Largest and Smallest Tours and Convex Hulls for Imprecise Points*

(Extended Abstract)

Maarten Löffler and Marc van Kreveld

Institute of Information and Computing Sciences
Utrecht University, the Netherlands
{mloffler, marc}@cs.uu.nl

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 tours and convex hulls, measured by length, and in the latter case also 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 $O(n)$ to $O(n^{13})$, and prove NP-hardness for some geometric problems on imprecise points.

1 Introduction

In computational geometry, many fundamental problems take a point set as input, on which some computation is done, such as 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.

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.

* This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) through the BRICKS project GADGET and through the project GOGO.

L. Arge and R. Freivalds (Eds.): SWAT 2006, LNCS 4059, pp. 375–387, 2006.

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 [15] 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.

Espilon Geometry is a framework for robust computations on imprecise points. Guibas *et al.* [11] 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. A related concept is that of *tolerance* [1]; see also [12] and [2].

Boissonnat and Lazard [4] 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. They give a polynomial time approximation algorithm.

Goodrich and Snoeyink [10] 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. Given a sequence of k polygons with a total of n vertices, Dror *et al.* [7] study the problem of finding a tour that touches all of them in order that is as short as possible. Higher dimensions are considered in [17].

Fiala *et al.* [9] 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.* [6] for a problem with data imprecision motivated from computational metrology, Cai and Keil [5] for visibility in an imprecise simple polygon, Sellen *et al.* [18] for precision sensitivity, and Yap [19] for a survey on robustness, which deals with computational imprecision rather than data imprecision.

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:

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 the square and circular region. 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. 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? Are they allowed to overlap?

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 you want an upper or a lower bound, or both, on it.

Table 1. Results

goal	measure	model	restrictions	running time
largest	area	line segments	parallel	$O(n^3)$
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	squares	non-intersecting	$O(n^{10})$
largest	perimeter	squares	equal size	$O(n^{13})$
smallest	area	line segments	parallel	$O(n \log n)$
smallest	area	squares	-	$O(n^2)$
smallest	perimeter	line segments	parallel	$O(n \log n)$
smallest	perimeter	squares	-	$O(n \log n)$

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, 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 Sections 3, 4 and 5. First, in the next section, some related issues are discussed.

2 Some Results on Spanning Trees and Tours

In this section we briefly discuss the impact of imprecision on another classical geometric problem, the Minimum Spanning Tree. Then we discuss our results on tours. Due to space limitations, we only give the results and very globally, the ideas needed to obtain them. Details can be found in the full paper [14].

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

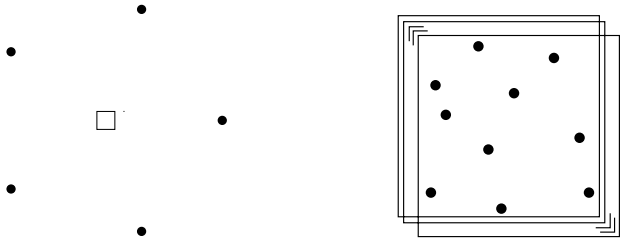


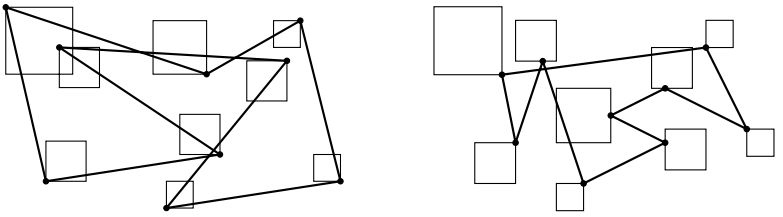
Fig. 1. (a) It is algebraically difficult to find the minimal MST. (b) It is combinatorially difficult to find the minimal MST.

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 both algebraically and combinatorially hard. In Figure 1(a), there are five fixed points and one imprecise one (in the square model, but it could also be a circle). 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 one. Thus the problem reduces to minimizing the sum of the distances from the imprecise point to the fixed points. This is algebraically a hard problem [3]. Furthermore, we can prove NP-hardness of smallest MST by reduction from the Steiner Minimal Tree problem. Given a set of n fixed points P in the plane, we can compute its Steiner Minimal Tree using a solution to the imprecise MST problem as follows. Take P as precise points, and add a set P' of $n-2$ imprecise points whose regions are squares or circles that contain P, see Figure 1(b). The shortest MST of $P \cup P'$ is the Steiner Minimal Tree of P.

