Algorithmica
DOI 10.1007/s00453-008-9174-2

Largest and Smallest Convex Hulls for Imprecise Points

Maarten Löffler · Marc van Kreveld

Received: 11 July 2006 / Accepted: 12 February 2008

Abstract Assume that a set of imprecise points is given, where each point is specified by a region in which the point may lie. We study the problem of computing the smallest and largest possible convex hulls, measured by length and by area. Generally we assume the imprecision region to be a square, but we discuss the case where it is a segment or circle as well. We give polynomial time algorithms for several variants of this problem, ranging in running time from $O(n \log n)$ to $O(n^{13})$, and prove NP-hardness for some other variants.

Keywords Computational geometry · Imprecision · Data imprecision · Convex hulls

1 Introduction

In computational geometry, many fundamental problems take a point set as input on which some computation is done, for example to determine the convex hull, the Voronoi diagram, or a traveling sales route. These problems have been studied for decades. The vast majority of research assumes the locations of the input points to be known exactly. In practice, however, this is often not the case. Coordinates of the points may have been obtained from the real world, using equipment that has some error interval, or they may have been stored as floating points with a limited number of decimals. In real applications, it is important to be able to deal with such imprecise points.

This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) under BRICKS/FOCUS grant number 642.065.503 and under the open competition project GOGO.

M. Löffler (✉) · M. van Kreveld
Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
e-mail: loffler@cs.uu.nl

M. van Kreveld
e-mail: marc@cs.uu.nl

When considering imprecise points, various interesting questions arise. Sometimes it is sufficient to know just a possible solution, which can be achieved by just applying existing algorithms to some point set that is possibly the true point set. More information about the outcome can be obtained by computing a probability distribution over all possibilities, for example using Monte Carlo methods and a probability distribution over the input points. In many applications it is also important to know concrete lower and upper bounds on some measure on the outcome, given concrete bounds on the input: every point is known to be somewhere in a prescribed region.

1.1 Related Work

A lot of research about imprecision in computational geometry is directed at computational imprecision rather than data imprecision. Regarding data imprecision, there is a fair amount of work that uses stochastic or fuzzy models of imprecision. Alternatively, an exact model of imprecision can be used.

Nagai and Tokura [22] compute the union and intersection of all possible convex hulls to obtain bounds on any possible solution. As imprecision regions they use circles and convex polygons, and they give an $O(n \log n)$ time algorithm. They also study the Minkowski sum of convex polygons and the diameter of a point set. Ostrovsky-Berman and Joskowicz [25] study the union of all possible convex hulls when the imprecision of the points is not independent, but the points depend linearly on a limited number of parameters.

Espilon Geometry is a framework for robust computations on imprecise points. Guibas *et al.* [15] define the notion of *strongly convex* polygons: polygons that are certain to remain convex even if the input points are perturbed within a disc of radius ε. They define an *ε-convex δ-hull* of a point set P to be a polygon with points of P as vertices that is convex even when the points move over a distance ε, yet has all vertices of the convex hull of P at most δ away from its boundary. They show that such a hull always exists when $\delta \geq 2\varepsilon$, and give an $O(n^3 \log n)$ algorithm to compute it. Related results are given in [4, 9, 18].

Abellanas *et al.* [1] define the *tolerance* of a geometric structure as the largest perturbation of the vertices such that the topology of the structure is guaranteed to stay the same. They focus mainly on the planar Delaunay triangulation, and show that its tolerance can be computed in linear time. They also study several subgraphs of the Delaunay triangulation.

On the other hand, Bandyopadhyay and Snoeyink [2] study the possible changes in topology for a fixed maximum perturbation ε. A triangle (or simplex in higher dimensions) with vertices in some point set is called *almost Delaunay* when a perturbation of the set of at most ε exists, such that the circumcircle of the perturbed triangle does not contain any other points. They show applications to the problem of folding proteins.

Khanban and Edalat [16] want to compute the Delaunay triangulation of a set of imprecise points, modeled as rectangles. They do this by defining the *in-circle test*, a test that decides whether a point is inside the circle through three other points, on imprecise points.

Boissonnat and Lazard [5] study the problem of finding the shortest convex hull of bounded curvature that contains a set of points, and they show that this is equivalent

to finding the shortest convex hull of a set of imprecise points modeled as circles that have the specified curvature (see also Sect. 2.2). They give a polynomial time approximation algorithm.

Goodrich and Snoeyink [14] study a problem where they are given a set of parallel line segments, and must choose a point on each segment such that the resulting point set is in convex position. This can be seen as a convexity test for points with one-dimensional imprecision. They present an algorithm that finds a solution, if it exists, in $O(n \log n)$ time. They also show how to minimize the area or perimeter of the polygon in $O(n^2)$ time.

The problem of finding the shortest tour for a set of imprecise points when the order is not fixed, has been studied before and is generally called the Traveling Salesman Problem with Neighborhoods, or (Planar) Group-TSP. This problem is known to be NP-hard. Mata and Mitchell [20] give a constant factor approximation algorithm for some region models; additional results can be found in [7, 27].

Given a sequence of k polygons with a total of n vertices, Dror *et al.* [10] study the problem of finding a tour that touches all of them in a given order and that is as short as possible. They give an $O(nk \log(n/k))$ algorithm when the input polygons are disjoint and convex, and prove that the problem is NP-hard for non-convex polygons. Higher dimensions are considered in [26].

Fiala *et al.* [12] consider the problem of finding distant representatives in a collection of subsets of a given space. Translated to our setting, they prove that maximizing the smallest distance in a set of n imprecise points, modeled as circles or squares, is NP-hard. Finally, we mention de Berg *et al.* [8] for a problem with data imprecision motivated from computational metrology, Cai and Keil [6] for visibility in an imprecise simple polygon, Sellen *et al.* [28] for precision sensitivity, and Yap [30] for a survey on robustness, which deals with computational imprecision rather than data imprecision.

The smallest possible convex hull of a set of imprecise points coincides with the notion of a *polygon transversal*: the smallest (convex) polygon intersecting a set of regions. Mukhopadhyay *et al.* [21] compute the smallest area convex polygon that intersects a set of parallel line segments in $O(n \log n)$ time, while Rappaport [23] computes the smallest perimeter polygon transversal of a set line segments in a constant number of orientations, also in $O(n \log n)$ time.

1.2 Problem Definition

All in all there has been little structured research into concrete bounds on the possible outcomes of geometric problems in the presence of data imprecision. When placing a traditional problem that computes some structure on a set of points in this context, two important questions arise:

1. What are imprecise points? That is, what are the restrictions on the input of the problem?
2. What are bounds on the outcome? That is, what kind of restrictions on the output of the problem do we want to infer from this?

The first question is what we are given. We model imprecise points by requiring the points to be inside some fixed region, without any assumption on where exactly in

their regions the points are, but with absolute certainty that they are not outside their regions. The question then arises what shape these regions should be given. Some natural choices are square and circular regions (or unit balls in the L_1 and L_2 metric). The square model for example occurs when points have been stored as floating point numbers, where both the x and y coordinates have an independent uncertainty interval, or with raster to vector conversion. The circular model occurs when the point coordinates have been determined by a scanner or by GPS, for example. Other models that may be interesting include the line segment model, the rectangle model, the regular k-gon model, the discrete point set model, or the Voronoi model (where the cells are the imprecision regions), mostly from a theoretical point of view. Another question is what kind of restrictions we impose on those regions. For example, all points can have the same kind of shape, but are they all of the same size? Do they have the same orientation? Can we assume they are disjoint?

The second question is what we actually want to know. Geometric problems usually output some complex structure, not just a number, so a measure on this structure is needed. For example, the convex hull of a set of points can be measured by area or perimeter, or maybe even other measures in some applications. Once a measure has been established, the question is whether an upper or a lower bound is desired, or both.

1.3 Results

All these questions together lead to a large class of problems that are all closely related to each other. This paper aims to find out how exactly they are related, which variants are easy and which are hard to compute, and to provide algorithms for the problems that can be solved in polynomial time. Since this type of problem has hardly been studied, we consider the classical planar convex hull problem.

We studied various variants of this problem, and our results are summarized in Table 1. These results are treated in detail in Sects. 3, 4 and 5. First, in the next section, some related issues are discussed.

2 Preliminaries

Before the main results are treated, we discuss some difficulties that occur when dealing with imprecise points. First we look at the Euclidean Minimum Spanning Tree problem for imprecise points, and then we take a closer look at the circular region model for imprecision.

2.1 Minimum Spanning Tree

To get an idea of how imprecision affects the complexity of geometric problems, consider the Minimum Spanning Tree (MST) problem in an imprecise context. In this case, we have a collection of imprecise points, and we want to determine the MST of, for example, minimal length. This means that we want to choose the points in such a way that the MST of the resulting point set is as small as possible. This problem is

Table 1 An overview of the complexity of the various variants. Two of the bounds are already known

Goal	Measure	Model	Restrictions	Running time
Largest	Area	Line segments	Parallel	$O(n^3)$
Largest	Area	Line segments	Non-intersecting, convex position[1]	$O(n^3)$
Largest	Area	Line segments	–	NP-hard[2]
Largest	Area	Squares	Non-intersecting	$O(n^7)$
Largest	Area	Squares	Non-intersecting, equal size	$O(n^3)$
Largest	Area	Squares	Equal size	$O(n^5)$
Largest	Perimeter	Line segments	Parallel	$O(n^5)$
Largest	Perimeter	Line segments	–	NP-hard[2]
Largest	Perimeter	Squares	Non-intersecting	$O(n^{10})$
Largest	Perimeter	Squares	Equal size	$O(n^{13})$
Smallest	Area	Line segments	Parallel	$O(n \log n)$ [21]
Smallest	Area	Squares	–	$O(n^2)$
Smallest	Perimeter	Line segments	Parallel	$O(n \log n)$ [23]
Smallest	Perimeter	Squares	–	$O(n \log n)$

[1]The 'convex position' restriction means that the endpoint of the input segments are in convex position

[2]The decision version of the first NP-hard problem is NP-complete, but not of the second one

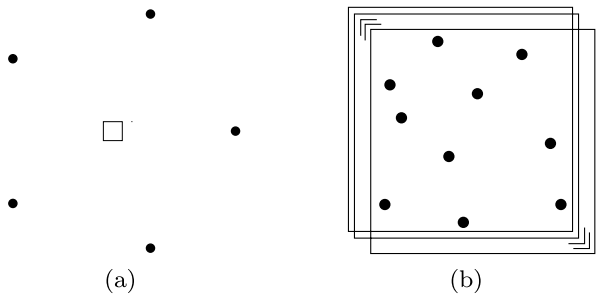

Fig. 1 (**a**) It is algebraically difficult to find the minimal MST. (**b**) It is combinatorially difficult to find the minimal MST

difficult in two different senses. It is combinatorially difficult to find the structure of the optimal solution, but even when we know the structure it is algebraically difficult to find the exact locations of the points.

Consider the input in Fig. 1a. It consists of five fixed points and one imprecise point (in the square model, but it could also be a circle or something else). No matter where the point is chosen in this square, the MST of the resulting set will connect all of the fixed points to the imprecise point. Thus the problem reduces to minimizing the sum of the distances from the imprecise point to the fixed points, and this requires finding roots of high degree polynomials, which is an algebraically difficult problem [3].

