TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

Finding Structures On Imprecise Points

By
M.J.M. Roeloffzen

Supervisors:

prof. dr. M.T. de Berg
dr. E. Mumford PDEng

Eindhoven, July 2009

Abstract

An imprecise point is a point in $\mathbb{R}^2$ of which we do not know the location exactly; we only know that it is somewhere within a region in $\mathbb{R}^2$. On such a set of imprecise points structures like the closest pair or convex hull are not uniquely defined. This leads us to study the following problem: Given a structure of interest, a set $\mathcal{R}$ of regions and a subset $\mathcal{C} \subseteq \mathcal{R}$, we want to determine if it is possible to place a point in each region of $\mathcal{R}$ such that points in regions of $\mathcal{C}$ form the structure of interest. We study this problem for the convex hull and closest pair and for various types of regions. For each variant we either give a NP-hardness proof or a polynomial-time algorithm.

Contents

Chapter 1

Introduction

Many geometric problems involve finding a structure in a set of points in $\mathbb{R}^2$, such as the *convex hull* or the *closest pair* of a set of points. These structures are well defined for any given set of points and many algorithms are available for computing these structures efficiently [BCKO08].

Most real-world data however is not exact. It often comes from finite-precision computations or imprecise measurements, such as GPS coordinates. Even though such data specifies coordinates for each point we do not know the exact location. We only know that for each point that it is within a region around the given coordinates. This imprecision can be modeled using a region in space for each point. For example discs can be used to model the imprecision caused by measurement errors, whereas rectangles can be used to model imprecision on the coordinates caused by finite precision computations. The input is then a set of regions in $\mathbb{R}^2$, where inside each region a point is located. Now different placements of points inside their regions can lead to different convex hulls and closest pairs on those points. Thus the question is what we want to compute on a set of these imprecise points and how.

Related work. To deal with computational imprecision Guibas et al. introduced a framework called *epsilon geometry* which is based on interval arithmetic and backward error analysis [SSG89]. In epsilon geometry predicates on geometric objects are accompanied by a parameter $\varepsilon > 0$ to indicate how much the objects may have to be perturbed to make the predicate true. For example two points *ε-coincide* if there is a point within ε distance from both points. A minus sign is used to indicate that a property remains intact even when all points are moved by a certain distance. A polygon is *-ε-convex* if the polygon is convex for every perturbation of its vertices by at most ε. They then show how to use these predicates to determine if a point is *δ-inside* an ε-convex polygon for $\delta > 0$. In a different paper [GSS90] they use the framework to compute a *strongly convex hull* of a set S of points, which is a -ε-convex polygon in which every point of S is δ-inside for $\delta > 0$.

Nagai and Tokura show how the union and intersection of all possible convex hulls on a set of imprecise points can be computed [NT01].

In light of the fact that different placements lead to different convex hulls Löffler and van Kreveld try to find the placement of points which maximizes or minimizes the area or perimeter of the convex hull [LK08]. They describe algorithms for this with running times ranging from $O(n \log n)$ to $O(n^{13})$ for various restrictions on the input regions. They also prove NP-hardness for finding the convex hull with the largest area and for finding the convex

hull with largest perimeter. They also did similar work for the smallest bounding box, smallest enclosing circle, width, diameter and closest pair of a set of imprecise points [LK07].

Fiale et al. introduce the notion of systems of *distant representatives* [FKP02]. Given a set $\mathcal{R}$ of sets of points and a distance d, a system of d-distant representatives is a choice of points, one from each set in $\mathcal{R}$, such that the distance between any two points is at least d. They then show that determining if such a system exists is NP-hard when the sets of $\mathcal{R}$ are unit discs and are allowed to overlap. Cabello proposes an approximation algorithm for this problem [Cab07].

Both Löffler and van Kreveld [LK08] and [LK07] and Fiala et al. [FKP02] focus on some numerical values, such as the area or distance, of the structures that can be made from a set of imprecise points. They do not consider the combinatorial properties. This is the focus of our work: To determine if a set of imprecise points can induce a structure with certain combinatorial properties. For example, given a set S of imprecise points, is it possible that two given points $p, q \in S$ form the closest pair. Next, we define the problem more precisely.

Problem description. In general the problem we study is defined as follows. We have a structure of interest, such as the convex hull or closest pair, a set of regions $\mathcal{R}$ in $\mathbb{R}^2$ and a subset $\mathcal{C} \subseteq \mathcal{R}$. The question we then want to answer is: Is it possible to place a single point in each region of $\mathcal{R}$ such that the points placed in regions of $\mathcal{C}$ form the structure of interest. If such a placement is possible we say that $\mathcal{C}$ is a *possible structure*. For example the structure of is the closest pair and the input is a set $\mathcal{R}$ of rectangles and a subset $\mathcal{C} \subseteq \mathcal{R}$ which contains two rectangles P and Q. Then $\mathcal{C}$ is a possible closest pair if and only if it is possible to place a point in each region of $\mathcal{R}$ such that the points $p \in P$ and $q \in Q$ are a closest pair.

We use sets of line segments, sets of squares or sets of discs as the input set $\mathcal{R}$. For different structures we may also impose some additional constraints on these regions, for example that regions have to be disjoint. For the rest of this document $\mathcal{R}$ will be used to denote the input set of regions and $\mathcal{C}$ to denote the subset for which we want to determine if it is a possible structure. The structures that we will look at are the closest pair and the convex hull.

Results. For the convex hull we introduce the notions of an *exact convex hull*, a *subset convex hull* and a *superset convex hull*. In an exact convex hull the points from regions of $\mathcal{C}$ are exactly its vertices, whereas for the subset (and superset) convex hull the points from regions in $\mathcal{C}$ are a subset (or superset) of the vertices.

The results for the decision problem to determine if it is possible to place a point in each region such that the points from $\mathcal{C}$ form a given structure are summarized in Table 1.1. Note that n is the number of regions in $\mathcal{R}$ and k is the number of regions in $\mathcal{C}$.

The NP-hardness proofs for the closest-pair problems are based on a reduction from the problem of finding distant representatives to the problem of determining if two regions are a possible closest pair and can be found in Chapter 2. We also show in Section 2.4 that counting the number of possible closest pairs is NP-hard. It is then shown that there are only $O(n)$ possible closest pairs when the regions are disjoint unit discs and we give a sweep line algorithm that runs in $O(n \log n)$ time and reports $O(n)$ pairs where all possible closest pairs are among the reported pairs. By placing a point in each region in a fairly simple manner it is also possible to determine for some pairs that they are in fact a possible closest pair. These pairs can also be found using a sweep line algorithm that runs in $O(n \log n)$ time.

Lastly the results for the exact, subset and superset convex hull problems can be found in Chapters 3, 5 and 4 respectively.

Regions	**Restrictions**	**Solution**
Possible Closest Pair		
line segments	disjoint, unit length, parallel	NP-hard
squares	disjoint, unit size, axis aligned	NP-hard
discs	disjoint, unit size	NP-hard
Exact Convex Hull		
line segments	none	NP-hard
Superset Convex Hull		
line segments	none	NP-hard
line segments	disjoint, unit size, parallel	$O(k^2 n \log n)$
Subset Convex Hull		
line segments	disjoint, parallel	$O(n \log n + k^2)$
rectangles	disjoint, axis aligned	$O(n \log n + k^{4+\varepsilon})$

Table 1.1: Results on the decision problems.

Preliminaries. Given a set $\mathcal{S}$ of regions, a choice of points, one from each region of $\mathcal{S}$ will simply be called a *placement* (of points in $\mathcal{S}$). Unless otherwise indicated, a placement refers to a choice of points from the regions of the input set $\mathcal{R}$.

