(S))| \right).$$

Proof. We use induction over $|S|$. If $p_\ell = p_r$, then $|S| = 1$ and $T(S) \leq c$. Otherwise, let $n = |S|, n_1 = |S_1|, n_2 = |S_2|$. By induction,

$$\begin{aligned} T(S) &\leq T(S_1) + T(S_2) + c \cdot |S| \\ &\leq c \cdot \left(n_1 + n_2 + n_1 \log n_1 + n_2 \log n_2 - \log |\text{OD}(S_1, \mathcal{Q}(S_1))| - \log |\text{OD}(S_2, \mathcal{Q}(S_2))| + n \right) \\ &\leq c \cdot \left(n_1 \log(2n_1) + n_2 \log(2n_2) - \log |\text{OD}(S_1, \mathcal{Q}(S_1))| - \log |\text{OD}(S_2, \mathcal{Q}(S_2))| + n \right) \end{aligned}$$

102:10 A Combinatorial Proof of Universal Optimality for Computing a Planar Convex Hull

Since, S_1, S_2 are computed via the median x-coordinate, $n_1, n_2 \leq \frac{n}{2}$. By Lemma 20,

$$-\log |\text{OD}(S_1, \mathcal{Q}(S_1))| - \log |\text{OD}(S_2, \mathcal{Q}(S_2))| \leq -\log |\text{OD}(S, \mathcal{Q}(S))| + (n - n_1 - n_2) \log n$$

Therefore,

$$\begin{aligned} T(S) &\leq c \cdot \left(n_1 \log n + n_2 \log n - \log |\text{OD}(S, \mathcal{Q}(S))| + (n - n_1 - n_2) \log n + n \right) \\ &= c \cdot \left(n + n \log n - \log |\text{OD}(S, \mathcal{Q}(S))| \right) \end{aligned} \quad \blacktriangleleft$$

We now apply Lemma 21 to the first call $A(P, (p_\ell, p_r))$ of Algorithm 1. Note that per definition, $\mathcal{Q}(P) = \mathcal{Q}$ and $|P| = n$. Lemma 21 guarantees that, regardless of the order I_P we receive the input P in, the running time is at most

$$c \cdot \left(n + n \log n - \log |\text{OD}(P, \mathcal{Q})| \right) = c \cdot \left(n + \log \frac{n^n}{|\text{OD}(P, \mathcal{Q})|} \right),$$

which proves Theorem 19.

5 Showing Universal Optimality

In Section 3, we showed for each point set P a universal lower bound (Theorem 12). This bound uses the quantity $V_{\max}(P)$. This quantity is the maximum m , for which there exists a list $W = (a_i, b_i, c_i)_{i=1}^n \subset \mathbb{Z}^{n \times 3}$ with m inputs $I_P \in \mathbb{I}_P$ for which W is a witness list of I_P .

In Section 4, we showed an upper bound for Algorithm 1 (Theorem 19). For each input I_P , the execution of Algorithm 1 on I_P (together with an edge between the leftmost and rightmost vertex in I_P) defines a *quadrangle tree* \mathcal{Q} . Our upper bound uses the quantity $|\text{OD}(P, \mathcal{Q})|$. This quantity counts the maximum number of ordered downdrafts of (P, \mathcal{Q}) .

Here, we show that Algorithm 1 is universally optimal by correlating these two quantities. We achieve this through two intermediate quantities: the number of *corner assignments* of W and the number of *hull lists* of P .

► **Definition 22.** Let $W = (a_i, b_i, c_i)_{i=1}^n$. A corner assignment picks a corner for each of the n triangles in W . Formally, it is a map $C : [n] \rightarrow [n]$ such that $C(i) \in \{a_i, b_i, c_i\}$ for all $i \in [n]$. Let $\text{CA}(W)$ denote the set of all corner assignments of W . Note that $|\text{CA}(W)| = 3^n$.

► **Definition 23.** We define $\text{HL}(P)$ as the set of all lists of $k = |\text{ch}(P)|$ integers such that $\forall H \in \text{HL}(P)$ there exists an input $I_P \in \mathbb{I}_P$ where H is a hull list of I_P . Note that $|\text{HL}(P)| = \frac{n!}{(n-k)!} \leq n^k$.

► **Lemma 24.** Let W be an arbitrary witness for P . Let \mathcal{Q} be an arbitrary quadrangle tree for P . There exists an injective map

$$\Phi : V(P, W) \hookrightarrow \text{OD}(P, \mathcal{Q}) \times \text{CA}(W) \times \text{HL}(P).$$

Proof. Fix $I_P \in V(P, W)$. By our canonical ordering of the hull lists, there exists exactly one $H \in \text{HL}(P)$ for which H is a hull list of I_P . We define our map Φ by additionally constructing for I_P a corner assignment C and ordered downdraft $\bar{\varphi}$.