But even when we disregard the algebraic problems, the problem is still difficult. We can prove that it is NP-hard by reduction from the Steiner Minimal Tree prob-

lem [13]. Given a set of n fixed points P in the plane, we can compute its Steiner Tree using a solution to the imprecise MST problem as follows. Take the set P as precise points, and add a set P' of $n-2$ imprecise points whose regions are squares or circles that contain P, see Fig. 1b. The shortest MST of $P \cup P'$ is the Steiner Minimal Tree of P.

2.2 Circular Model

Perhaps the most natural way of modeling imprecision is by allowing every point to be inside a circular region. The convex hull problem then becomes:

Problem 1 *Given a set of circles, choose a point inside each circle such that the area/perimeter of the convex hull of the resulting point set is as large/small as possible* (*see Fig.* 2).

Two difficulties are introduced by using circular regions. The first difficulty is that the combinatorial complexity of the problem increases. In the case of the square model we can use the notion of extreme points in some directions. With circles this is not possible since there are no special directions any more. The second difficulty is of an algebraic kind. Even when we know which circles have to be chosen to obtain the largest/smallest area/perimeter, it is not easy to find out where exactly in the circles the points should be.

One special case of this problem has been studied before. For the problem of finding the smallest perimeter for a set of unit size circles, Boissonnat and Lazard [5] show that this problem can be approximated in polynomial time. The question of whether it can be solved exactly in polynomial time is left open, and has to our knowledge not yet been answered. The same problem for smallest area is also stated as an open problem in [5].

One remark to make here is that given the algebraic complexity of the problem, one could argue that an exact solution cannot be computed. For example in the case of the smallest perimeter, even in a simple situation with only three circles, the coordinates of the optimal points within the circles will generally be roots of some polynomials of degree six. These roots cannot be computed exactly, only approximated. With this idea in mind, one could say that an approximation is the best we can get in any case, and therefore a good polynomial time approximation is a good solution.

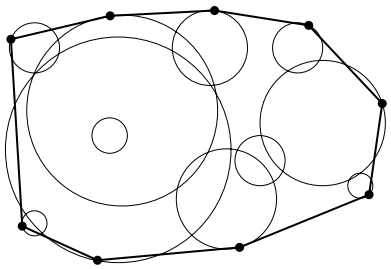

Fig. 2 The largest area convex hull for a set of circles

3 Largest Convex Hull

We now present our results on the imprecise convex hull problem. This section deals with computing the largest possible convex hull; the smallest convex hull is treated in the next section. We first use the line segment model, in which every point can be anywhere on a line segment. This problem does not have much practical use, but it will be extended to the square model later.

3.1 Line Segments

The problem we discuss in this section is the following:

Problem 2 *Given a set of parallel line segments, choose a point on each line segment such that the area of the convex hull of the resulting point set is as large as possible* (*see Fig.* 3*a*).

3.1.1 Observations

First we will show that we can ignore the interiors of the segments in this problem, that is, we only have to consider the endpoints.

Lemma 1 *There is an optimal solution to Problem* 2 *such that all points are chosen at endpoints of the line segments.*

Proof Suppose there exists a set of points P that has one point on every segment, has maximal area, and a minimal number of points that are not at an endpoint of their segments, and yet contains a point p that is not at an endpoint of its segment. If p is not a vertex of the convex hull, just move it to one of the endpoints of its segment. The new convex hull will be of equal or larger size, contradicting the choice of P.

If p is a vertex of the convex hull, and we move it over its segment, the area of the polygon changes as a linear function, if we maintain the combinatorial structure of the original hull. The maximum of this function is at one of the endpoints. Move p to this endpoint, and the area of the polygon increases. It is possible that the polygon

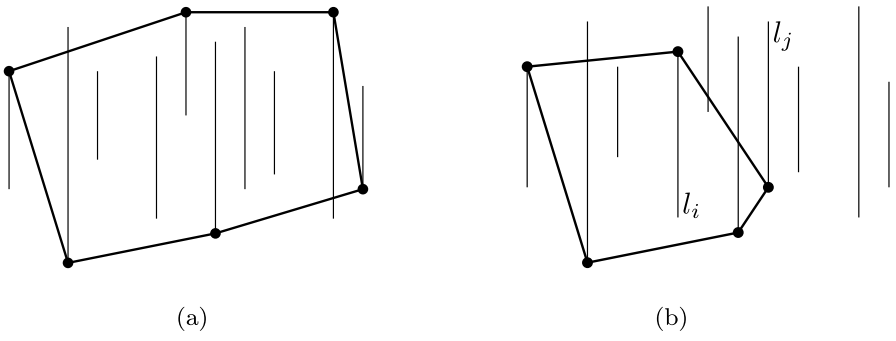

Fig. 3 (**a**) The largest convex hull for a set of line segments. (**b**) The polygon P_{ij}

is no longer convex or that some points of P no longer lie within the polygon, but correcting this can only increase the area of the convex hull further. Once again we have a contradiction with the choice of P. We conclude that P does not exist, and the lemma is proven. □

Note that the lemma does not make use of the restriction that the segments are parallel, and also applies to general sets of line segments. From now on however, we do enforce this restriction. Without loss of generality, we assume the segments to be oriented vertically.

3.1.2 Algorithm

Let $L = \{l_1, l_2, \ldots, l_n\}$ be a set of n line segments, where l_i lies to the left of l_j if $i < j$. Let l_i^+ denote the upper endpoint of l_i, and l_i^- denote the lower endpoint of l_i. Now we need to pick one of each pair of endpoints to determine the largest area convex hull. We use a dynamic programming algorithm that runs in $O(n^3)$ time and $O(n^2)$ space. The key element of this algorithm is a polygon which is defined for each pair of line segments.

For $i \neq j$, define the polygon P_{ij} as the largest possible polygon that is the convex hull of some choice of endpoints to the left of l_i and l_j, and uses the top of l_i and the bottom of l_j, see Fig. 3b. In other words, it is the convex hull of a set of endpoints l_k^+ for some values $k \leq i$, and endpoints l_k^- for some values $k \leq j$, where not both l_k^+ and l_k^- can occur for the same k, such that the area of this convex hull is maximized. Note that P_{ij} is defined both for the case $i < j$ and $i > j$.

We consider the polygon P_{ij} that starts at l_i^+ and ends at l_j^-, and optimally solves the subproblem to the left of these points, that is, contains only vertices l_k^+ with $k < i$ or l_k^- with $k < j$, but not both for the same k, such that the area of the polygon is maximal, see Fig. 3b. Note that P_{ij} will be convex.

Now, we will show how to compute all P_{ij} using dynamic programming. The solution to the original problem will be either of the form P_{kn} or P_{nk} for some $0 < k < n$, and can thus be computed in linear time once all P_{ij} are known.

When $1 < i < j$, then we can write

$$P_{ij} = \max_{k<j;k\neq i} \left(P_{ik} + \Delta l_i^+ l_j^- l_k^- \right)$$

Of course we maximize over the area of the polygons. In words, we choose one of the lower points to the left of l_j, and add the new point l_j^- to the polygon P_{ik} that optimally solves everything to the left of the chosen point l_k^-. Analogously, when $1 < j < i$, we can write

$$P_{ij} = \max_{k<i;k\neq j} \left(P_{kj} + \Delta l_i^+ l_j^- l_k^+ \right)$$

When $i = 1$ or $j = 1$, we can use the same formulas to compute P_{ij}, but we need the additional option to just choose the line segment $\overline{l_i^+ l_j^-}$ with area 0, in case there are no more points to the left of the new one.

The algorithm runs in $O(n^3)$ time and requires $O(n^2)$ space. This is because we do not have to actually store the entire polygon P_{ij} for each i and j, but only the next point on the upper chain when $i > j$ or the lower chain when $i < j$, and the area of the polygon. When we scan the known polygons while determining a new one, we only have to add the area of a triangle to the stored area, and take the maximum of those numbers. We do not need to enforce convexity, because a non-convex solution can never be optimal.

Theorem 1 *Given a set of n arbitrarily sized, parallel line segments, the problem of choosing a point on each segment such that the area of the convex hull of the resulting point set is as large as possible can be solved in* $O(n^3)$ *time.*

3.1.3 Arbitrary Orientations

The above algorithm works for parallel line segments. When the line segments are allowed to have arbitrary orientations, the most general version of the problem, where segments are allowed to intersect, becomes NP-hard, and the decision version NP-complete. We prove this by reduction from SAT. Given an instance of SAT, we make the following construction.

We start with a large circle, and divide it into enough arcs, that is, at least as many as the number of variables plus the number of clauses in the SAT instance, see Fig. 4a. The arcs do not need to have the same length. We separate these arcs by precise points (degenerate line segments). The solution will contain at least the convex hull of these precise points. We will make sure never to place any (parts of) line segments outside this circle, so maximizing the area of the convex hull is now equal to maximizing the sum of the areas within the arcs.

For each Boolean variable b in the SAT instance, we take an empty arc and add the configuration of Fig. 4b inside. This configuration consists of two precise points l and r that were already added, a segment parallel to $\overline{lr}$ with endpoints t and f, and two sets of points P_b and Q_b. The points of P_b are placed so that they are all on the convex hull of $\{l, f, r\} \cup P_b \cup Q_b$, but none is on the convex hull of $\{l, t, r\} \cup P_b \cup Q_b$, and the points of Q_b are placed so that they are all on the convex hull of $\{l, t, r\} \cup P_b \cup Q_b$, but none is on the convex hull of $\{l, f, r\} \cup P_b \cup Q_b$. The whole configuration is symmetric by design. The idea is that to maximize the area within this configuration, we either need t and all points in Q_b, or f and all points

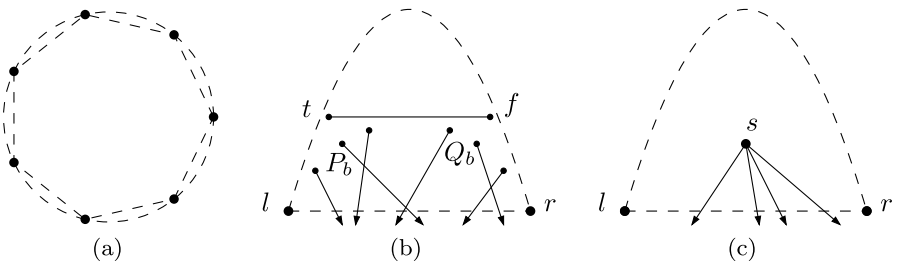

Fig. 4 (**a**) The division of the circle into independent arcs. (**b**) A variable. (**c**) A clause

in P_b. The first case represents the value `true` for this variable, and the second case represents the value `false`. The points in P_b and Q_b will have their other endpoints in the clauses, or if they are only present to achieve symmetry they are simply precise points.

For each clause in the SAT instance, we also take an empty arc, and add just a single point s in it, see Fig. 4c. Now we make s the other endpoint of one segment from each variable that occurs in this clause. For example, if the clause is $a \vee b \vee \neg c$, then we make s the other endpoint of one of the points in P_a, one of the points in P_b, and one of the points in Q_c. For the area to be maximal, of at least one of these three segments the point must be chosen in s, which is only possible when a is `true`, b is `true` or c is `false`.

Let A^* be the area of the convex hull of the set of points that contains the fixed points, all clause points s, and within every variable configuration the point t and the point set Q. Now an assignment to the variables to satisfy the SAT instance can be made if and only if a solution to the convex hull maximization problem of area A^* exists.