As mentioned before, $\mathcal{C}$ is said to be a *possible structure* if and only if there is a placement of points in which the points from $\mathcal{C}$ are exactly those that contribute to the structure. Similarly $\mathcal{C}$ is an *impossible structure* if for every possible placement $\mathcal{C}$ does not form the structure.

To avoid confusion between sets of regions, regions and points we will use calligraphic letters $(\mathcal{R}, \mathcal{C})$ to indicate sets of regions, capital letters (P, Q) to indicate regions and lower case letters to indicate points (p, q). Furthermore when a point corresponds to a certain region, the region and point will be indicated by the same letter $(p \in P, q \in Q)$.

Chapter 2

Closest Pair

A *closest pair* of a set S of points is a pair $p, q \in S$ that have the shortest distance between any pair of points in S, that is, a pair such that $dist(p, q) = \min\{dist(r, s) \mid r, s \in S \wedge r \neq s\}$ where $dist(p, q)$ is the distance between p and q. A closest pair does not have to be unique; there can be multiple pairs of points that have the same distance between them. In terms of imprecise points the closest pair problem is the following.

Possible Closest Pair
Input: A set of regions $\mathcal{R}$ and a pair $P, Q \in \mathcal{R}$.
Output: YES if $\{P, Q\}$ is a possible closest pair, NO otherwise.

In this section it is shown that determining if a certain pair of regions can form the closest pair is NP-hard for regions modeled as discs, squares or line segments, using a reduction from the problem of finding distant representatives, which was introduced by Fiala et al. [FKP02]. The problem of finding distant representatives is defined as follows.

Distant Representatives
Input: A set of regions $\mathcal{R}$ and a distance d.
Output: YES if a point can be placed in each region of $\mathcal{R}$ such that $dist(p, q) \geq d$ for any two points $p \in P$ and $q \in Q$, where $P, Q \in \mathcal{R}$, NO otherwise.

In the rest of this section we will be using two different distance measure between regions. Let P and Q be two regions then the minimal distance between two regions is defined as follows $dist^-(P, Q) = \min\{dist(p, q) \mid p \in P, q \in Q\}$. Analogous the maximal distance between two regions is $dist^+(P, Q) = \max\{dist(p, q) \mid p \in P, q \in Q\}$. Now we can show that from an instance of Distant Representatives an instance of Possible Closest Pair can be created that returns YES if and only if the instance of Distant Representatives returns YES.

Theorem 1. Distant Representatives *can be reduced to* Possible Closest Pair *in polynomial time.*

Proof. Let $\mathcal{R}$ be the input set for Distant Representatives and d the minimum distance separating any two points. Add two regions P and Q such that $dist^-(P, Q) = d$ and for every region $S \in \mathcal{R}$ we have $dist^-(S, P) \geq d$ and $dist^-(S, Q) \geq d$. This can be done by first constructing regions P and Q of the required type, for example squares or discs. Then compute the position of Q with respect to P by vertically aligning a lowest point p of P with

a highest point q of Q and placing q a distance d below p. Then we can compute a leftmost point r in these two regions and a rightmost point s of the regions of $\mathcal{R}$. Then place P and Q such that r is horizontally aligned with s and $dist(r, s) = d$.

If P and Q are a possible closest pair, then there is a placement in which all points from regions in $\mathcal{R}$ have at least distance d between them.

If P and Q are not a possible closest pair then there is no placement such that the point from regions in $\mathcal{R}$ have distance d between them. □

Next we show for three different types of regions that Distant Representatives is NP-hard. It then follows from Theorem 1 that also Possible Closest Pair is NP-hard for those types of regions.

Later in this section we show that it is also NP-hard to count the number of possible closest pairs. However for unit discs only pairs of discs for which the minimal distance is at most some easily computable value can be possible closest pairs. We present an algorithm that uses this to find a linear number of pairs of discs where all possible closest pairs are amongst the pairs that are found. Lastly we show that for a pair of unit discs whose minimal distance is at most some easily computable value a simple placement exists in which the points from this pair are the closest pair.

2.1 Squares

In this section we will show that Possible Closest Pair is NP-hard when the input is a set of disjoint axis aligned squares of unit size. We prove this using a reduction from planar 3-SAT.

A propositional formula is satisfiable if there is a valuation of its boolean *variables* that makes the formula true. A SAT formula is a conjunction of *clauses* where each clause is a disjunction of *literals* and each literal is either a variable or negated variable. In a 3-SAT formula all clauses contain exactly three literals. From a 3-SAT formula we can construct a graph where the variables and clauses are nodes and there is an edge between a clause node and a variable node for each occurrence of the variable in that clause. In planar 3-SAT this graph is planar and can be drawn without crossings such that all variable nodes are on one line and the clause nodes are to the left and right of this line. For the remainder of this section the following formula, ϕ, will be used as a running example. The graph representation of ϕ is shown in Figure 2.1.

$$\phi = (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee d) \wedge (\neg b \vee c \vee \neg d) \wedge (a \vee \neg c \vee \neg d)$$

Now we can define planar 3-SAT as follows.

Planar 3-SAT
Input: A planar 3-SAT formula with a planar embedding.
Output: YES if the formula is satisfiable, NO otherwise.

To make a reduction from Planar 3-SAT to Distant Representatives we have to generate a set of regions $\mathcal{R}(\psi)$ from a 3-SAT formula ψ, such that ψ is satisfiable if and only if there is a placement for $\mathcal{R}(\psi)$ in which every two points have at least distance d between them. Constructing $\mathcal{R}(\psi)$ is done by creating a set of regions for each node and edge in the

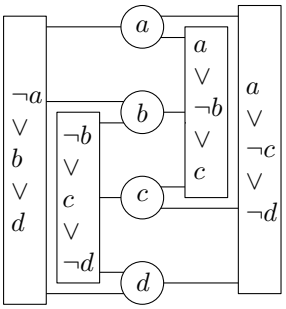


Figure 2.1: Planar embedding for the 3-SAT formula ϕ.

graph representation of ψ. The union of all these sets is $\mathcal{R}(\psi)$. These sets are called *gadgets* and there are three different kinds of them, the *variable gadget*, the *clause gadget* and the *edge gadget*. First a short description of the gadgets is given and then we explain them in more detail. Then lastly we explain how to compute the gadgets from a given planar 3-SAT formula and its graph.

- *Variable gadget.* The variable gadget represents a boolean variable. It allows exactly two different placements which correspond to the true and false values of the boolean variable.
- *Edge gadget.* The edge gadget propagates the value of a literal from a variable gadget to a clause gadget. It is a sequence of squares which starts with a square in a variable gadget and ends with a square in a clause gadget. Since values correspond to placements, the edge gadget should only allow placements where the first square and the last square in the sequence correspond to the same value.
- *Clause gadget.* The clause gadget is a fixed configuration of squares. It has the property that if the three incoming edge gadgets it is connected to have a placement corresponding to false then there is no placement of points in the clause gadget with distance d between the points. If at least one of the incoming edge gadgets has a placement corresponding to true then such a placement does exist.

2.1.1 Variable gadget.

The basis of the variable gadget are two vertically aligned squares which are placed such that the distance between two opposite corners is exactly d. Then the only possible placements are a point on the top left corner of the top square and a point on the bottom right corner of the bottom square or a point on the top right corner of the top square and the bottom left corner of the bottom square. This can then be extended to a sequence of squares by adding squares, again vertically aligned, below the bottom one such that the distance between the bottom left corner of one square and bottom right corner of the next is exactly d. This is illustrated in Figure 2.2a.