Longest Tour. We consider the problem of computing the longest tour that visits a sequence of n axis-parallel squares in a given order. The tour may have self-intersections, see Figure 2(a). We can prove that every vertex of the tour will be at a corner of a square. Given an arbitrary starting square and some vertex v of it, the longest tour up to some vertex w of the i-th square consists of a longest tour to one of the four vertices of the $(i-1)$-st square, and one more segment to w. Hence, the longest tour can be constructed incrementally in $O(1)$ time for each next square. We obtain:

Theorem 1. *Given an ordered set of n arbitrarily sized, axis-aligned squares, the problem of choosing a point in each square such that the perimeter of the resulting polygon is as long as possible can be solved in $O(n)$ time.*

Shortest Tour. Next we study the problem of computing the shortest tour that visits a sequence of n axis-parallel squares in a given order. In this case, vertices of the optimal tour can also be on edges of squares, see Figure 2(b). We can show that the shortest tour can be seen as a combination of two shortest one-dimensional tours, one in the x-projection and one in the y-projection. Therefore we know where the shortest tour changes direction from top to bottom or vice versa, and left to right or vice versa. The shortest tour also satisfies the principle



(a) (b)

Fig. 2. (a) The longest perimeter solution. (b) The shortest perimeter solution.

of reflection, and therefore we can transform the shortest tour problem to a geodesic shortest path problem in a simple polygon (ignoring some details and complications that are handled in the full paper). We obtain:

Theorem 2. *Given an ordered set of n arbitrarily sized, axis-aligned squares, the problem of choosing a point in each square such that the perimeter of the resulting polygon is as short as possible can be solved in $O(n)$ time.*

Largest or Smallest Area Simple Tour. If we require that the resulting tour has no self-intersections, that is, it is a simple polygon, then we can also minimize or maximize the enclosed area. We can show that this problem is NP-hard. The reduction from planar 3-SAT is in the full paper. It is also NP-hard to determine the longest simple tour, but the proof does not extend to the shortest simple tour. We have:

Theorem 3. *Given an ordered set of n arbitrarily oriented line segments, the problem choosing a point on each segment such that the area of the resulting polygon is as large as possible is NP-hard. The same problem for smallest area and for largest perimeter is also NP-hard.*

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, where 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.

Line Segments. The problem we discuss in this section is the following:

Problem 1. *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 Figure 3(a)).*

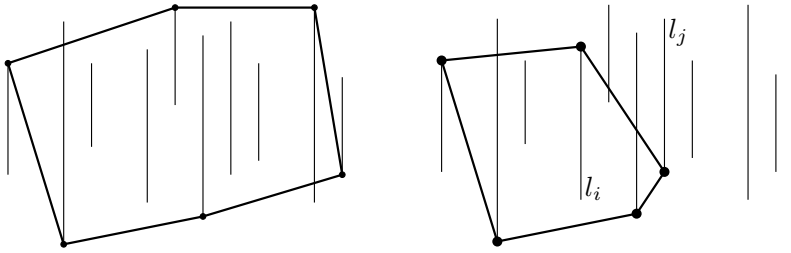


Fig. 3. (a) The largest convex hull for a set of line segments. (b) The polygon p_{ij}.

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 1 such that all points are chosen at endpoints of the line segments.*

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. 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 $i \neq j$, we consider the polygon 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 Figure 3(b). Note that p_{ij} will be convex.

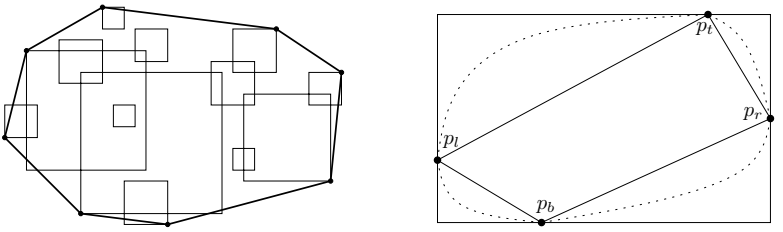
Now, we will show how to compute all p_{ij} using dynamic programming. The solution to the original problem will be either of the form p_{kn} or p_{nk} for some $0 < k < n$, and can thus be computed in linear time once all p_{ij} are known.