It is well known that rational points are dense on the unit circle, and we can construct m points that are all at least $\frac{\pi}{m}$ radians apart with coordinates dat depend quadratically on m. Between two such fixed points l and r, we make a symmetric construction with points on a grid parallel to $\overline{lr}$. If the variable is used k times, this grid needs $2k$ cells in the lr direction, and k^2 cells in the perpendicular direction. If we place the grid in a rectangle of width half the length of $\overline{lr}$, and height $\frac{1}{2m}$ times the length of $\overline{lr}$, then the variables do not interfere with each other. Thus all constructed points are rational points of polynomial complexity. This analysis shows that the decision problem is in NP.

Theorem 2 *Given a set of n arbitrarily oriented, possibly intersecting line segments, the problem of choosing a point on each segment such that the area of the convex hull of the resulting point set is as large as possible is NP-hard. The decision version of the problem is NP-complete.*

3.1.4 All Endpoints in Convex Position

The status of the problem for arbitrarily oriented line segments that do not intersect is still open. There is, however, another special situation that we can solve. If the endpoints of the input line segments are in convex position, and the segments do not intersect, we can also solve the problem in $O(n^3)$ time. An example of such a set of line segments is shown in Fig. 5a. Because the points are in convex position, there is a cyclic ordering on them that we can use.

To solve the problem in this case, we also use a dynamic programming approach. Let p and q be endpoints of different line segments, and let p' and q' be their respective other ends. We define P_{pq} as the chain that connects p to q in positive (counterclockwise) direction, such that the area of the region enclosed by the chain and $\overline{pq}$ is maximal over all valid chains that connect p to q, see Fig. 5b. A chain is valid if it does not contain both ends of any input line segment. When p' is between p and q (in positive direction), we also define P'_{pq} as the chain that connects p to q

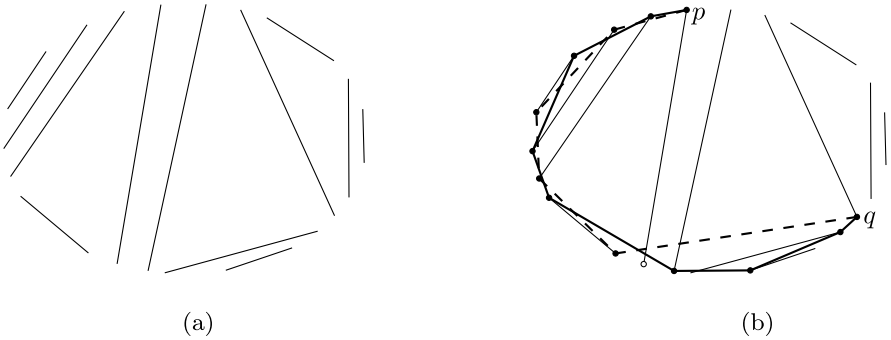

Fig. 5 (**a**) A set of line segments whose endpoints are in convex position. (**b**) The polygons P_{pq} (*solid*) and P'_{pq} (*dashed*)

in positive direction and maximizes the area enclosed by it, but is not allowed to use any points between p' and q. With a slight abuse of notation, we use P_{pq} and P'_{pq} both for the chains and for the areas of their corresponding polygons.

If we know all P_{pq}, then we can solve the problem in $O(n^2)$ time, since the optimal solution will be of the form $P_{pq} + \overline{qp}$ for some p and q. To compute all P_{pq}, we find the following recursive relations between the P and P' values.

Let p and q be endpoints of different line segments. If there are no points between them, then $P_{pq} = \overline{pq}$. Else, if p' is not between p and q, then there exists a point r between p and q such that P_{pq} is $\overline{pr} + P_{rq}$. If p' is between p and q, then either $P_{pq} = P'_{pq}$, or there is a point r between p' and q such that $P_{pq} = P'_{pr} + P_{rq}$.

If P'_{pq} is defined, we know that p' is between p and q. If there are no points between p and p', then $P'_{pq} = \overline{pq}$. Else, there exists a point r between p and p', such that $P'_{pq} = P_{pr} + \overline{rq}$.

In all cases, we have written P_{pq} and P'_{pq} in terms of shorter chains and at most one variable point. Since there are a quadratic number of such chains, we can compute them all in $O(n^3)$ time and $O(n^2)$ space.

Theorem 3 *Given a set of n arbitrarily sized, arbitrarily oriented, non-intersecting line segments with their endpoints in convex position, the problem of choosing a point on each segment such that the area of the convex hull of the resulting point set is as large as possible can be solved in* $O(n^3)$ *time.*

3.2 Squares

The problem we discuss in this section is the following:

Problem 3 *Given a set of axis-aligned squares, choose a point in each square such that the area of the convex hull of the resulting point set is as large as possible* (*see Fig. 6a*).

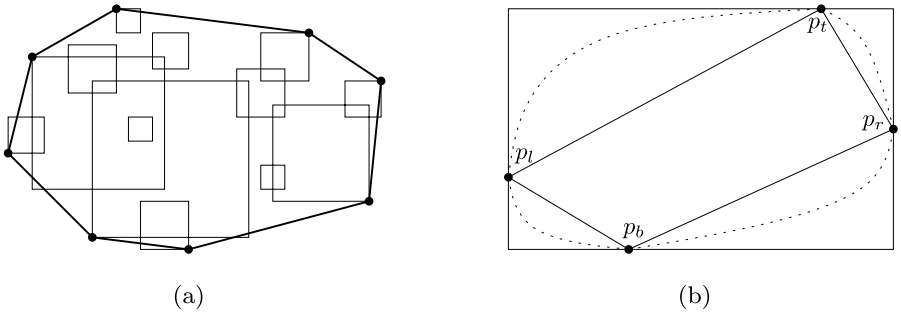

Fig. 6 (**a**) The largest area convex hull for a set of squares. (**b**) The four extreme points

3.2.1 Observations

Once again we observe that the points will never have to be chosen in the interior of the squares. In fact we only have to take the corners of the squares into account.

Lemma 2 *There is an optimal solution where all points lie at a corner of their square.*

Proof Suppose there exists a set of points P that has one point in every square, has maximal area, and a minimal number of points that are not at a corner of their squares, and yet contains a point p that is not at a corner of its square. If p is not a vertex of the convex hull, just move it to one of the corners of its square. The new convex hull will be of equal or larger size, contradicting the choice of P.

If p is a vertex of the convex hull, and we move it around in its square, the area of the polygon changes as a linear function in the coordinates of p as we maintain the combinatorial structure. The maximum of this function is at one of the corners. Move p to this corner, and the area of the polygon increases. It is possible that the polygon is no longer convex or that some points of P no longer lie within the polygon, but correcting this can only increase the area of the polygon further. Once again we have a contradiction with the choice of P. We conclude that P does not exist, and the lemma is proven. □

First we define the four *extreme* points of the convex hull we are trying to compute as the leftmost, topmost, rightmost and bottommost points. These points divide the hull into four chains that connect them. These chains have some useful properties. For example, the chain that connects the leftmost point p_l to the topmost point p_t will always stay within the triangle $\Delta p_l p_t s$, where s is the intersection between the vertical line through p_l and the horizontal line through p_t. The extreme points and the triangles that surround the four chains are shown in Fig. 6b.

Lemma 3 *All vertices on the top left chain are top left corners of their squares, all vertices on the top right chain are top right corners of their squares, all vertices on the bottom left chain are bottom left corners of their squares, and all vertices on the bottom right chain are bottom right corners of their squares.*

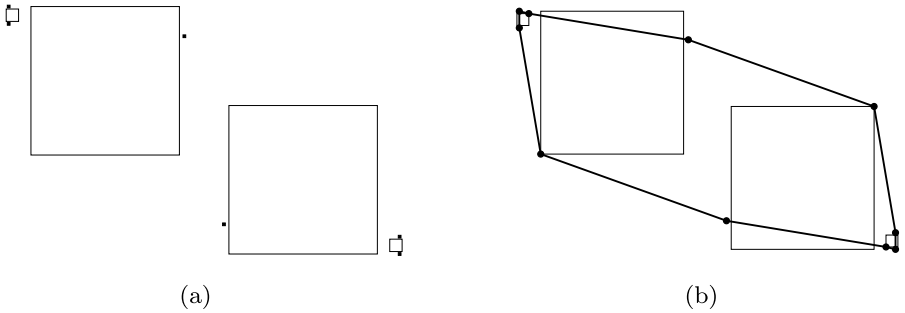

Fig. 7 The four extreme points of the optimal solution need not be corners of the extreme squares. (**a**) Ten input squares. (**b**) The optimal solution

Proof All vertices on the top left chain will have the outside of the hull above them and to their left. This means they have to be top left corners of their squares, because otherwise we could move the point to the left or upwards and the area of the convex hull would increase. Similar arguments apply to the other three chains. □

In general it is not easy to find the extreme points. For example, it could be that none of the extreme points in the optimal solution is in one of the extreme squares in the input, see for example Fig. 7. Here the topmost and bottommost squares are the large ones, and the leftmost and rightmost squares are the medium ones. However, in the optimal solution the extreme points will all be corners of the small squares.

3.2.2 Algorithm for Non-overlapping Squares

When we restrict the problem to non-overlapping squares, we can solve the problem in $O(n^7)$ time. The idea behind the solution is to divide the squares into groups of squares of which we know that only two of their corners are feasible for an optimal solution, and then to use the algorithm for parallel line segments (Problem 2) on these groups.

When the four extreme points are known, we can use this information to solve the problem in $O(n^3)$ time. However, how to find those points still remains a difficult problem, so we try all possible combinations, hence the total of $O(n^7)$.

The four extreme points p_l, p_t, p_r and p_b divide the plane as shown in Fig. 8. From p_l we draw a line to the right, from p_b one upwards, from p_r one to the left and from p_t one downwards. These four lines intersect at four intersection points. For a square to be able to have its point on the top left chain, its upper left corner needs to be in the rectangle between p_l and p_t (actually even in the upper left triangle). An analogous property holds for the other chains. If a square has the potential to be included on more than two chains, this means that it must have at least one of the four intersection points in its interior. Since the squares do not overlap, there can be at most four such squares. Of these squares we simply try every possible combination of corners, of which there are only constantly many, so we can assume from now on that every square has at most two potential corners.

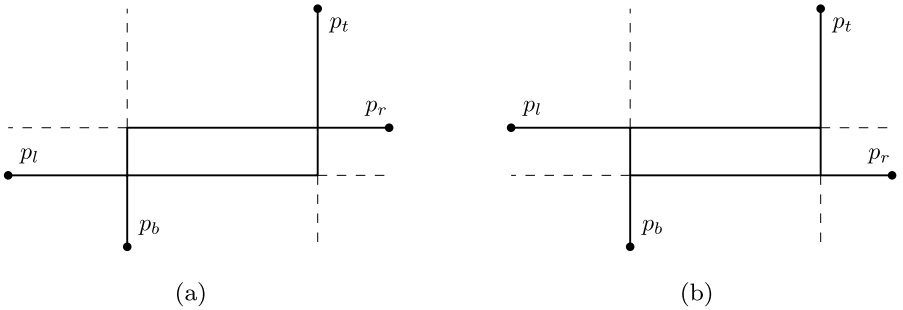

Fig. 8 The four extreme points can divide the plane in two different ways

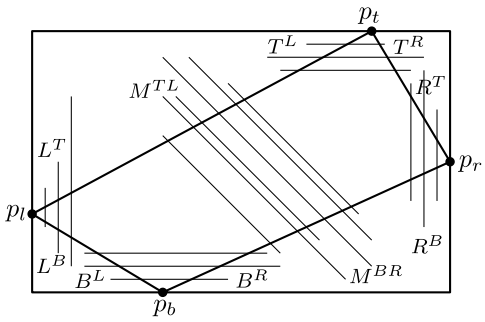

Fig. 9 The squares can be divided into five groups of parallel line segments

Now that all squares have only two potential corners, we can represent them by line segments. We see that a segment can be of six possible kinds, as there are six ways of picking two of four points. These six kinds may however have only four orientations: horizontal, vertical or one of the two diagonal directions. In fact, not all four can appear at the same time in our problem. Indeed, a diagonal line segment has to have its endpoints in two opposite triangles (see Fig. 9). This means that if there are line segments in both diagonal directions, they have to intersect, and thus their original squares have to overlap, which was not allowed. Therefore, there can only be diagonal segments in one direction.