The two placements for the entire variable gadget correspond to the two values of the variable. Each edge gadget has one square in a variable gadget. A placement in this square is considered to correspond to a false value of the literal that the edge represents if a point in that square is placed towards the clause that the edge is connected to. For an edge gadget

representing a non-negated literal which is to the right of the variable gadget the placement corresponding to a false value for that literal has a point placed to the right. When the variable is true then the non-negated literals are also true, so a true placement of the variable should also imply a true placement for the square that is part of the edge gadget of a non-negated literal. As can be seen in Figure 2.2b for edges on the right side they can only start at odd squares when starting to count from zero at the top square whereas on the left side they can only start at even squares. Edges corresponding to negated literals however should start at even squares on the right side and odd squares on the left. Figure 2.5 shows $\mathcal{R}(\phi)$, the set of squares corresponding to the formula ϕ. It is a good illustration of how edge gadgets are connected to the variable gadgets which are market by grey rectangles.

The number of squares in the variable gadget depends on the number of edges connected to a variable node. Because we don't want different edge gadgets to influence each others placements, other then through a clause of variable gadget, there should be enough space between squares of different edge gadgets. Therefore there should be at least two squares between two squares of the variable gadget that are part of an edge gadget. If we combine that with the fact that an edge can only connect to either the odd or even squares on one side we may need $4k$ squares for a variable node with k edges. Note that in the example of Figure 2.5 edges are placed closer together to keep the example small, but this may cause problems for the set of regions corresponding to other formulas.

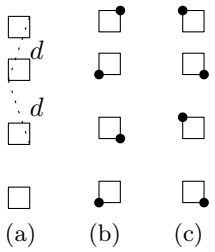


Figure 2.2: A variable gadget without placement (a), with a placement corresponding to true (b) and with a placement corresponding to false (c)

2.1.2 Edge gadget

The edge gadget is a sequence of squares $S_1 \dots S_k$ where S_1 is also part of a variable gadget and S_k is part of a clause gadget. We consider a placement of S_1 on the bottom corner towards the clause to correspond to false and the bottom corner away from the clause to correspond to true. We define true and false placements for S_k in the same way, bottom corner towards the clause means the literal is false and bottom corner away from the clause means true. The sequence of squares is *x-monotone*. This means that for an edge going from a variable to a clause on the right, each square S_i will be completely to the right of S_{i-1}. Because the gadget for an edge going from right to left is symmetric to an edge going from left to right, we assume the gadget goes from left to right.

The edge gadget should propagate the value of a literal from a variable gadget to a clause gadget. We can achieve this by only allowing placements where either s_1 and s_k are in a true position or both are in a false position. However we can relax this requirement a little. If s_1 is in a true position and s_k in a false position then this only makes it harder to find a placement

for the clause gadget. So instead we only require that if s_1 is in a false position then s_k must also be and that if s_1 is in a true position there has to be some placement where s_k is also in a true position.

To ensure that s_k is in a false position if s_1 is also in false position we will ensure that for each S_i its false position s_i^f is exactly distance d from s_{i-1}^f and for every point $s_i \in S_i$, such that $s_i \neq s_i^f$, we have $dist(s_i, s_{i-1}^f) < d$. How this is achieved depends on the direction of the edge.

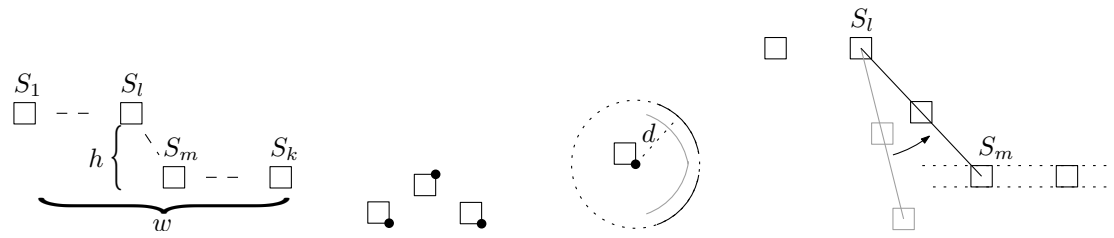
The edge gadget can be split into three parts as shown in Figure 2.3a. We define $\mathcal{E}_1 = \{S_1 \dots S_l\}, \mathcal{E}_2 = \{S_{l+1} \dots S_m\}$ and $\mathcal{E}_3 = \{S_{m+1} \dots S_k\}$. In the $\mathcal{E}_1$ and $\mathcal{E}_3$ squares are horizontally aligned and spaced such that a top right corner of one square is exactly distance d from the bottom right corner of the next, like the squares in a variable gadget are spaced vertically. We need to make sure that the edge gadget can have a certain width, as indicated by w, where the width of a sequence of squares is the horizontal distance between the lower right corner of the first square in the sequence and the lower right corner of the last square. We will do this by making the width of $\mathcal{E}_3$ variable. By only placing horizontally aligned squares only widths that are multiples of d' can be achieved, where d' is the distance between two lower right corners of consecutive squares in the sequence of horizontally aligned squares. To give $\mathcal{E}_3$ a width l that is not divisible by d' we start with a sequence of $\lceil l/d' \rceil$ horizontally aligned squares. Then we make the sequence shorter by lifting a square S_i out from the sequence such that the previous and next square can move closer together while still maintaining that s_i^f is the only point in S_i that is is exactly distance d from s_{i-1}^f and s_{i+1}^f. This is illustrated in Figure 2.3b. Note that this can also be used to ensure that the false position of s_k is the bottom right corner and not the top right corner by adding an extra square if needed and then making the edge shorter by lifting squares out to get the right width to connect the edge to the clause. How to compute what this width is will be discussed in Section 2.1.4.

Then there is $\mathcal{E}_2$ that is used to compensate for a difference in height h between S_1 and S_k, where the height of a sequence is the vertical distance between the lower right corners of the first and last square. A square can be placed higher or lower than the previous square in the sequence by making sure that the whole square except a corner is inside a circle with radius d from the previous false position, as illustrated in Figure 2.3c. Say that $\mathcal{E}_3$ has to go down by some distance h then we can compute how many squares we need at least to cross this height, namely using an as steep as possible slope while maintaining that the sequence is x-monotone. Then using the maximal slope we may actually cross more than height h, but since the number of squares in the sequence is now known we can compute which slope we need to ensure that S_m is at the right height as illustrated in Figure 2.3d.

Because the sequence of squares is x-monotone it is easy to see that there is a placement where S_k has a true placement if S_1 has a true placement. Simply take the placement where all points are in false position so at the top right or bottom right corner of their squares and then shift all points to the left corner. The distances between the points remain the same since they are all shifted over the same distance in the same direction, so the placement still has at least distance d between all points, but now both s_1 and s_k are in a true position.

2.1.3 Clause gadget

The clause gadget is a fixed configuration of squares and is shown in Figure 2.4a. Black double arrows indicate that the distance between the two endpoints of the arrow is exactly d.