When $1 < i < j$, then we can write $p_{ij} = \max_{k<j} \left(p_{ik} + \triangle l_i^+ l_j^- l_k^-\right)$. 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^-. When $1 < j < i$ the expression is symmetric, and $i = 1$ or $j = 1$ is a similar but simpler case. The algorithm runs in $O(n^3)$ time and requires $O(n^2)$ space. We do not need to worry about convexity, because a non-convex solution can never be optimal.

Theorem 4. *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.*

Squares. The problem we discuss in this section is the following:

Problem 2. *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 Figure 4(a)).*



(a) (b)

Fig. 4. (a) The largest area convex hull for a set of squares. (b) The four extreme points.

Observations. Once again we observe that the points will not 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.*

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. The extreme points and the triangles that surround the four chains are shown in Figure 4(b).

Lemma 3. *All vertices on the top left chain are top left corners of their squares, and similar for the other 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 Figure 5(a). 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.

Algorithm for Non-overlapping Squares. When we restrict the problem to non-overlapping squares, we can solve this 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 reuse the algorithm for Problem 1 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 $O(n^7)$.

We call a corner of a square *candidate* if it is in the correct triangle to possibly be part a chain, so for example the top left corner is candidate if it is in the top left triangle, see Figure 4(b). If the squares do not overlap, there can be only two squares that have more than two candidate corners. We ignore these squares (we

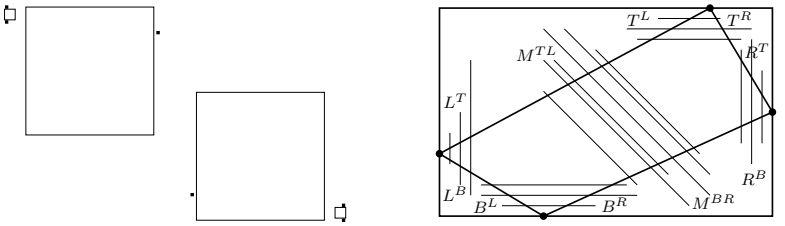


Fig. 5. (a) The four extreme points need not be in the extreme squares. (b) The squares can be divided into five groups of parallel line segments.

just try all possibilities), and note that the rest of the squares all have at most two candidate corners, and can therefore be reduced to line segments. Further note that there are only a limited number of orientations, and those of the same kind are adjacent, as in Figure 5(b). There are six possible kinds of line segments, of which only five may appear at the same time, which implies that we can divide the segments into five groups. The figure is schematic since the segments cannot be extended to non-overlapping squares, but it would require squares of very different sizes to obtain a linear number in each group.

We will now solve the situation of Figure 5(b) in $O(n^3)$ time. 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 idea of the algorithm is to compute, for each pair of endpoints, the optimal solution connecting them via the lower left side. This can be done by reusing the algorithm for the parallel line segment problem, and distinguishing cases for in which group the two points are. The details can be found in [13].

Theorem 5. *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.*

Unit Size Squares. The extra $O(n^4)$ 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 Figure 5(a). 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 $O(n^3)$.

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.*

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, since now there are only a constant number of possibilities for the extreme points.

Theorem 6. *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.*

For overlapping squares, the problem remains open. However, for overlapping squares of equal size, we can solve the problem in $O(n^5)$ time, see [13].

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.

Line Segments. The problem we discuss in this section is the following:

Problem 3. *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.*

Lemma 5. *In the optimal solution, if a line segment defines a vertex of the convex hull, and there are other vertices on the hull strictly on both sides of the supporting line of this segment, then the point on this segment must be chosen at one of the endpoints.*

We denote the leftmost segment by s_l and the rightmost segment by s_r. We define two chains, the *top chain* c_t and the *bottom chain* c_b 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. Symmetrically, the bottom chain is the lower half of the convex hull of the set of all upper endpoints of the input segments, see Figure 6(a). If the top and bottom chains do not intersect, there is a zero

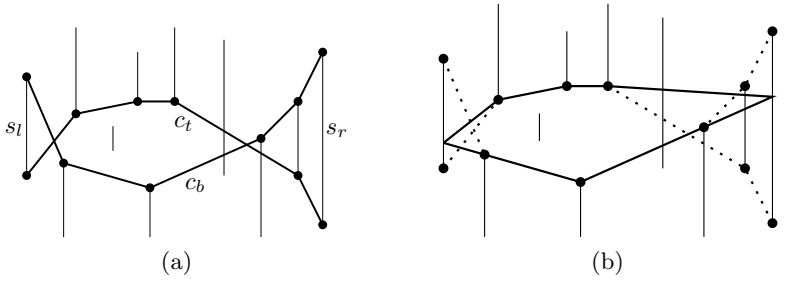


Fig. 6. (a) The top chain c_t and bottom chain c_b. (b) The optimal solution.

area solution that can be found in linear time [8]. Therefore, we assume next that they intersect.