Furthermore, since all line segments have to reach over the quadrilateral $\Diamond p_l p_t p_r p_b$, line segments of the same kind have to be close to each other, that is, their intersection intervals have to be consecutive. There are six possible kinds of line segments, of which we have seen that only five may appear at the same time, which implies that we can divide the segments into five groups, as shown in Fig. 9. Of course, the segments in the figure cannot be extended to non-overlapping squares, but it is hard to draw a picture in which they can, as the squares would have to have very different sizes if we want several squares in each of the five groups.

We will now solve the situation of Fig. 9 in $O(n^3)$ time. The bases L, R, T, B, and M stand for the left, right, top, bottom, and middle (diagonal) sets of line segments. The superscripts denote the endpoints of these segments.

Note that any convex hull of a choice of points in this situation must follow these sets of endpoints in the correct order. That is, it starts at the left extreme point, then

goes to a number of points of L^B, then to a number of points of B^L, then to the bottom extreme point, and so on. It cannot, for example, go to a point of L^B, then to a point of B^L, and then back to a point of L^B.

The algorithm will repeatedly take two of the ten sets of endpoints, and for each combination of a point in one, and a point in the other set, compute the optimal subsolution connecting those points in linear time, based on earlier results. The subsolutions are computed in the following order:

- For each pair of points in L^T and L^B, we compute the optimal solution connecting them around the left side, using the algorithm for parallel line segments.
- For each pair of points in B^L and B^R, we compute the optimal solution connecting them around the lower side, using the algorithm for parallel line segments.
- For each pair of points $p \in M^{TL}$ and $q \in L^B$, compute the optimal chain connecting them that does not use any other point of M^{TL}. This can be done by trying a linear number of points $r \in L^T$ as the point to connect p to, and using the known optimal chain between r and q.
- For each pair of points $p \in M^{TL}$ and $q \in B^L$, compute the optimal chain connecting them around the left side that does not use any other points of M^{TL} and B^L. We do this by trying a linear number of points $r \in L^B$ as the point to connect q to, and combining this with the known optimal chain between p and r, computed in the previous step.
- For each pair of points $p \in M^{BR}$ and $q \in B^L$, compute the optimal chain connecting them that does not use any other point of M^{BR}. This can be done by trying a linear number of points $r \in B^R$ as the point to connect p to, and using the known optimal chain between r and q.
- For each pair of points $p \in M^{TL}$ and $q \in M^{BR}$, compute the optimal chain connecting them around the lower left side that does not use any other points of M^{TL} and M^{BR}. We can do this by trying a linear number of points $r \in B^L$ as the leftmost point of B^L that is used, and then combining the chains between p and r and between q and r that we computed in the two previous steps.
- For each pair of points $p \in M^{TL}$ and $q \in M^{BR}$, compute the optimal chain connecting them around the lower left side, which is allowed to use other points of M^{TL} and M^{BR}. We do this by using an adjusted version of the algorithm for parallel line segments. The optimal chain connecting p to q either uses another point from M^{TL} or M^{BR}, or it does not and uses the chain computed in the previous step. This means we must take the maximum of the formula given in Sect. 3.1.2, and the optimal chain of the previous step.
- In a symmetrical way, for each pair of points in M^{TL} and M^{BR}, compute the optimal chain connecting them around the upper right side that does not use any other points of M^{TL} and M^{BR}.
- Finally, check a quadratic number of pairs of a point from M^{TL} and a point from M^{BR}, and for each pair combine the chains of the previous two steps. The optimal solution is the maximum of these pairs.

The algorithm given above works when we assume that each set of endpoints is used at least once by the optimal solution. Of course, that need not be the case. But if from a certain group no point is used, then we also know that all points of the opposite

group may be used, and we are left with a smaller problem that can be solved in a similar way as described above. This means we can just try solving the problem under the assumption that one or more of the groups do not appear in the optimal solution, and then pick the best solution without increasing the time bound.

Theorem 4 *Given a set of n arbitrarily sized, non-overlapping, axis-aligned squares, the problem of choosing a point in each square such that the area of the convex hull of the resulting point set is as large as possible can be solved in* $O(n^7)$ *time.*

3.2.3 Unit Size Squares

The extra factor $O(n^4)$ that comes from the fact that it is hard to determine the extreme points, relies on situations where the size of the squares differs greatly, such as in Fig. 7a. When the squares have equal size, we show that there are only constantly many squares that can give the extreme points, thus reducing the running time of the above algorithm to $O(n^3)$.

For simplicity we assume general position, that is, no two squares have the same x- or y-coordinates. It is not true that all of the extreme points need to be corners of the extreme squares, as is shown in Fig. 11, but we do have the following property:

Lemma 4 *In the largest area convex hull problem for axis-aligned unit squares, an extreme square in the input set gives one of the extreme points of the optimal solution.*

Proof Let l be the vertical line at the leftmost x-coordinate in the input set, and let S_{left} be the square that has its left side at l. This square must clearly contribute a vertex to the optimal solution H, because otherwise the addition of one of its left corners would improve the optimal solution. If one of its left corners is used, then this must be the leftmost extreme point of H.

If the top right (the bottom right case is symmetric) corner p of S_{left} is part of H, then p must be part of the top right chain of H, by Lemma 3. If p is the topmost or the rightmost point on this chain, it is also an extreme point of H.

Now assume that p is not the topmost or rightmost point on its chain, see Fig. 10. Then the topmost point q has to be above and to the left of p. Suppose there is another point on the top right chain between q and p. Then this point must also be a top right corner of its square, and it must lie to the left of p. But since all squares are equally large, the left side of this square has to be to the left of l, a contradiction. So there are no points between p and q.

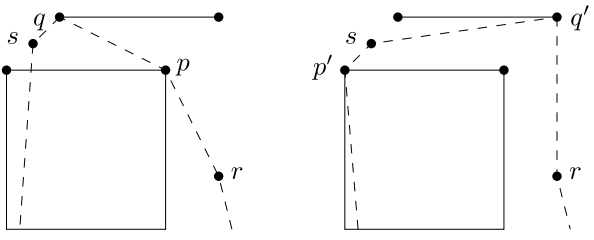

Fig. 10 Including the leftmost point increases the area

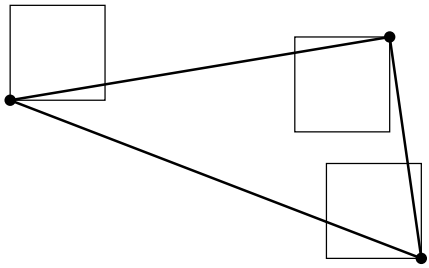

Fig. 11 The optimal solution does not use the top edge of the topmost square, since moving it down increases the area of the convex hull

There is also a point r that is the first on the chain to the right of and below p in the optimal solution H. Now q has to be at the top left corner of its square, because otherwise the square would once again lie to the left of l. Let the top left point of S_{left} be p', and the top right point of the same square as q be q'. If there are points on the top left chain above the horizontal line through p, let s be the one closest to q. If there are none, let s be the intersection of the upper left chain and the segment $\overline{pp'}$.

Now take H, and take p' instead of p, and q' instead of q. The resulting solution H' has a larger area than H, contradicting the assumption that p was not the topmost or rightmost point on its chain. This is because the triangle Δpqs is not larger than the triangle $\Delta pq's$. For the rest of the plane, all points inside H will also be inside H'. Furthermore, p' was not part of H so H' really is larger than H. H' is not necessarily convex, but making it convex will only increase the area. □

Note that this lemma also applies to overlapping squares.

As a consequence of this lemma, the largest convex hull problem for non-overlapping axis-aligned unit squares can be solved in $O(n^3)$ time. In the simple situation where the leftmost square gives the left extreme point, the topmost square the top extreme point, etc, this is easy to see, because then we have only 2^4 possible configurations for the extreme points, and we can just solve each problem using the $O(n^3)$ time algorithm described in Sect. 3.2.2. However, it is also possible that the topmost square gives for example the left extreme point, as shown in Fig. 11, where the top extreme point of the optimal solution is not in one of the extreme squares. However, this can only happen when the leftmost square is the same as the topmost square. In that case we have three possible points of that square to try, and when we try for example the lower left point we can just take the reduced problem and search for the extreme squares again (which can be done in constant time if we sorted them first). This procedure has to be followed at most four times, since we find an extreme point every time, thus the total number of configurations to try is still constant.

Theorem 5 *Given a set of n equal size, non-overlapping, axis-aligned squares, the problem of choosing a point in each square such that the area of the convex hull of the resulting point set is as large as possible can be solved in $O(n^3)$ time.*

3.2.4 Overlapping Unit Squares

For overlapping squares, the problem remains open. However, for overlapping squares of equal size, we can solve the problem in $O(n^5)$ time. Figure 12a shows

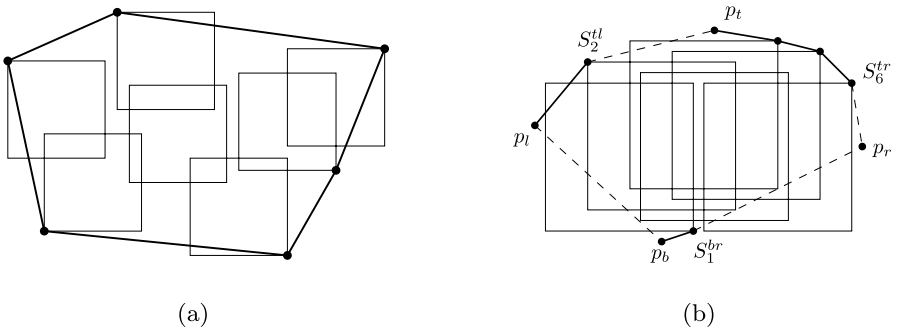

Fig. 12 (**a**) The largest convex hull for a set of intersecting unit squares. (**b**) The structure P_{2601}

this situation. We can solve this problem with a variation on the dynamic programming solution to Problem 2.

Assume the four extreme points p_l, p_t, p_r and p_b to be known. By Lemma 4 there are only a constant number of possibilities for them, and trying them all does not increase the time bound asymptotically. We call the remaining squares $S_1, \ldots, S_{n-4}$, sorted from left to right. For square S_i, we denote the top left corner by S_i^{tl}, the top right corner by S_i^{tr}, the bottom left corner by S_i^{bl}, and the bottom right corner by S_i^{br}. Abusing notation slightly, we also denote $S_0^{tl} = p_l$, $S_0^{tr} = p_t$, $S_0^{bl} = p_l$ and $S_0^{br} = p_b$.