(a) Three segments of edge gadget

(b) Shortening a horizontal segment

(c) Placing squares under different slopes

(d) Finding the right slope

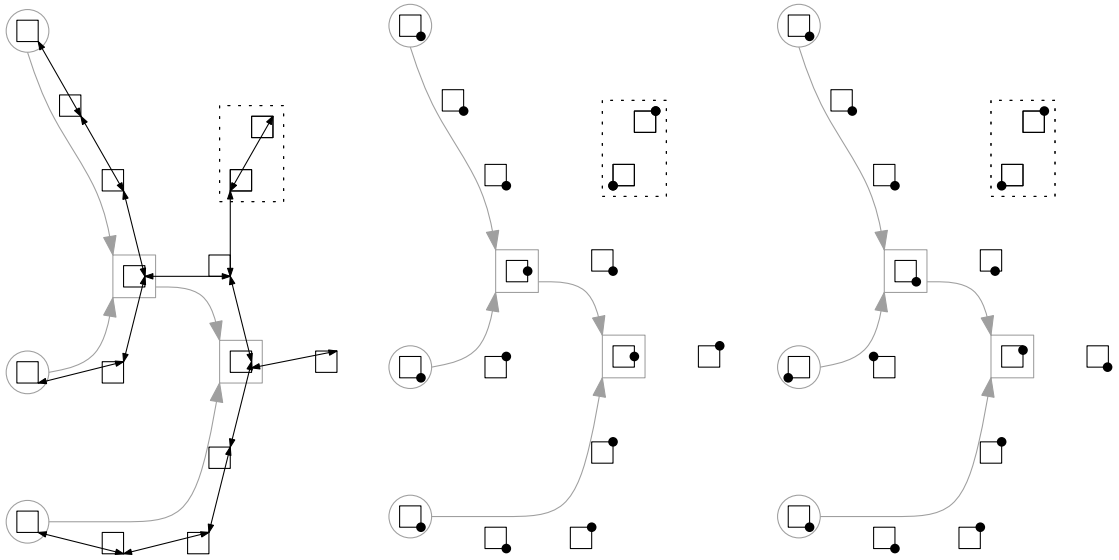
Figure 2.3: Constructs used for building an edge gadget

Squares with grey circles around them are the squares that are also part of an edge gadget and are called the *incoming squares*. Square with grey squares around them are *merge squares* where the values of two incoming squares are combined. The arrows indicate the *frame* along which values are propagated between squares. Lastly the two squares with a dashed rectangle around them are a small construct outside the frame which only allows one placement, namely with points on opposite corners. This structure is used to limit the possible placements for a point in the square below it.

The clause gadget has two important properties. The first is that if all points on the incoming squares are in a false position, that is at the bottom right corners, then no placement with distance d between points is possible for the clause gadget. If all points at the incoming squares are indeed placed at their lower right corners then the squares along the arrows to the merge squares have only one placement with distance d. This is shown in Figure 2.4b. This is because every next square along the arrows has only one point at distance d from the placement of the previous square. Then the point in the top merge square can only be placed in the middle of its right edge. The point on the square between the two merge squares can then only be placed at the lower right corner. The point on the lower merge square must then be placed in the middle of the right edge. Then all points in the square on the right have a distance smaller then d from the point in the lower merge square so there is no valid placement for this point. Hence, there is no placement with distance d possible if all incoming squares have false placements.

The second property is that if at least one incoming square has a true placement, that is in the lower left corner, then a placement for the clause gadget is possible. First we look at the case where the middle incoming square has a point in true position and the top and bottom incoming square a point in false position. The placement of points of squares along the arrows from the bottom and top incoming squares to the merge squares remains the same. The point on the square between the middle incoming square and the top merge square can be placed at the lower left corner now. The point on the top merge square can then be placed at the lower right corner. The point on the square between the merge squares can then be placed a little to the left and the point on the lower merge square can be placed a little higher. Then the lower right corner of the square to the right of the lower merge square has distance more then d from the point on the merge square so a point can be placed there. The resulting placement is shown in Figure 2.4c. A placement when either the top or bottom incoming square has a point in true position is similar, point along the arrows are moved back and then the points in the merge square can be moved either up or down.

It is easy to see that if more then one incoming square has a true placement a placement for the clause gadget is also possible, since it only allows more freedom to place the points.



(a) Squares of the clause gadget, black arrows indicate a distance d

(b) Placement for a clause that is false

(c) Placement for a clause that is true

Figure 2.4: The clause gadget and a false and true placement, grey elements are not part of the gadget.

2.1.4 Computing gadgets from a formula

With these gadgets we can construct the set $\mathcal{R}(\psi)$ from a planar 3-SAT formula ψ and the planar embedding of its graph representation $G(\psi)$. For the formula ϕ the set $\mathcal{R}(\phi)$ is shown in Figure 2.5.

First we compute the squares for the variable gadgets. The number of edges for each variable can be extracted from the graph. In the graph representation all variables occur on one line so we vertically align all variable gadgets such that they are in the same order as in the graph. The distance between the lowest point of one gadget and the highest point of the next should be at least the height of a clause gadget.

Next the edge gadgets will be computed. Again from $G(\psi)$ we can get the order in which the edges are on the right and left side of a variable node. From this order and the literal the edge represents we compute which squares in the variable gadget start an edge. The edges can be stored in two lists, one on the right side and one on the left. An edge gadget that is not yet connected to a clause gadget is called an *open edge* whereas an edge that is connected to a clause gadget is called a *closed edge*. From this list we take three open edges that will have to connect to the same clause and have no open connections between them in the list. Such three edges always exist because the graph representation is planar and the variables are in the same order as in the planar embedding. Without loss of generality assume these three gadgets go to the right of the variables. The edge gadgets are then constructed as follows. First all three are extended horizontally until the rightmost square of the edge gadgets is at

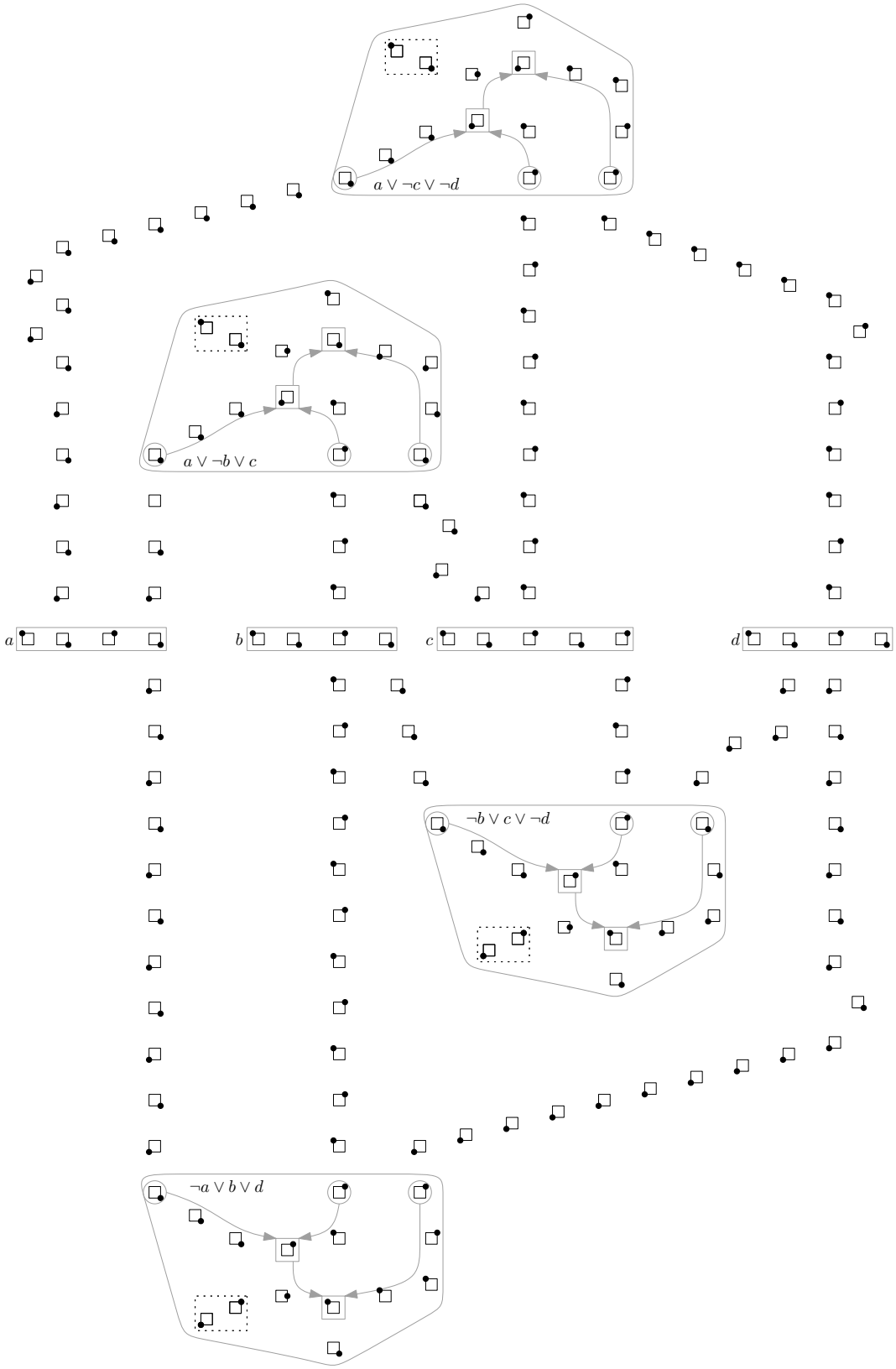


Figure 2.5: The set $\mathcal{R}(\phi)$ of squares for the planar 3-SAT formula ϕ. The placement corresponds to the valuation where all variables are true. Note that the figure is rotated 90°