For a point p on s_l, there is a *tangent point* $a_l(p)$ on the top chain such that the line through p and $a_l(p)$ does not go through the region below the top chain. When there are more than one such points we choose the one that lies most to the right. Similarly, we define $b_l(p)$ as the tangent point on the bottom chain, and for q on s_r we define two tangent points $a_r(q)$ and $b_r(q)$ on the top and bottom chains. All those tangent points are vertices of the chains.

Lemma 6. *If the points p on s_l and q on s_r are known, the optimal solution is the polygon that consists of p, $a_l(p)$, the piece of the top chain between $a_l(p)$ and $a_r(q)$, $a_r(q)$, q, $b_r(q)$, the piece of the bottom chain between $b_r(q)$ and $b_l(p)$, $b_l(p)$, and back to p, provided that this polygon is convex. If it is not, then p and q will be connected by a straight line above the top chain or below the bottom chain (see Figure 6(b)).*

Algorithm. We will use these observations to construct an efficient algorithm. First we note that the two chains can be computed in $O(n \log n)$ time using conventional convex hull algorithms, and then we show that we can find the optimal solution using the chains in $O(n)$ time, yielding a total of $O(n \log n)$ time.

To find the location of the points p on s_l and q on s_r, we use the fact that they can be found independent of each other.

Lemma 7. *The individual optimal locations for p and q, minimizing the area of p and q respectively add to the intersection of the chains, are the same as the location of p and q in the optimal solution.*

The important point is that, in the optimal solution, p and q will never be connected directly to each other, but always via the chain. The individual solutions can be computed in linear time, after the chains are known. The computation of the chains takes $O(n \log n)$ time.

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.*

Squares. The problem we discuss in this section is the following:

Problem 4. *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 Figure 7(a)).*

Lemma 8. *In the optimal solution, only the leftmost, rightmost, topmost, and bottommost vertices of the hull need not be corners of their squares.*

The situation is similar to the line segment case. There are now four *extreme squares* S_l, S_r, S_t and S_b, and for these four squares, the points must lie on

the inner edge. We call the points p_l, p_r, p_t and p_b. We now have four chains of corners that could be included in the convex hull, see Figure 7(b). The optimal solution for fixed p_l, p_r, p_t and p_b connects these points to their tangent points on the chains or directly to each other if the result would not be convex.

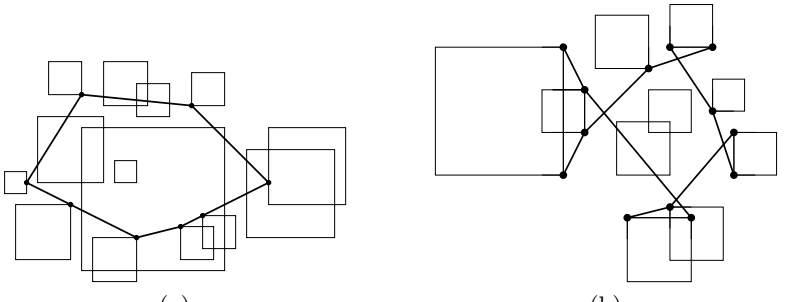
The critrical difference between the line segment case and the square case, is that the locations of the four extreme points are no longer independent. It can really happen that in the optimal solution two or more of the extreme points are connected by straight line segments, rather than via the chains. This means we need a different approach to solve the problem, and is the reason why this variant cannot be easily solved in $O(n \log n)$. We describe a case distinguishing algorithm that runs in $O(n^2)$ time in [13].

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, such as the perimeter. In this section we will briefly consider the relevant differences between the two measures.