For $i \in [n]$, observe that if $I_P[i]$ is not in $\text{ch}(P)$ then $I_P[i]$ lies strictly inside the triangle formed by $(I_P[a_i], I_P[b_i], I_P[c_i])$. Therefore, at least one of $I_P[i] \prec_{\mathcal{Q}} I_P[a_i]$, $I_P[i] \prec_{\mathcal{Q}} I_P[b_i]$ or $I_P[i] \prec_{\mathcal{Q}} I_P[c_i]$ holds. We construct the corner assignment C by arbitrarily choosing for

$$\begin{array}{ccc}
[n] - H & \xrightarrow{I_P} & P - \text{ch}(P) \xleftarrow{I'_P} [n] - H \\
\downarrow C & & \downarrow \varphi \quad \downarrow C \\
[n] & \xrightarrow{I_P} & P \xleftarrow{I'_P} [n]
\end{array}
\qquad
\begin{array}{ccc}
X & \xrightarrow{I_P} & \varphi^{-1}(\{v\}) \xleftarrow{I'_P} X \\
\downarrow C & & \downarrow \varphi \quad \downarrow C \\
\{j\} & \xrightarrow{I_P} & \{v\} \xleftarrow{I'_P} \{j\}
\end{array}$$

■ **Figure 4** (left) the commuting diagram corresponding to (I_P, I'_P, φ, C) . (right) In our proof, we chase this diagram from v encountering the integer j , the set X , and the fiber $\varphi^{-1}(\{v\})$.

each $i \in [n]$, the value $C(i) \in \{a_i, b_i, c_i\}$ such that $I_P[i] \prec_Q I_P[C(i)]$. We then construct a downdraft φ by setting $\varphi(I_P[i]) = I_P[C(i)]$. To obtain an ordered downdraft, we order for all $q \in P$ each fiber $\varphi^{-1}(\{q\})$ via the following rule:

$$\forall I_P[i], I_P[j] \in \varphi^{-1}(\{q\}) : \quad I_P[i] \prec_{\bar{\varphi}} I_P[j] \quad \Leftrightarrow \quad i < j$$

Then $\bar{\varphi}$ is an ordered downdraft and we have defined the map $\Phi(I_P) = (\bar{\varphi}, C, H)$.

We now show that this map Φ is injective. Suppose that I_P and I'_P are elements of $V(P, W)$ such that $\Phi(I_P) = (\bar{\varphi}, C, H) = \Phi(I'_P)$. Observe that, per definition, the hull list $H \in \Phi(I_P) = \Phi(I'_P)$ is a sequence of integers with the following property: if u is the x 'th point on $\text{ch}(P)$ then $I_P[H[x]] = I'_P[H[x]] = u$. Suppose for the sake of contradiction that there exist integers $i, i' \in [n]$ with $i \neq i'$ such that $u = I_P[i] = I'_P[i']$. We pick the pair (i, i') such that u is maximal in the partial order \prec_Q .

First, consider the case where u is a point on $\text{ch}(P)$. Let u be the x 'th element in the canonical ordering of $\text{ch}(P)$. Then $u = I_P[H[x]] = I'_P[H[x]]$. However, we chose u such that $u = I_P[i] = I'_P[i']$ and so $i = i'$, a contradiction.

Alternatively, suppose that $u \notin \text{ch}(P)$. By maximality of our choice of u , it follows that for all points v' with $u \prec_Q v'$, there exists a unique integer j' such that $v' = I_P[j'] = I'_P[j']$. We define $v := \varphi(u)$. By the definition of a downdraft, $u \prec_Q v$ and so there is a unique integer $j \in [n]$ with $v = \varphi(u) = I_P[j] = I'_P[j]$.

We observe that the objects I_P, I'_P, φ and C are all maps. In Figure 4 we illustrate the diagram corresponding to these maps. Recall that H is the hull list of both $\Phi(I_P)$ and $\Phi(I'_P)$. The left diagram in Figure 4 commutes. Indeed, by our definition of the downdraft φ , we chose φ such that $\varphi(I_P[x]) = I_P[C(x)]$ for all $x \in [n] \setminus H$.

Consider chasing the left square of this diagram from $v = \varphi(u)$ (see Figure 4, right). From v , we may first apply I_P^{-1} to obtain j . We then apply C^{-1} to j to get a set of indices $X := C^{-1}(I_P^{-1}(\{v\}))$. Conversely, we may chase this diagram from v by first applying φ^{-1} to obtain a set of points, and then applying I_P^{-1} to obtain a set of indices. Since the diagram commutes, also $I_P^{-1}(\varphi^{-1}(\{v\})) = X$. Next, we focus on the right square of the diagram. Since $v = I'_P[j]$, we have $C^{-1}(I_P^{-1}(\{v\})) = X$. We may chase the right square from v by first applying φ^{-1} to obtain a set of points, and then applying $I_P'^{-1}$. Since this diagram commutes, it follows that $I_P'^{-1}(\varphi^{-1}(\{v\})) = X$ too. We now use this set X to obtain a contradiction:

We assumed that $(\bar{\varphi}, C, H) = \Phi(I_P) = \Phi(I'_P)$. Recall that $\prec_{\bar{\varphi}}$ induces a total order on $\varphi^{-1}(\{v\})$. Per definition of $\Phi(I_P)$, all points $p \in \varphi^{-1}(\{v\})$ are ordered by their index x defined via $p = I_P[x]$. The rank of a value u with respect to an ordered set is its index in the order. We can consider the rank of u in two sets: X and $\varphi^{-1}(\{v\})$. Recall that we ordered for all $q \in P$ each fiber $\varphi^{-1}(\{q\})$ via the following rule:

$$\forall I_P[a], I_P[b] \in \varphi^{-1}(\{q\}) : \quad I_P[a] \prec_{\bar{\varphi}} I_P[b] \quad \Leftrightarrow \quad a < b$$

It follows that the rank of u satisfies:

$$\text{Rank}_{\prec_{\bar{\varphi}}} \left(u, \varphi^{-1}(\{v\}) \right) = \text{Rank}_{<} \left(i, I_P^{-1}(\varphi^{-1}(\{v\})) \right) = \text{Rank}_{<} \left(i, X \right)$$

Similarly,

$$\text{Rank}_{\prec_{\varphi}}(u, \varphi^{-1}(\{\varphi(u)\})) = \text{Rank}_{\prec}(i', C^{-1}(\{j\})) = \text{Rank}_{\prec}(i', X).$$

Since i and i' both have the same rank in X , it follows that $i = i'$, contradiction. This shows that $I_P = I'_P$, so our constructed map Φ is injective. \blacktriangleleft

The above lemma immediately implies the following relation between $V_{\max}(P)$ and $\text{OD}(P, \mathcal{Q})$:

► **Corollary 25.** *For every quadrangle tree \mathcal{Q} for P ,*

$$V_{\max} \leq |\text{OD}(P, \mathcal{Q})| \cdot 3^n \cdot n^k.$$

Finally, we are ready to show universal optimality of Algorithm 1:

► **Theorem 10.** *If we define \mathcal{A} as the set of all correct comparison-based convex hull algorithms (Definition 5) then the algorithm in [7] for $d = 2$ is universally optimal.*

Proof. Let \mathcal{Q} denote the quadrangle tree that corresponds to executing Algorithm 1. We apply Theorem 19 to note that there exists a constant c such that:

$$\max_{I_P \in \mathbb{I}_P} \rho(A, I_P) \leq c \cdot \left(n + \log \frac{n^n}{|\text{OD}(P, \mathcal{Q})|} \right).$$

By Corollary 25,

$$\log \frac{n^n}{|\text{OD}(P, \mathcal{Q})|} \leq \log \frac{3^n \cdot n^k \cdot n^n}{V_{\max}} = n \log(3) + k \log n + \log \frac{n^n}{V_{\max}}$$

Since $\log(3) \leq 2$ and $\log n^n \leq 2n + \log n!$, this yields

$$\log \frac{n^n}{|\text{OD}(P, \mathcal{Q})|} \leq 4n + k \log n + \log \frac{n!}{V_{\max}}$$

We may now apply the universal lower bound from Theorem 12 to obtain the theorem. \blacktriangleleft

References

- 1 Peyman Afshani, Jérémy Barbay, and Timothy Chan. Instance-optimal geometric algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 129–138. Association for Computing Machinery, 2009. doi:10.1109/FOCS.2009.34.
- 2 Peyman Afshani, Jérémy Barbay, and Timothy M. Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1), March 2017. doi:10.1145/3046673.
- 3 Mark de Berg, Matthew Katz, A Frank Van der Stappen, and Jules Vleugels. Realistic input models for geometric algorithms. In *Symposium on Computational geometry (SoCG)*, pages 294–303, 1997. doi:10.1145/262839.262986.
- 4 Jeff Erickson. Dense point sets have sparse delaunay triangulations or “... but not too nasty”. *Discrete & Computational Geometry*, 33:83–115, 2005. doi:10.5555/3115459.3115691.
- 5 Bernhard Haeupler, Richard Hladík, Václav Rozhoň, Robert E Tarjan, and Jakub Tetěk. Universal optimality of dijkstra via beyond-worst-case heaps. In *Symposium on Foundations of Computer Science (FOCS)*, pages 2099–2130. IEEE, 2024. doi:10.1109/FOCS61266.2024.00125.
- 6 Bernhard Haeupler, Richard Hladík, John Iacono, Václav Rozhoň, Robert E. Tarjan, and Jakub Tetěk. Fast and simple sorting using partial information. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3953–3973, 2025. doi:10.1137/1.9781611978322.134.

- 7 David G Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM Journal on Computing*, 15(1):287–299, 1986. doi:10.1137/0215021.
- 8 Jiří Matoušek, János Pach, Micha Sharir, Shmuel Sifrony, and Emo Welzl. Fat triangles determine linearly many holes. *SIAM Journal on Computing*, 23(1):154–169, 1994. doi:10.1109/SFCS.1991.185347.
- 9 Ivor van der Hoog, Eva Rotenberg, and Daniel Rutschmann. Simpler optimal sorting from a directed acyclic graph. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 350–355. SIAM, 2025. doi:10.1137/1.9781611978315.26.