least is distance d to the right of the squares of any other gadget that is constructed so far. The clause will be placed such that the middle connection is already at the right y-coordinate. Since the clause is a fixed configuration the y-coordinates for the top and bottom connection can then be computed. Then we can construct the sloped part of the edge to get the edge gadget at the right y-coordinate. Lastly we add another horizontal part to get last squares and possibly lift out squares to give this horizontal part the right length. Then the clause can be placed at the end of the edges. These three edges are then closed and we can find another three open edges.

Lemma 1. *The set $\mathcal{R}(\psi)$ of squares of a planar 3-SAT formula ψ has a placement where all points have at least distance d between them if and only if ψ is satisfiable.*

Proof. First note that the squares in different edge gadgets do not influence each others placements other then through the variable or clause gadget. This is because we leave at least two squares between the starting squares of different edge gadgets.

If we assume that ψ is not satisfiable then for each valuation V_u of the variables there is a clause C of which all literals are false. If we place points in the variable gadgets such that they correspond to the valuation V_u, then for the clause gadget representing C the first square of all the edge gadgets connected to this clause gadget have a false placement. Then by construction of the edge gadgets the only valid placement will have the last square of the edge gadgets in a false position as well. Then there is no placement possible with distance d between points in the clause gadget, so also not for $\mathcal{R}(\psi)$. Because every placement of points in the variable gadgets corresponds to some valuation V, every placement of the variable gadgets leads to some clause not having a placement with distance d between points, hence there is no placement with distance d between points for $\mathcal{R}(\psi)$.

If ψ is satisfiable then there is a valuation V_s for which all clauses have at least one literal that is true. By placing the points in the variable gadget according to V_s it is guaranteed that for every clause there is at least one edge gadget where the first square has a true placement. Then we also know there is a placement for the edge gadgets such that there is an incoming square in each clause gadget that has a true placement. In every clause gadget there is then a placement for which the points have distance d from each other. Because squares do not interfere with each others placements other then at the connection points all points in $\mathcal{R}(\psi)$ have at least distance d between them. □

To be able to use this to show that Distant Representatives is NP-hard for squares it also has to be shown that $\mathcal{R}(\psi)$ can be constructed in polynomial time.

Lemma 2. *The set $\mathcal{R}(\psi)$ of regions can be constructed in polynomial time from a planar 3-SAT formula ψ.*

Proof. Placing a single square in a gadget can be done in $O(1)$ time, so the only issue is how many squares there are in $\mathcal{R}(\psi)$. The number of squares in the variable gadget is linear in the number of literals n in ψ. Because d is just a constant that also means that the distance between the highest and lowest square is $O(n)$. Each edge is first extended past all other squares that are already placed and then adds something to the width the bounding box of the squares that are placed so far. The last horizontal part only adds a constant number of squares, however the diagonal section may add $O(n)$ to the width since $O(n)$ squares may be needed to cross the difference in height between the start and end of an edge and these squares are x-monotone. Since there are $O(n)$ clauses the width of the bounding box of all

squares is $O(n^2)$. Each edge can then only contain $O(n^2)$ squares. Lastly the clause gadget only has a constant number of squares.

There are $O(n^3)$ squares and each can be placed in $O(1)$ time, so $\mathcal{R}(\psi)$ can indeed be computed in polynomial time in n. □

From Lemma 1 and 2 we can then conclude the following Theorem.

Theorem 2. Distant Representatives *is NP-hard for regions that are disjoint axis aligned squares of unit size.*

From this and Theorem 1 we conclude the following corollary.

Corollary 1. Possible Closest Pair *is NP-hard for regions that are disjoint axis aligned squares of unit size.*

2.2 Discs

For unit discs that may intersect NP-hardness of Distant Representatives was proven by Fiala et al. [FKP02], but this leaves the problem open for disjoint unit discs. In this section we prove NP-hardness of Distant Representatives for disjoint unit discs using in a similar way as for squares in Section 2.1, namely using a reduction from Planar 3-SAT. The only difference are the gadgets, which are now sets of discs instead of squares. Putting the gadgets together is done in the same way as for squares, so we will only describe the gadgets.

2.2.1 Variable gadget

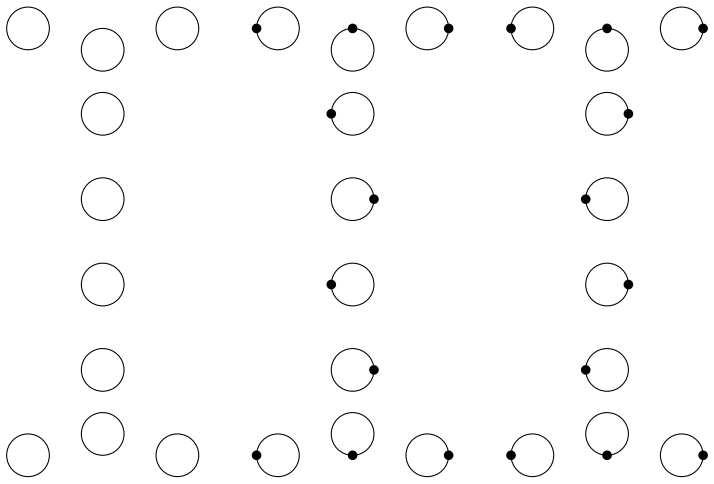
The variable gadget is a sequence of vertically aligned discs spaced such that for two consecutive discs it holds that the distance between the rightmost point of one disc and the leftmost point of the other is exactly d. On each end of the sequence three extra discs are placed to make sure points can only be placed on the right and leftmost points of the discs in the sequence. A variable gadget with four discs in the sequence in shown in Figure 2.6a. The two possible placements, one corresponding to true and one to false have points alternating between the left and rightmost point of the discs in the sequence, as shown in Figure 2.6b and 2.6c.

The discs in the sequence are the discs that can be used in an edge gadget. A true placement for the literal of that edge gadget is then a placement with the point away from the clause. So for an edge going to the right the true position is the leftmost point of the disc. The length of the sequence again depends on the number of edges that the variable node has in the graph representation. Also to make sure that discs in different edges don't influence each others placements directly there should be at least four discs between two discs that are part of two edge gadgets. Combining that with the fact that edges can only be attached to every second disc in the sequence of the variable gadget we may need up to $6k$ discs for a variable node with k edges.

2.2.2 Edge gadget

The main part of the edge gadget is a sequence of discs $S_1 \dots S_k$ where S_1 is a disc in the variable gadget and S_k is a disc in the clause gadget.