For different $h, i, j, k \in \{0, \ldots, n-4\}$, we define the structure P_{hijk} to be the set of four chains that consists of a chain going from p_l to S_h^{tl} via a number of top left corners of squares S_m with $m < h$, a chain going from p_t to S_i^{tr} via a number of top right corners of squares S_m with $m < i$, a chain going from p_l to S_j^{bl} via a number of bottom left corners of squares S_m with $m < j$, and a chain going from p_b to S_k^{br} via a number of bottom right corners of squares S_m with $m < k$, such that no square participates on two different chains, and such that the area of the region bounded by these chains and the segments $\overline{S_h^{tl} p_t}$, $\overline{S_i^{tr} p_r}$, $\overline{S_j^{bl} p_b}$ and $\overline{S_k^{br} p_r}$ is maximal, see Fig. 12b.

If $h > i, j, k$ we can compute P_{hijk} in linear time using the structures P_{mijk} for $m < h$

$$P_{hijk} = \max_{m<h; m\neq i,j,k} \left(P_{mijk} + \Delta S_m^{tl} S_h^{tl} p_t \right)$$

If one of the other indices is the largest, a similar expression holds. There are $O(n^4)$ structures to compute in linear time, so the algorithm runs in $O(n^5)$ time and $O(n^4)$ space. Once all of the P's are computed, we pick the one with the largest area, and the problem is solved.

Theorem 6 *Given a set of n equal size, possibly overlapping, axis-aligned squares, the problem of choosing a point in each square such that the area of the convex hull of the resulting point set is as large as possible can be solved in $O(n^5)$ time.*

4 Smallest Convex Hull

In this section we will investigate the problem of finding the smallest area convex hull of a set of imprecise points. As in the previous section we will first look into the line segment model, and then move on to squares.

4.1 Line Segments

The problem we discuss in this section is the following:

Problem 4 *Given a set of parallel line segments, choose a point on each line segment such that the area of the convex hull of the resulting point set is as small as possible* (*see Fig.* 13).

This problem is equivalent to that of finding the smallest area convex polygon *transversal* of the line segments: the smallest convex polygon that intersects all segments. Mukhopadhyay *et al.* [21] study this problem and give an $O(n \log n)$ algorithm to solve it. Here we give a summary of the ideas of their algorithm, since they are relevant for the square case we will treat in the next section.

Observe that if a line segment contributes a vertex to the optimal solution, and the segment is not the leftmost or rightmost one, then this vertex must be on one of the endpoints of the segment. Otherwise we could move it up or down while decreasing the area until we reach an endpoint or the vertex disappears. Denote the leftmost segment by s_l and the rightmost segment by s_r.

To solve Problem 4, we define the *top chain* and the *bottom chain* of the set of segments. The top chain is a polyline connecting the lower endpoint of s_l to the lower endpoint of s_r, and is defined as the upper half of the convex hull of the set of all lower endpoints of the input segments, see Fig. 14a. The bottom chain is defined symmetrically.

The regions enclosed between the top and bottom chain is the *greatest common region*, or *GCR*, of the set of line segments. This is the largest region that is part of the convex hull of every possible choice of points on the segments, or equivalently, the intersection of all those convex hulls. If this regions is empty (i.e., the bottom chain always stays above the top chain), then there is a stabber of the segments, which leads to a zero area solution. This can be found in linear time [11].

To find the optimal solution, we need to find a point on s_l and a point on s_r such that the polygon we get by drawing tangents from those two points to the top and

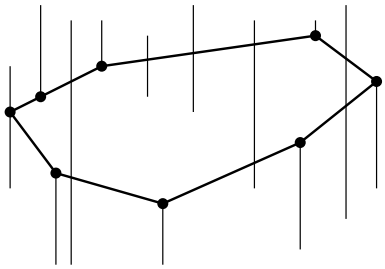

Fig. 13 The smallest convex hull for a set of line segments

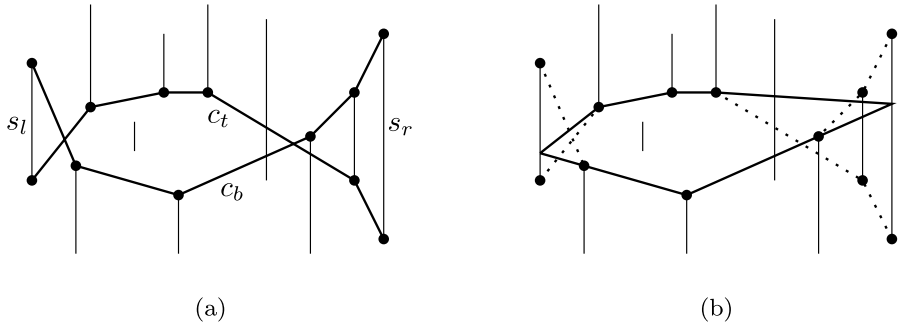

Fig. 14 (**a**) The top chain c_t and bottom chain c_b. (**b**) The optimal solution

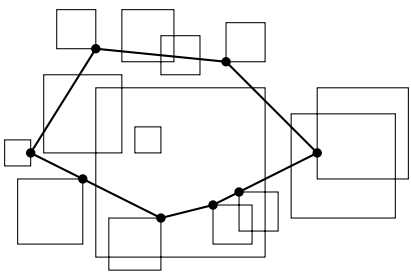

Fig. 15 The smallest convex hull for a set of squares

bottom chains, see Fig. 14b, has minimum area. The two positions are independent: if they would be connected by a straight line (that passes above the top chain or below the bottom chain), we could move them up or down together and decrease the area. We can compute their individual optimal positions in linear time. The total time is $O(n \log n)$, because we need to compute the chains.

Theorem 7 *Given a set of n arbitrarily sized, parallel line segments, the problem of choosing a point on each segment such that the area of the convex hull of the resulting point set is as small as possible can be solved in* $O(n \log n)$ *time.*

4.2 Squares

The problem we discuss in this section is the following:

Problem 5 *Given a set of axis-aligned squares, choose a point in each square such that the area of the convex hull of the resulting point set is as small as possible* (*see Fig.* 15).

4.2.1 Observations

The squares can be divided into those that define a vertex of the optimal convex hull, and those that do not. Of the squares that define a vertex, there are only four for which this point does not lie at a corner, and for those we can show it must at least be on some line segment.

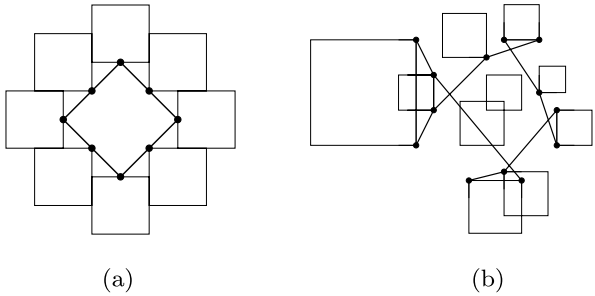

Fig. 16 (**a**) Up to four vertices of the smallest convex hull may not be a corner of one of the squares. (**b**) The top left, bottom left, top right, and bottom right chains

Lemma 5 *In the optimal solution, only the leftmost, rightmost, topmost, and bottommost vertices of the convex hull need not be corners of their squares.*

Proof Suppose some other vertex of the convex hull is not at the corner of its square. Suppose for example that this vertex lies between the leftmost and the topmost vertices. This means that moving the vertex either down or to the right will decrease the area of the convex hull. Since the vertex is not at a corner, it can move in at least one of those directions. Thus the convex hull cannot be optimal. □

These four squares will be called the *extreme squares* in four directions, and for these four squares, the points must lie on the inner edge. We call these squares S_l, S_r, S_t and S_b, meaning the leftmost, rightmost, topmost and bottommost squares. Here leftmost means the square with the leftmost right side, etc. We call their points the four *extreme points*, and denote them with p_l, p_r, p_t, and p_b, respectively. An example where there are indeed four points not at a corner is shown in Fig. 16a.

Similar to the case of parallel line segments, we now define four chains connecting corners of the squares. The *top left chain*, for example, will be the chain connecting the bottom right corner of S_l to the bottom right corner of S_t, via other bottom right corners of squares, such that the region to the lower right of the chain is convex and contains a point of every square. In the same way we can define the *top right chain*, the *bottom right chain*, and the *bottom left chain*. An example is shown in Fig. 16b. In the case where two of the extreme squares are the same, one of the chains reduces to a single point.

For every location of the point p_l, there is a *tangent point* $a_{lt}(p_l)$ on the top left chain such that the line through p_l and $a_{lt}(p_l)$ does not go through the region to the lower right of the top left chain. When there are more than one such points we choose the one that lies most to the upper right. Similarly, we define $a_{lb}(p_l)$ as the tangent point on the bottom left chain. For every point p_t we define tangent points $a_{tl}(p_t)$ and $a_{tr}(p_t)$ on the top left and top right chains, for every p_r we define two tangent points $a_{rt}(p_r)$ and $a_{rb}(p_r)$ on the top right and bottom right chains, and finally we define for every point p_b two tangent points $a_{bl}(p_b)$ and $a_{br}(p_b)$ on the bottom left and bottom right chains. All those tangent points are vertices of the chains. Note that they may also be corners of the extreme squares.

Lemma 6 *If the points p_l, p_t, p_r and p_b are known, in the optimal solution the point p_l is connected to p_t by a straight line segment if this segment does not intersect the*

top left chain, and otherwise via the piece of the top left chain between $a_{lt}(p_l)$ and $a_{tl}(p_t)$. Similarly p_t is connected to p_r by a straight line segment or via the piece of the top right chain between $a_{tr}(p_t)$ and $a_{rt}(p_r)$, p_r is connected to p_b by a straight line segment or via the piece of the bottom right chain between $a_{rb}(p_r)$ and $a_{br}(p_b)$, and p_b is connected to p_l by a straight line segment or via the piece of the bottom left chain between $a_{bl}(p_b)$ and $a_{lb}(p_l)$.

Proof The optimal solution H contains the convex hull of p_l, p_t, p_t and p_b. If there is a vertex q of the top left chain that is to the top left of the segment $\overline{p_l p_t}$, but is not contained in the optimal solution, then the solution is invalid. This is because q is a bottom right corner of its square, so the whole square is disjoint from H. Similar reasoning applies to the other three chains. □

In the degenerate case where the squares S_l, S_t, S_r and S_b are just points, this implies that the four chains together form the optimal solution.

Note that just as in the case of line segments, we can still find out if a zero area solution exists in linear time, since a solution of zero area still corresponds to a stabber, and a stabber of a set of squares can be computed in $O(n)$ time if it exists [11]. However, this is no longer directly related to the greatest common region.

4.2.2 Algorithm

So far everything is still the same as in the case of line segments. However, we can no longer use the approach we used in the case of the line segments, because now the locations of the four extreme points are no longer independent. It can really happen that in the optimal solution, two of those points are directly connected by a straight line, as can be seen in the example in Fig. 15. Instead, we use an alternative approach.

Each of the four extreme points can move over an edge of its square. We divide this edge into a linear number of intervals. Within each interval, if the point would be chosen there, the vertices on the chains the point would be connected to are the same. This means the endpoints of the intervals are exactly the points that lie on a line through two consecutive vertices of one of the chains. The resulting intervals are shown in Fig. 17.