One important observation concerns the way the size of a polygon changes when only one point is moving, while the rest remains fixed. The area of the polygon will be a linear function of the moving point, while the perimeter is a hyperbolic function with a minimum. In the case of convex hulls, this only applies as long as the combinatorial structure of the hull does not change. Secondly, note that 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 explicitly taken care of. 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.



(a) (b)

Fig. 7. (a) The smallest convex hull for a set of squares. (b) The top left, bottom left, top right, and bottom right chains.

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 [13].

6 Conclusions

We studied the problem of computing the largest or smallest convex hull of a set of imprecise points; our results are in Table 1. The problem of finding the smallest convex hull seems to be easier than finding the largest convex hull: the running times are better, and there are fewer restrictions. It also seems that for the largest convex hull the area is easier to maximize than the perimeter, while for the smallest convex hull the perimeter is easier to minimize than the area.

Many problems are 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 regions are allowed to intersect? Also, what results can be obtained for the circle model? For the problems that do not have an efficient solution, the study of approximation algorithms is interesting. Thirdly, for many other problems in computational geometry, imprecision in the data and the bounds on the effect on the outcome of an algorithm should be studied.

References

1. M. Abellanas, F. Hurtado, and P. A. Ramos, Structural tolerance and Delaunay triangulation, *Inf. Proc. Lett.* 71:221–227, 1999.
2. D. Bandyopadhyay and J. Snoeyink, Almost-Delaunay simplices: nearest neighbour relations for imprecise points, *Proc. 15th ACM-SIAM Sympos. on Discr. Algorithms*, pages 410–419, 2004.
3. C. Bajaj, The algebraic degree of geometric optimization problems, *Discr. Comput. Geom.*, 3:177–191, 1988.
4. J.-D. Boissonnat and S. Lazard, Convex hulls of bounded curvature, *Proc. 8th Canad. Conf. on Comput. Geom.*, pages 14–19, 1996.
5. L. Cai and J. M. Keil, Computing visibility information in an inaccurate simple polygon, *Internat. J. Comput. Geom. Appl.* 7:515–538, 1997.
6. M. de Berg, H. Meijer, M.H. Overmars, and G.T. Wilfong, Computing the angularity tolerance, *Int. J. Comput. Geometry Appl.* 8:467–482, 1998.
7. M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell, Touring a Sequence of Polygons, *Proc. 35th ACM Sympos. Theory Comput.*, 2003
8. H. Edelsbrunner, Finding transversals for sets of simple geometric figures, *Theor. Comp. Science* 35:55–69, 1985.
9. J. Fiala, J. Kratochvil, and A. Proskurowski, Systems of distant representatives, *Discrete Applied Mathematics* 145:306–316, 2005.
10. M. T. Goodrich and J. Snoeyink, Stabbing parallel segments with a convex polygon, *Computer Vision, Graphics, and Image Processing* 49:152–170, 1990.

11. L. Guibas, D. Salesin, and J. Stolfi, Constructing strongly convex approximate hulls with inaccurate primitives, *Algorithmica* 9:534–560, 1993.
12. A. A. Khanban and A. Edalat, Computing Delaunay triangulation with imprecise input data, *Proc. 15th Canad. Conf. on Comput. Geom.*, pages 94–97, 2003.
13. M. Löffler and M. van Kreveld, *Largest and Smallest Convex Hulls for Imprecise Points*, Technical Report UU-CS-2006-019, Department of Computing and Information Science, Utrecht University, 2006.
14. M. Löffler and M. van Kreveld, *Largest and Smallest Tours for Imprecise Points*, manuscript, 2006.
15. T. Nagai and N. Tokura, Tight error bounds of geometric problems on convex objects with imprecise coordinates, *Jap. Conf. Discr. Comp. Geom.*, 252–263, 2000.
16. Y. Ostrovsky-Berman and L. Joskowicz, Uncertainty Envelopes, *Abstracts 21st European Workshop on Comput. Geom.*, pages 175–178, 2005.
17. V. Polishchuk, J. S. B. Mitchell, Touring Convex Bodies - A Conic Programming Solution, *Proc. 17th Canad. Conf. on Comput. Geom.*, pages 290–293, 2005.
18. J. Sellen, J. Choi, and C.-K. Yap, Precision-sensitive Euclidean shortest paths in 3-space, *SIAM J. Comput.* 29:1577–1595, 2000.
19. C.-K. Yap, Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*, Chapman & Hall/CRC, 2004. Chapter 41, pages 927–952.