Lets consider an edge gadget that goes from a variable gadget on the left to a clause gadget on the right. As with squares the sequence is split into three parts. $\mathcal{E}_1 = \{S_1 \dots S_l\}$



(a) no placement (b) true placement (c) false placement

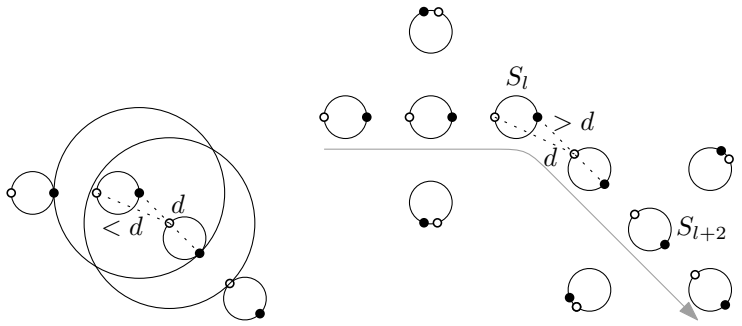
Figure 2.6: A variable gadget

is a horizontal alignment of discs that is starts in the variable gadget, false positions are the rightmost points of the discs. $\mathcal{E}_2 = \{S_{l+1} \dots S_m\}$ has its discs aligned along some slope and is used to bridge the difference in y-coordinate between S_1 and S_k. The false positions for discs in $\mathcal{E}_2$ are the points furthest along the slope towards S_m. $\mathcal{E}_3 = \{S_{m+1} \dots S_k\}$ is a horizontal alignment of discs connected to the clause gadget, where false positions are the rightmost points of the discs. In general the distance between two false positions of consecutive discs in any of the three parts is exactly d. The exceptions are described below.

The difficulty with the edge gadget for discs is in the bends between the different parts. This occurs between S_l and S_{l+1} and between S_m and S_{m+1}. We look at the bend between S_l and S_{l+1}. When placing S_{l+1} along the slope it is easy to place it such that only its false position s^f_{l+1} has distance d from the false position s^f_l of S_l as shown in Figure 2.7a. However in this placement the true positions s^t_{l+1} and s^t_l, which are at the opposite sides of the discs have a distance less then d from each other, as illustrated by the open points in Figure 2.7a. The disc along the slope can be moved further away such that $dist(s^t_l, s^t_{l+1}) = d$, but then other points in the disc along the slope also have distance d or more from the false position of the other disc. To avoid that we add two discs above and below S_{l-1} and S_{l+2} to force a point placed in that disc to either be on the true or false position, but not on any other position. This is illustrated in Figure 2.7b.

The same extra discs can also be used to stretch an edge so that it can have any length and not just multiples of d. By placing discs above and below two consecutive discs, S_i and S_{i+1}, in the sequence they can be moved some distance apart while still maintaining that if S_i has a false placement then also S_{i+1} must have a false placement.

In such a connection it is also easy to see there is a placement where all points are in true position. If all points along the sequence are placed in true position then by construction the points in a straight section have distance d between them and we also ensured that the true positions in a bend also have distance d between them, hence all points have distance d between them.



(a) True positions have distance less then d between them

(b) Adding discs forces points to be on true or false position

Figure 2.7: Problem and solution for making a bend in an edge gadget. Solid points are false positions whereas open ones are true positions.

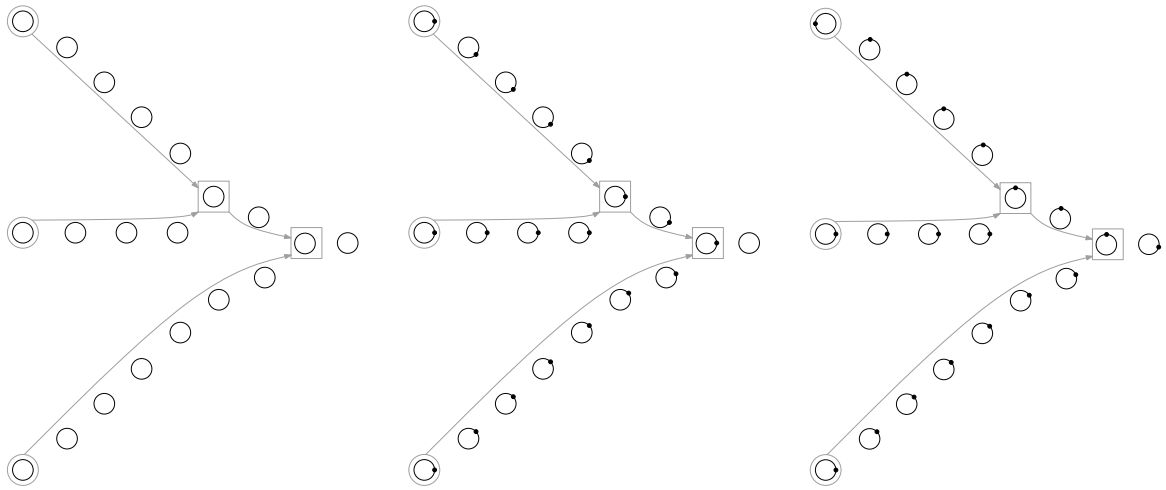
2.2.3 Clause gadget

The clause gadget is shown in Figure 2.8a and consists of a framework used to propagate true or false positions from the incoming discs, marked by grey circles, to merge discs which are marked by grey squares. The incoming discs are also part of edge gadgets. Note that false positions are rightmost points on the incoming discs, so towards the clause, and true position are leftmost points on the incoming discs, so away from the clause.

If the points on the incoming discs are in a false position then the point on the top merge disc can only be placed at its rightmost position. In turn there is only one position at distance d on the next disc along the grey arrows. Similarly the false position from the bottom incoming disc is propagated to the bottom merge square where the only point at distance d from the other points is the rightmost point of the disc. However if a point is placed on the rightmost point there is no placement possible for the disc to the right of the merge disc, so no placement with distance d between the points is possible when the points on the incoming discs are in false position. The placement for all discs of the clause gadget except the rightmost disc is shown in Figure 2.8b. For the rightmost disc no placement is possible with distance d to all other points.

If one of the points on the incoming discs is on its true position, say the point on the top incoming disc, then points can be placed differently. The points along the arrow from the top incoming disc to the top merge disc can be placed back along that arrow. Then the point on the top merge disc can be placed at the top most point. Then also the point on the disc between the two merge discs can be placed at its topmost point as well as the point on the bottom merge disc. Then also the point on the disc to the right of the bottom merge disc can be placed. The other points can be placed in the same way as when all incoming discs are in false position. Then all points are placed and have distance d between each other so a placement for the clause gadget is possible when the top incoming disc has a true placement. This placement is show in Figure 2.8c

When one of the other incoming discs has a true placement also the points can be moved back along the arrows such that there is eventually enough space for the point in the rightmost disc to be placed.



(a) No placement (b) False placement (c) True placement

Figure 2.8: The clause gadget for discs with two different placements.

2.2.4 Computing gadgets from a formula

The construction of the gadgets is basically the same as for squares. First the variables are constructed and positioned then the starting discs for the edges are computed. Then the edges are extended and clauses are added. An example of such a construction is shown in Figure 2.9, which is $\mathcal{R}(\phi)$. Note that the clause gadgets used here are smaller versions of the clause gadget that is described, in this example they work, but the smaller clauses are a problem when edges are close together. They are merely used to keep the example small.

Since we can construct $\mathcal{R}(\psi)$ for a formula ψ in polynomial time in the size of ψ and $\mathcal{R}(\psi)$ has a placement of points with distance d between points if and only if ψ is satisfiable we can conclude with the following theorem.

Theorem 3. Distant Representatives *is NP-hard for regions that are disjoint unit discs.*

From this and Theorem 1 we conclude the following corollary.