If we would only consider one extreme point at a time, as we did with the line segments, then the point would have to be at one of the endpoints of these intervals, since the area function is piecewise linear, and only changes exactly at those endpoints. However, when we consider all four extreme points simultaneously, then the

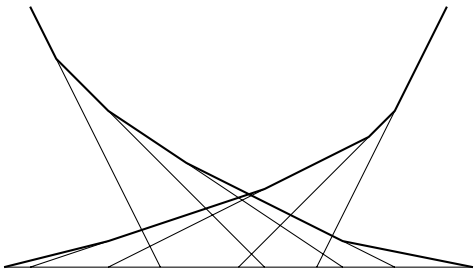

Fig. 17 The lower left and the lower right chain divide the upper edge of S_b into a linear number of intervals

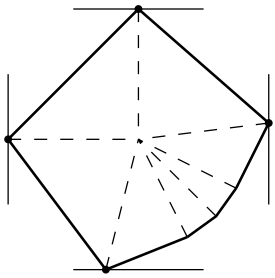

Fig. 18 The area of a solution can be decomposed into triangles that depend on at most two variables

optimal location of one or more of the points may also be somewhere in the middle of an interval, as is already apparent in Fig. 16a.

Assume we know for each of the four extreme points the interval on which they must be in the optimal solution. We then know the tangent points on the chains they must be connected to if we do not look at the other extreme points. Since we know this for each extreme point, we can see whether they will be connected to each other or to a chain. For example, the left point p_l has a point $a_{lt}(p_l)$ on the top left chain it could be connected to, and the upper point p_t has a point $a_{tl}(p_t)$ on the top left chain it could be connected to. If $a_{lt}(p_l)$ lies to the lower left of $a_{tl}(p_t)$, then p_l and p_t will be connected to their tangent points, and not to each other. If $a_{lt}(p_l)$ lies to the upper right of $a_{tl}(p_t)$, then p_l and p_t will be connected by a straight line segment. If $a_{lt}(p_l)$ is equal to $a_{tl}(p_t)$, then we do not know yet.

We can now write the area of the convex hull as a polynomial in four variables that specify the exact locations of the extreme points within their interval. This polynomial will have degree at most 2, because we can decompose the area into triangles that depend on at most two of the variables, as shown in Fig. 18. We can find the minimum of this polynomial within the bounds given by the intervals on which the points can move, in constant time. In the case where we do not know whether two points will be connected by a straight line segment or via a point on a chain, we simply try both and add an extra restriction to the variables in the one case.

We can now easily solve the problem in $O(n^4)$ time. In the optimal solution, each of the four extreme points needs to be on one of the intervals of its segment. This means a total of $O(n^4)$ possible combinations of intervals, and for each combination the solution requires solving a polynomial that does not depend on n. However, many of the combinations of intervals seem to be redundant, because a solution using them can clearly be seen not to be optimal. Indeed we can show that we do not need to spend $O(n^4)$ time to solve this problem, and improve it to $O(n^2)$.

We observe that each connection between two of the extreme points must be one of three types. We call the connection of *type 0* if the points are connected by a single line segment that does not touch the respective chain in the optimal solution. We call the connection of *type 1* if the points are connected by a single line segment that touches the respective chain. We call the connection of *type 2* if they are not connected by a single line segment. In Fig. 19 examples are shown of only type 0 connections, only type 1 connections, only type 2 connections, and an example with two connections of type 0, one of type 1, and one of type 2.

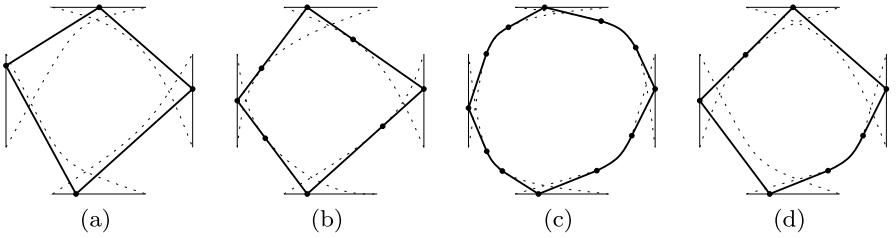

Fig. 19 The points can be connected either (**a**) directly, (**b**) just touching a point on the chain, (**c**) via multiple points of the chain. (**d**) Multiple cases can appear together

There are only a constant number of possible combinations of connections between the four extreme points, namely 3^4. If we can compute the optimal solution of each type and test their validity in less than $O(n^4)$ time, then we can just pick the best one and we have a faster algorithm. We make the following observations.

- Each connection of type 1 reduces the number of possible combinations of intervals by a linear factor.
- Every pair of connections of type 2 divides the problem into two independent subproblems.
- Every pair of adjacent connections of type 0 removes the need to divide one of the extreme squares into intervals, reducing the number of possible combinations of intervals by a linear factor.

The first statement is true because there is only a linear number of pairs of intervals for the two extreme points in question for which the point to which they can be connected on the chain is the same, and this is required for a type 1 connection.

The second statement is similar to the independence of the extreme points in the case of line segments. Since we know by assumption that the two connections are of type 2, the optimal subproblems together have to be the optimal solution for the complete problem. The two connections could be adjacent, giving one subproblem of linear complexity and one of potentially cubic complexity, or they could be opposite to each other, giving two problems of potentially quadratic complexity.

The third statement means that if one of the extreme points has only connections of type 0, there is no need to divide its edge into intervals, since the structure of the convex hull, and therefore the polynomial describing the area, will be the same.

When one or more of these patterns occur, we can solve the problem in less time. For example, take type 0-1-2-2. This type contains two type 2 connections, and also a type 1 connection. The type 2 connections divide the problem into two independent subproblems. The smallest subproblem can be solved in linear time, by just choosing the best out of a linear number of intervals. For each interval we know their tangent points on the chains, so it can be solved in constant time. The larger subproblem contains a type 1 connection. This means there is only a linear number of combinations of intervals such that the connection is really of type 1. We can find them and store them easily in quadratic time. Now we look at a quadratic number of groups of three intervals; one combination of two that we just stored, and one from the remaining extreme point (the one between the type 0 and type 2 connections). For each three intervals, we can solve the problem in constant time since all tangent points are known.

Table 2 The 21 possible types

0-0-0-0	0-0-0-1	0-0-0-2	0-0-1-1	0-0-1-2	0-1-0-1	0-1-0-2
1-1-1-1	1-1-1-2	1-1-1-0	1-1-2-2	1-1-2-0	1-2-1-2	1-2-1-0
2-2-2-2	2-2-2-0	2-2-2-1	2-2-0-0	2-2-0-1	2-0-2-0	2-0-2-1

We can also check in constant time whether all connections are of the correct types (in particular the type 0 connection), by looking at the tangent points. We have now solved both subproblems in $O(n^2)$ time. We finally need to check whether the subsolutions together yield a convex shape. If they don't, then the type was not 0-1-2-2. If they do, then it is a potential optimal solution.

There are 3^4 possible types, but after removing symmetries only 21 remain, see Table 2. Note that in all 21 types at least one of these three patterns has to occur, and thus every type can be solved in $O(n^3)$ time. Furthermore, in all but one case (and its symmetric variants), actually two of these patterns occur together, and they can be solved in $O(n^2)$ time. All these types can be solved in a similar way to the one described above. The one exception is the one shown in Fig. 19d.

This type has only one of the three patterns. However, we show we can solve it in linear time.

Lemma 7 *The pattern with a type* 0 *connection*, *a type* 1 *connection*, *a type* 0 *connection and a type* 2 *connection*, *in that order*, *can be solved in* $O(n)$ *time*.

Proof We assume the types of connections to be as in Fig. 19d, other cases are symmetric. Since the top right and bottom left connections do not touch the chains, we do not need to look at these two chains any more. There will be one point p on the top left chain that is the tangent point of the top left connection. Now the top left connection is a single line segment from the leftmost extreme point to the topmost extreme point, and is still allowed to rotate around p within some interval such that it does not intersect the top left chain, see Fig. 20a.

For a fixed position of the top left connection, and therefore the points p_l and p_t, the optimal solution is easy to see. Find the two consecutive points on the bottom right chain such that the leftmost extreme point has its y-coordinate between the y-coordinates of those two points. The bottommost extreme point will be on the line through these two points, because moving it in either direction from that position would increase the area. In the same way, the rightmost point must be on the line through the two points that have their x-coordinates closest to that of the topmost extreme point, as in Fig. 20a.

Now if we start with p as the bottom leftmost point on the top left chain, and rotate the line connecting the leftmost extreme point to the topmost extreme point around it in clockwise direction, the bottommost two points of the bottom right chain that determine the position of the bottommost extreme point will only move upwards, while the two points that determine the position of the rightmost extreme point will also only move upwards, see Fig. 20b. When the line rotates far enough, p will change to the next point of the top left chain, but still the points on the bottom right chain only move towards the upper right. This means that after a linear number of steps we have tried all possibilities and found the optimal solution.

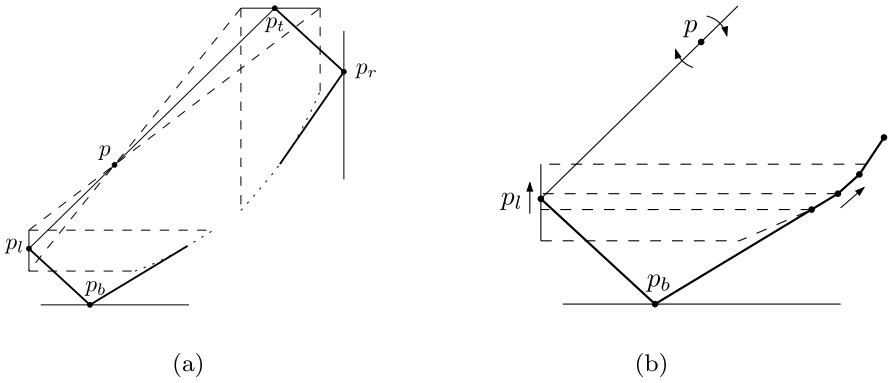

Fig. 20 (**a**) The special case 0-1-0-2. (**b**) The *bottom left* part zoomed in

Because we assumed the type is 0-1-0-2, we still have to make sure that the solutions we check are indeed of this type. This means that we have to check that the lower left and upper right connections do not intersect their respective chains. We can check this on average in constant time every step, if we keep track of the slope of the connections and the first vertex of the chain it would intersect. Also, the points on the lower right chain have to be in the correct order, otherwise the found hull is not convex and may even intersect itself. We can check this in constant time as well. □

We conclude that every situation can be solved in $O(n^2)$ time, and therefore the whole problem can be solved in $O(n^2)$ time.

Theorem 8 *Given a set of n arbitrarily sized, possibly overlapping, axis-aligned squares, the problem of choosing a point in each square such that the area of the convex hull of the resulting point set is as small as possible can be solved in* $O(n^2)$ *time.*

5 Perimeter Versus Area

Until now we have only considered area of the convex hull as the measure to maximize or minimize, but there are other measures that can be used. The other natural measure for determining the size of convex polygons is the perimeter. In this section we will analyse the relevant differences between those two measures, and how the results of the previous sections can be extended to the perimeter measure.

One important observation is the way the size of a polygon changes when only one point is moving over a line, while the rest remains fixed. The area of the polygon is just a linear function of the moving point, but the perimeter is not: The distance between two points changes as a symmetric hyperbolic function with a minimum when one point moves over a straight line. This means the perimeter changes as the sum of two such functions, as long as the combinatorial structure of the hull does not change. Another important difference is that the area of a polygon is invariant

under many kinds of transformations, including shearing, while the perimeter is only invariant under rigid transformations.

More specific to the problem at hand, when we want to maximize the area of a polygon, convexity is automatically achieved. When we want to maximize the perimeter, however, convexity has to be taken care of explicitly. When looking for minimal size, this works the other way around. A minimal perimeter polygon will automatically be convex, while a minimal area polygon is generally not.