Corollary 2. Possible Closest Pair *is NP-hard for regions that are disjoint unit discs.*

2.3 Line segments

Also for disjoint horizontal line segments of unit length we can show NP-hardness of Distant Representatives using a reduction from Planar 3-SAT. Since the reduction is similar to that for squares and discs, only the gadgets will be given and $\mathcal{R}(\phi)$ as an example.

2.3.1 Variable gadget

The variable gadget is a vertically aligned sequence of line segments. The distance between the rightmost endpoint of one segment and the leftmost of the next is exactly d, so both possible placements have points alternating between the left and right endpoint along the sequence as illustrated in Figure 2.10.

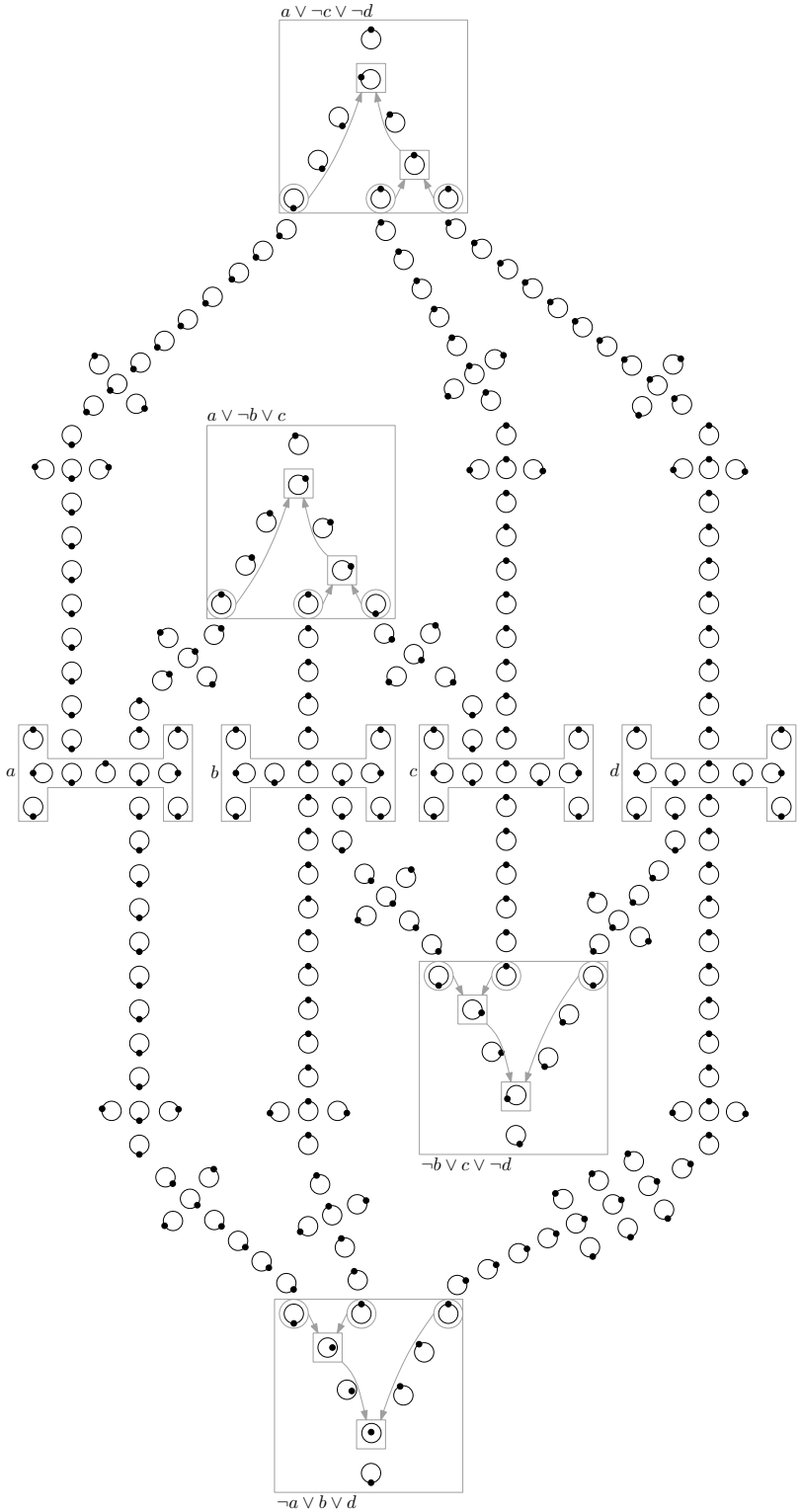


Figure 2.9: The set $\mathcal{R}(\phi)$ of discs with a placement according to the satisfying assignment in which all variables are true.

To ensure that the line segments in different edge gadgets are not to close together there should always be at least one line segment between two line segments of the variable gadget that are part of different edge gadgets. Since false placements are again the endpoints towards the clause and true placements away from the clause we may need up to $3k$ line segments for a variable node with k edges.

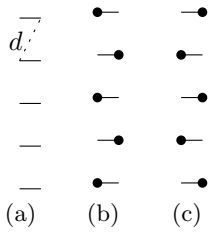


Figure 2.10: A variable gadget without placement (a), with a placement corresponding to true (b) and one corresponding to false (c)

2.3.2 Edge gadget

The edge gadget for line segments also consists of two horizontal parts and a section with a slope. For the horizontal part line segments are just aligned horizontally and one line segment can be lifted out of the sequence to shorten it in the same way as described for squares. For the sloped section line segments are aligned along the slope. The slope cannot be to steep to ensure that every next line segment is completely to the right of the previous one for an edge going to the right. Figure 2.11 shows an example of an edge gadget for line segments.

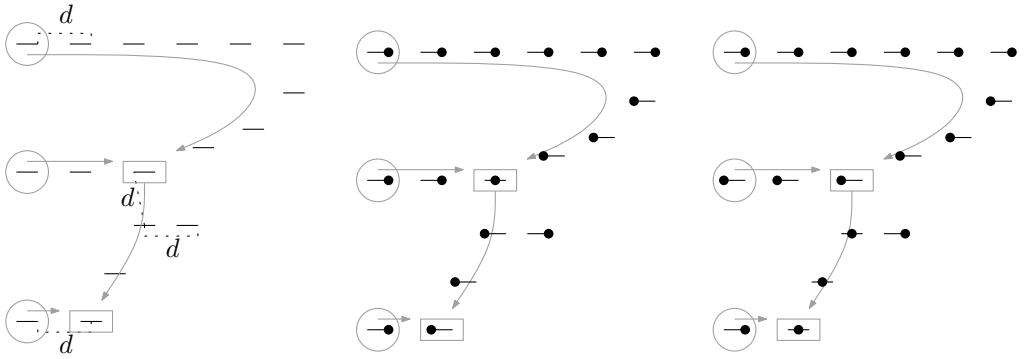
2.3.3 Clause gadget

The clause gadget for line segments is shown in Figure 2.12a. The grey circles indicate the *incoming line segments* which are also part of edge gadgets, and the grey rectangles indicate the *merge line segments* where the values of two incoming line segments are combined. The arrows indicate how placements are propagated. In case the incoming line segments have a false placement, so the points are placed at the right endpoint then for each line segment along the arrows there is only one point at distance d from the previous points along the arrow. This then leads to the placement show in Figure 2.12b where the distance between the lowest two point is less then d. So a placement with distance d between points is not possible when all points in the incoming line segments are in false position.

When one of the incoming line segments has a true placement, so a point placed on the left endpoint of the line segment. Then there is more room to place points further back along

Figure 2.11: Example of an edge gadget with line segments

the arrows. The case when the middle incoming line segment has a true placement is shown in Figure 2.12c.



(a) line segments of the clause gadget

(b) Placement for a clause that is false

(c) Placement for a clause that is true

Figure 2.12: The clause gadget and a false and true placement, grey elements are not part of the gadget.