We can adjust all of the above algorithms to the perimeter measure in a more or less straightforward fashion. The time bounds for the largest convex hull indeed become worse, $O(n^3)$ for line segments becomes $O(n^5)$, and $O(n^7)$ for squares becomes $O(n^{10})$. On the other hand, the time bounds for the smallest convex hull become better; all problems considered can be solved in only $O(n \log n)$ time. The details of the changed algorithms can be found in the remainder of this section.

Finally, the perimeter measure introduces some computational difficulty. All of the above mentioned adjusted algorithms require the comparison of sums of square roots of integers, and it is not clear whether this can be done efficiently. Therefore, we assume a computation model that allows real arithmetic in the remainder. Another implication is that the problems described are not known to be in NP, so we cannot prove anything NP-complete.

5.1 Longest Perimeter

When looking for the largest perimeter rather than area, we still only have to look at the endpoints of the line segments or the corners of the squares. The function that describes the size is no longer linear, but because it has no maximum, there will still always be at least one direction in which a point can be moved such that the size does not decrease. However, we cannot simply duplicate the proofs of Lemmas 1 and 2, because by moving points to increase the perimeter we may lose convexity.

Lemma 8 *Given a set of line segments or squares, there is a choice of points such that every point is at an endpoint or corner, and the perimeter of the convex hull of the resulting point set is as large as possible.*

Proof Assume there is an optimal solution with a point q not at an endpoint or corner. If the point is not a vertex of the convex hull, we can just move it anywhere without decreasing the length of the perimeter. So assume the point is a vertex of the convex hull. Let l be a line segment such that q is not at an endpoint of this segment, but is allowed to move anywhere on the segment.

Let p and r be the neighboring vertices of q on the convex hull. The length of the chain pqr is a hyperbolic function of the position of q on l. This function has a single minimum, so there is at least one direction in which q can move such that the length of pqr increases. Since we assumed the position of q to be optimal, moving q in this direction must change the combinatorial structure of the convex hull. This means that there must be another vertex of the convex hull on either the line through p and q, or through q and r.

Now assume without loss of generality that l is vertical, and that moving q down over l increases the length of pqr. In this case there must be a point s on the line

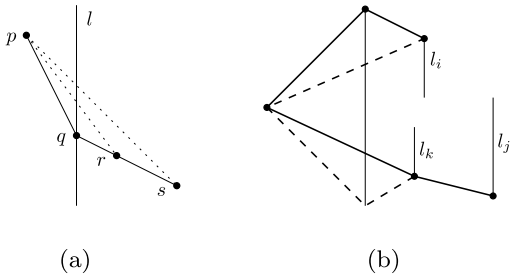

Fig. 21 (**a**) Moving q down increases the perimeter. (**b**) The optimal solution P_{ij} (*solid*) does not contain the smaller optimal subsolution P_{ik} (*dashed*)

through q and r that is the next vertex of the convex hull from r, see Fig. 21a. If p is on the same side of l as r and s are, mirror p in l and note that this does not influence the length of pqr or pqs as q moves over l. Now note that the length of pqr is minimized exactly when q is at the intersection point of l and $\overline{pr}$. Since moving q down increases this length, q must be at or below this intersection point. This means, however, that the intersection point of l and $\overline{ps}$ has to be at or above q. Thus moving q down will also increase the length of pqs, and the length of the convex hull will grow even though the combinatorial structure changes. Thus q is not at its optimal position, contradicting the first assumption of this proof. □

Even though we still only have to look at a few possible locations for every point, we cannot easily extend the algorithms in Sects. 3.1 and 3.2. The reason for this is that we now explicitly have to ensure convexity of the solutions in the dynamic programming algorithm for line segments. The algorithm for squares heavily relies on the algorithm for line segments.

The fundamental property that the dynamic programming algorithm is based on, is that the optimal solution up to some line segments l_i and l_j always consists of some smaller optimal solution and one extra line segment. This property no longer holds in the case of perimeter. An example is shown in Fig. 21b. Here the solid line represents the optimal solution from the top of l_i to the bottom of l_j. However, this chain does not include the optimal solution from the top of l_i to the bottom of l_k, which is represented by the dashed line. The reason for this is that adding the segment from l_k^- to l_j^- to the dashed line yields a non-convex chain, and making it convex decreases the length. In the case of area there was no problem, because then making it convex increased the area.

We can solve the problem in $O(n^5)$ time instead of $O(n^3)$, by keeping the optimal solution for every pair of points, and every direction in which the points are connected. This way, we can ensure convexity. Since there are only a linear number of directions that lead to another point, this leads to an $O(n^5)$ time algorithm.

Define $P_{ij\sim km}$ to be the chain that starts at the top of l_i, then goes to the left to the top of l_j, then via a number of other tops to the leftmost point, then back to the right via a number of bottoms to the bottom of l_k, and finally to the bottom of l_m, such that it is convex and of maximal length, and doesn't use both the top and bottom endpoint of any segment. Abusing notation slightly, we also use $P_{ij\sim km}$ to denote its length. Now we can define a recursive relation similar to the one in Sect. 3.1.

If $i < m$, then

$$P_{ij\sim km} = \max_{h<k;h\neq i,j} \left(P_{ij\sim hk} + \overline{l_k^- l_m^-} \;\middle|\; \measuredangle l_h^- l_k^- l_m^- \text{ is convex} \right)$$

We maximize over the length of the chains. If $i > m$ a symmetric expression holds, where we maximize over the tops of the line segments to the left of l_j, rather than the bottoms of the line segments to the left of l_k.

Theorem 9 *Given a set of n arbitrarily sized, parallel line segments, the problem of choosing a point on each segment such that the perimeter of the convex hull of the resulting point set is as large as possible can be solved in $O(n^5)$ time.*

For arbitrary line segments, the NP-hardness proof for the largest area problem still holds unchanged for the largest perimeter. The proof does not depend on the measure, only on the fact that the value of a variable is equal in its true and false states. However, it is not known whether this problem is in NP, since checking a solution involves comparing sums of square roots [24].

Theorem 10 *Given a set of n arbitrarily oriented, possibly intersecting line segments, the problem of choosing a point on each segment such that the perimeter of the convex hull of the resulting point set is as large as possible is NP-hard.*

For squares, all of the observations in Sect. 3.2.1 still apply. Most of them are independent of the measure of the size, except for the last observation, which states that the extreme points do not need to be in the extreme squares, but this is also true for perimeter with the same example (in Fig. 7a). Also the reduction from the set of squares to five groups of line segments does not use the measure of the convex hull, so we can still do this for the perimeter.

We can solve this situation in $O(n^6)$ time and $O(n^4)$ space now, in a similar way as for area, but by looking at pairs of two points at a time instead of just one, because we need to ensure convexity. This gives $O(n^4)$ subsolutions to compute, and it takes $O(n^2)$ time to compute them. In each step below, we will select four points from certain groups. We always require that the two first mentioned points are directly connected, and the two last mentioned points are directly connected, while we vary the chain between the second and third point. We assume for now that at least two points from every group are used.

Figure 9 is repeated in Fig. 22 for convenience.

- For each group of two points in L^T and two points in L^B, we compute the optimal solution connecting them around the left side, using the algorithm for parallel line segments.
- For each group of two points in B^L and two points in B^R, we compute the optimal solution connecting them around the lower side, using the algorithm for parallel line segments.
- For each group of two points $p, p' \in L^T$, a point $q' \in L^B$ and a point $q \in B^L$, compute the optimal chain connecting them. Note that this chain will not use any other points from B^L. This can be done by trying a linear number of points $r \in L^B$

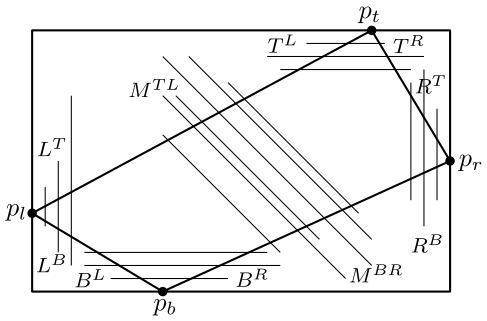

Fig. 22 Five groups of line segments give ten sets of endpoints

as the point to connect q' to, and using the known optimal chain between p, p' and r, q'. We need to ensure that $\angle rq'q$ is convex.

- For each group of two points in B^R, a point in B^L and a point in L^B, compute the optimal chain connecting them symmetric to the previous step.
- For each group of two points p, $p' \in L^T$ and two points $q', q \in B^R$, compute the optimal chain connecting them around the lower left. This can be done by choosing a point $r \in L^B$ and a point $s \in B^L$, and combining the solutions from p, p' to r, s and from r, s to q', q of the previous two steps. This step is critical: it is the only step that really takes $O(n^6)$ time.
- For each group of two points p, $p' \in L^T$, a point $q' \in B^R$ and a point $q \in M^{BR}$, compute the optimal chain connecting them, by choosing another point $r \in B^R$ and using the optimal solution from p, p' to r, q' and ensuring convexity of $\angle rq'q$.
- For each group of a point in M^{TL}, a point in L^T, a point in B^R and a point in M^{BR}, compute the optimal chain connecting them by choosing another point from L^T, symmetric to the previous step.
- For each group of two points in M^{TL} and two points in M^{BR}, compute the optimal chain connecting them around the lower left using an adjusted version of the algorithm for line segments.
- Symmetrically, for each group of two points in M^{TL} and two points in M^{BR}, compute the optimal chain connecting them around the upper right.
- Choose from $O(n^4)$ possible combinations of points in M^{TL} and M^{BR} the one with the largest perimeter.

Every step can be executed in $O(n^6)$ time. This only works when in the optimal solution there are at least two points in every group, but as in the case of area the problem becomes easier when one of the groups has only one or even zero points in the optimal solution, and there still are only a constant number of easier situations to deal with.

Using this algorithm, we can solve the problem of finding the longest perimeter solution for a set of non-intersecting squares in a total of $O(n^{10})$ time, since we still need a factor of $O(n^4)$ to find the four extreme points.

Theorem 11 *Given a set of n arbitrarily sized, non-overlapping, axis-aligned squares, the problem of choosing a point in each square such that the perimeter of the convex hull of the resulting point set is as large as possible can be solved in* $O(n^{10})$ *time.*

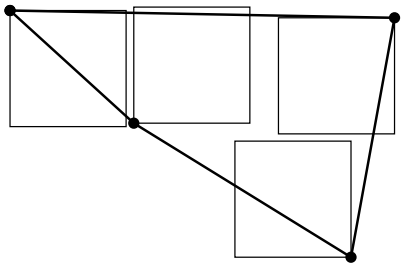

Fig. 23 The topmost square does not give one of the extreme points

Unlike the area case, we cannot improve the algorithm when the squares have equal size. Figure 23 shows an example where the optimal solution, denoted by the thick line, uses the topmost square at the bottom left corner while this is not an extreme point. So Lemma 4 cannot be extended to the perimeter measure. Again, the problem is that making a non-convex polygon (such as the one on the right in Fig. 10) convex does not increase the perimeter, while it does increase the area.

The algorithm for squares of equal size that does not require the squares to be non-overlapping can be adjusted to the perimeter measure. The same idea as in the previous two algorithms applies: we need to ensure convexity and therefore need to define a structure on pairs of points from each of the four groups instead of just one point. Furthermore, we now need to try all combinations of the four extreme points since we cannot use Lemma 4.

We try all possible combinations for p_l, p_t, p_r and p_b, which are $O(n^4)$ possibilities. For each one, we now define the structure $P_{ii'jj'kk'll'}$ to be the set of four chains that consists of a chain going from p_l to $S^{tl}_{i'}$ via a number of top left corners of squares P_m with $m < i$ and then to S^{tl}_i with a straight connection, a chain going from p_t to $S^{tr}_{j'}$ via a number of top right corners of squares P_m with $m < j$ and then to S^{tr}_j with a straight connection, a chain going from p_l to $S^{bl}_{k'}$ via a number of bottom left corners of squares P_m with $m < k$ and then to S^{bl}_k with a straight connection, and a chain going from p_b to $S^{br}_{l'}$ via a number of top left corners of squares P_m with $m < l$ and then to S^{br}_l with a straight connection, such that no square participates on two different chains, such that all chains are convex, and such that the total length of these chains is maximal. This gives $O(n^8)$ different structures, that can all be computed in linear time by varying the rightmost point of $\{i, i', j, j', k, k', l, l'\}$. The total algorithm runs in $O(n^{13})$ time and $O(n^8)$ space.

Theorem 12 *Given a set of n equal size, possibly overlapping, axis-aligned squares, the problem of choosing a point in each square such that the perimeter of the convex hull of the resulting point set is as large as possible can be solved in $O(n^{13})$ time.*

5.2 Shortest Perimeter

When looking for the shortest perimeter, we can expect less difficulties with convexity than in the case of the longest perimeter. However, the fact that the size is no longer a linear function, but a hyperbolic one with a minimum may cause some problems.

We can still prove that almost all vertices of the optimal solution must be at endpoints or corners of their line segments or squares. When a vertex of the convex hull is allowed to move in two opposite directions (for example up and down), and its neighboring vertices are on both sides of this direction (so one on the left and one on the right), the minimum of the hyperbolic function occurs exactly when the vertex is between its two neighbors, and then it is not a vertex that determines the shape of the convex hull. In every other situation, it can move to make the perimeter smaller. Thus this situation cannot occur in the optimal solution, and we have, as with the area, that only the leftmost and rightmost line segments, or the four extreme squares, can have a vertex of the convex hull that is not at an endpoint or corner.

The $O(n \log n)$ time algorithm for line segments can be adapted easily to work for the perimeter. In fact, Rappaport [23] describes an algorithm that solves this problem when the line segments can have up to some constant k different orientations in $O(3^k n \log n)$ time.

Theorem 13 *Given a set of n arbitrarily sized, parallel line segments, the problem of choosing a point on each segment such that the perimeter of the convex hull of the resulting point set is as small as possible can be solved in $O(n \log n)$ time.*

The algorithm for squares can also be extended to the perimeter measure without problems. The main difference is that the functions to minimize on each interval are now no longer linear. They will now contain some square roots, but since there are at most eight terms with square roots with variables under them, they can be rewritten to polynomials in constant time. This yields constant degree polynomials, so if we assume their roots can be found at all, then they can be found in constant time.

However, we can use some properties of the perimeter to actually achieve a better algorithm. Instead of looking at the original problem, we mirror the four chains in the extreme line segments, which does not change the length of the solution when the points can only move over those segments. The result is shown, rotated 45°, in Fig. 24. The thick line denotes the optimal solution. We now want to find a shortest path between the chains, where it should be noted that the segment on the far left is the same as the segment on the far right.

The tangent line of two chains in this situation can be computed in linear time [29]. The optimal solution can be found, for example, by again trying all possible combinations of which chains are touched by the optimal solution and checking if

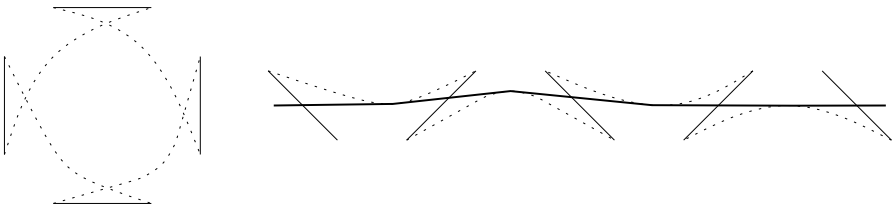

Fig. 24 The four chains can be mirrored in the extreme line segments to simplify the shortest perimeter problem

the solutions are feasible. This yields a linear time algorithm once the chains are known, while the computation of the chains takes $O(n \log n)$ time.

Theorem 14 *Given a set of n arbitrarily sized, possibly overlapping, axis-aligned squares, the problem of choosing a point in each square such that the perimeter of the convex hull of the resulting point set is as small as possible can be solved in* $O(n \log n)$ *time.*

6 Conclusions

We studied the problem of computing the largest or smallest convex hull of a set of imprecise points. We have seen many variants of the problem, varying in goal, measure, model and restrictions, and many different algorithms to solve them. Our results are given in Table 1.

These results suggest that the problem of finding the smallest area or perimeter convex hull is easier than finding the largest convex hull. The running times are better, and there are fewer restrictions with the smallest hull. For the largest convex hull it seems important that the regions do not intersect. It also looks like the area is easier to maximize than the perimeter, while the perimeter is easier to minimize than the area.

6.1 Future Work

Many problems are still open, and there are various directions of research to be pursued. Most notably, what is the status of the problem of finding the largest convex hull when the squares are allowed to intersect? Also, what results can be obtained for the circle model? Another open problems is computing the largest convex hull for a set of arbitrarily oriented non-intersecting line segments.

A second direction of future work concerns approximation. For the problems that have no efficient exact algorithms, how well can they be approximated efficiently? This applies to the open problems noted above, as well as some of the problems for which polynomial algorithms have been given but with high exponents, such as the $O(n^{10})$ time bound, and of course the NP-hard problems. Recently we gave linear time approximation schemes for a number of convex hull maximization problems [19].

Thirdly, for many other problems in computational geometry, data imprecision and its effect on the outcome of algorithms should be studied. Such problems include the longest and shortest minimum spanning trees for a set of imprecise points, the Delaunay triangulation, and any other geometric structure that is uniquely defined on a set of points and can somehow be measured. In [17] we studied various basic geometric measures under this model.

Higher-dimensional versions are also open.

References

1. Abellanas, M., Hurtado, F., Ramos, P.A.: Structural tolerance and Delaunay triangulation. Inf. Process. Lett. **71**, 221–227 (1999)
2. Bandyopadhyay, D., Snoeyink, J.: Almost-Delaunay simplices: nearest neighbour relations for imprecise points. In: Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms, pp. 410–419 (2004)
3. Bajaj, C.: The algebraic degree of geometric optimization problems. Discrete Comput. Geom. **3**, 177–191 (1988)
4. Barber, B.C.: Computational geometry with imprecise data and arithmetic. TR-377-91, Ph.D. Thesis, Princeton University, Department of Computer Science (1993)
5. Boissonnat, J.-D., Lazard, S.: Convex hulls of bounded curvature. In: Proceedings of the 8th Canadian Conference on Computational Geometry, pp. 14–19 (1996)
6. Cai, L., Keil, J.M.: Computing visibility information in an inaccurate simple polygon. Int. J. Comput. Geom. Appl. **7**, 515–538 (1997)
7. de Berg, M., Gudmundsson, J., Katz, M.J., Levcopoulos, C., Overmars, M.H., van der Stappen, A.F.: TSP with neighborhoods of varying size. J. Algorithms **57**(1), 22–36 (2005)
8. de Berg, M., Meijer, H., Overmars, M.H., Wilfong, G.T.: Computing the angularity tolerance. Int. J. Comput. Geom. Appl. **8**, 467–482 (1998)
9. Desaulniers, H., Stewart, N.F.: Robustness of numerical methods in geometric computation when problem data is uncertain. Comput. Aided Des. **25**, 539–545 (1993)
10. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.S.B.: Touring a sequence of polygons. In: Proceedings of the 35th ACM Symposium on Theory of Computing, pp. 473–482 (2003)
11. Edelsbrunner, H.: Finding transversals for sets of simple geometric figures. Theor. Comput. Sci. **35**, 55–69 (1985)
12. Fiala, J., Kratochvil, J., Proskurowski, A.: Systems of distant representatives. Discrete Appl. Math. **145**, 306–316 (2005)
13. Garey, M., Graham, R., Johnson, D.: The complexity of computing Steiner minimal trees. SIAM J. Appl. Math. **32**, 835–859 (1977)
14. Goodrich, M.T., Snoeyink, J.: Stabbing parallel segments with a convex polygon. Comput. Vis. Graph. Image Process. **49**, 152–170 (1990)
15. Guibas, L., Salesin, D., Stolfi, J.: Constructing strongly convex approximate hulls with inaccurate primitives. Algorithmica **9**, 534–560 (1993)
16. Khanban, A.A., Edalat, A.: Computing Delaunay triangulation with imprecise input data. In: Proceedings of the 15th Canadian Conference on Computational Geometry, pp. 94–97 (2003)
17. van Kreveld, M., Löffler, M.: Largest bounding box, smallest diameter, and related problems on imprecise points. In: Proceedings of the 10th Workshop on Algorithms and Data Structures. LNCS, vol. 4619, pp. 447–458. Springer, Berlin (2007)
18. Li, Z., Milenkovic, V.: Constructing strongly convex hulls using exact or rounded arithmetic. Algorithmica **8**, 345–364 (1992)
19. Löffler, M., van Kreveld, M.: Approximating largest convex hulls for imprecise points. In: Proceedings of the 5th Workshop on Approximation and Online Algorithms. LNCS, vol. 4927, pp. 89–102. Springer, Berlin (2008)
20. Mata, C., Mitchell, J.S.: Approximation algorithms for geometric tour and network design problems. In: Proceedings of the 11th ACM Symposium on Computational Geometry, pp. 360–369 (1995)
21. Mukhopadhyay, A., Kumar, C., Greene, E., Bhattacharya, B.: On intersecting a set of parallel line segments with a convex polygon of minimum area. Inf. Process. Lett. **105**(2), 58–64 (2008)
22. Nagai, T., Tokura, N.: Tight error bounds of geometric problems on convex objects with imprecise coordinates. In: Proceedings of the 4th Japanese Conference on Discrete and Computational Geometry. LNCS, vol. 2098, pp. 252–263. Springer, Berlin (2000)
23. Rappaport, D.: Minimum polygon transversals of line segments. Int. J. Comput. Geom. Appl. **5**(3), 243–256 (1995)
24. O'Rourke, J.: Sum of Square Roots. http://maven.smith.edu/~orourke/TOPP/P33.html
25. Ostrovsky-Berman, Y., Joskowicz, L.: Uncertainty envelopes. In: Proceedings of the 21st European Workshop on Computational Geometry, pp. 175–178 (2005)
26. Polishchuk, V., Mitchell, J.S.B.: Touring convex bodies—a conic programming solution. In: Proceedings of the 17th Canadian Conference on Computational Geometry, pp. 290–293 (2005)

27. Safra, S., Schwartz, O.: On the complexity of approximating TSP with neighborhoods and related problems. In: Proceedings of the European Symposium on Algorithms. LNCS, vol. 2832, pp. 446–458. Springer, Berlin (2003)
28. Sellen, J., Choi, J., Yap, C.-K.: Precision-sensitive Euclidean shortest paths in 3-space. SIAM J. Comput. **29**, 1577–1595 (2000)
29. Toussaint, G.T.: Solving geometric problems with the rotating calipers. In: Proceedings of the IEEE Mediterranean Electrotechnical Conference (1983)
30. Yap, C.-K.: Robust geometric computation. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, pp. 927–952. Chapman & Hall/CRC, Boca Raton (2004). Chap. 41