2.3.4 Computing gadgets from a formula

For line segments this is done in the same way as for squares so we will only show $\mathcal{R}(\phi)$ here, which can be seen in Figure 2.13. Note that we cheated a little on the construction, since some edge do not have an edge between them in the variable gadget, but this is done simply to keep the example small.

Since we can construct $\mathcal{R}(\psi)$ for a formula ψ in polynomial time in the size of ψ and $\mathcal{R}(\psi)$ has a placement of points with distance d between points if and only if ψ is satisfiable we can conclude with the following theorem.

Theorem 4. Distant Representatives *is NP-hard for regions that are horizontal line segments of unit size.*

From this and Theorem 1 we conclude the following corollary.

Corollary 3. Possible Closest Pair *is NP-hard for regions that are horizontal line segments of unit size.*

2.4 Number of possible closest pairs

Here we consider the problem of counting the number of possible closest pairs, which we will name Count Closest Pairs.

Count Closest Pairs
Input: A set of regions $\mathcal{R}$.
Output: The number of possible closest pairs in $\mathcal{R}$.

Although it is NP-hard to determine if a certain pair of regions is a possible closest pair, this does not imply it is not possible to count the number of possible closest pairs. Showing that counting the number of possible closest pairs is NP-hard requires a separate proof.

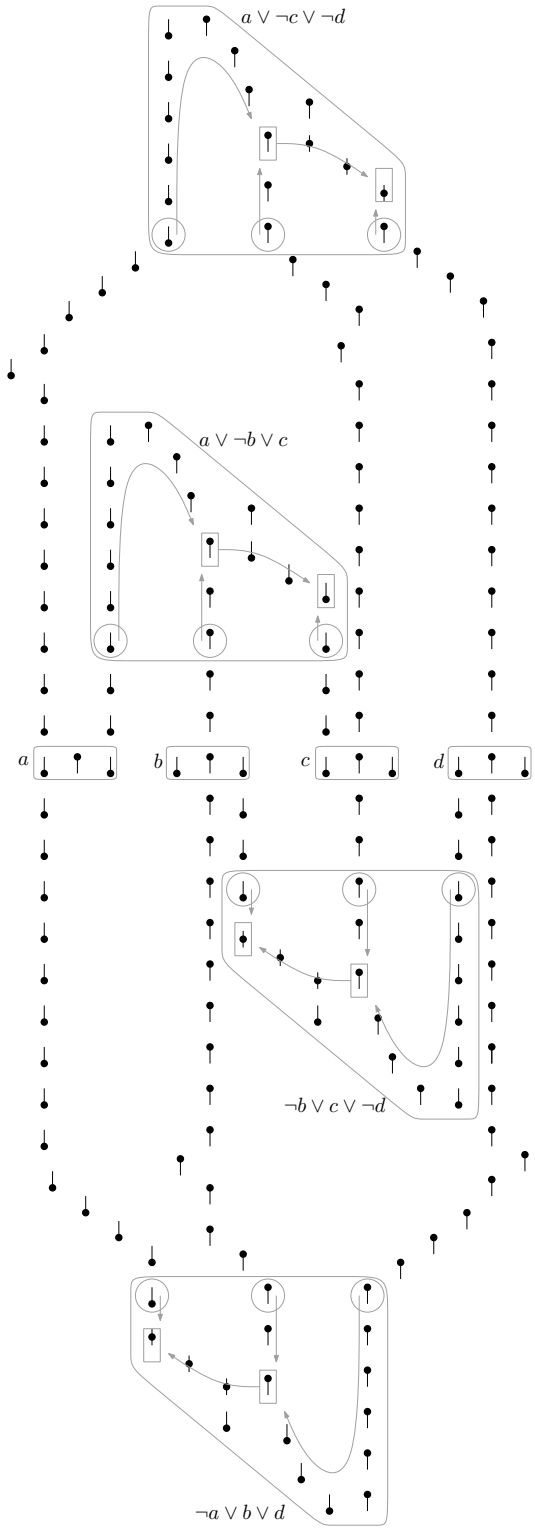


Figure 2.13: the set $\mathcal{R}(\phi)$ of line segments corresponding to the formula ϕ with a placement where all variables are true.

Theorem 5. Count Closest Pairs *for regions modeled as disjoint unit discs, disjoint axis aligned squares of unit size or disjoint horizontal line segments of unit length is NP-hard.*

Proof. We prove this using a reduction from Distant Representatives. Let $\mathcal{S}$ be the set of input regions for Distant Representatives and d the distance that should be between any two points.

Let R and S be two regions, such that $dist^+(R,S) = d$ and for any region $T \in \mathcal{S}$ we have $dist^-(T,R) > d$ and $dist^-(T,S) > d$. Let $\mathcal{S}_1 = \mathcal{S} \cup \{R,S\}$, then we count the number of possible closest pairs in $\mathcal{S}_1$. Let k be the number of possible closest pairs. Note that because $dist^+(R,S) = d$, for each placement the closest pair has at most distance d between them.

Next another two regions, P and Q, are constructed such that $dist^-(P,Q) = d$ and for every $T \in \mathcal{S}_1$ we have $dist^-(T,P) > d$ and $dist^-(T,Q) > d$. Let $\mathcal{S}_2 = \mathcal{S} \cup \{P,Q\}$, then we count the number of possible closest pairs in $\mathcal{S}_2$. Let $T,U \in \mathcal{S}_1$ be one of the k possible closest pairs in $\mathcal{S}_1$. Then there is a placement of points in $\mathcal{S}_1$ where (t,u) is a closest pair. Moreover because of R and S we know $dist(t,u) \leq d$. Since P and Q have minimum distance d to each other and to any other region, this implies that (t,u) is also a possible closest pair in $\mathcal{S}_2$. So all closest pairs for $\mathcal{S}_1$ are also possible for $\mathcal{S}_2$.

The regions that can form a possible closest pair in $\mathcal{S}_2$ that is not in $\mathcal{S}_1$ are P and Q. Since they have a minimal distance of more than d to any other region, there is no placement in which a point from P or Q forms a closest pair with a point from a region of $\mathcal{S}_1$. Hence Count Closest Pair will either return k or $k+1$ for $\mathcal{S}_2$, with k meaning (P,Q) is not a possible closest pair and $k+1$ that (P,Q) is a possible closest pair.

If (P,Q) is a possible closest pair then there is a placement for $\mathcal{S}_1$ and, hence, for $\mathcal{S}$, such that all points have distance d between them. So the answer to Distant Representatives on $\mathcal{S}$ would be YES.

If (P,Q) is not a possible closest pair then there is no placement for $\mathcal{S}_1$ such that all points have distance d between them. Because points in R and S can always be placed such that they have distance d from any other point and from each other this also implies there is no placement for $\mathcal{S}$ where all points have at least distance d to each other. So then the answer to Distant Representatives on $\mathcal{S}$ is NO.

If Count Closest Pairs could be solved in polynomial time then using this method also Distant Representatives could be solved in polynomial time. From Theorems 2, 3 and 4 we know that Distant Representatives in NP-hard for line segments, squares and discs. Hence Count Closest Pairs is NP-hard as well. □

2.4.1 Number of possible closest pairs for disjoint unit discs.

Theorem 5 shows that it is hard to count the exact number of possible closest pairs. In some cases, however is is possible to find good (subquadratic) bounds on the number of possible closest pairs. Next we show how to do this in the case of disjoint unit discs.

Let $\mathcal{S}$ be a set of disjoint unit discs. We define $d^- = \min(\{dist^-(P,Q) | P,Q \in \mathcal{S} \wedge P \neq Q\})$ and $d^+ = \min(\{dist^+(P,Q) | P,Q \in \mathcal{S} \wedge P \neq Q\})$. For disjoint unit disks d^- and d^+ are easy to determine, since they are both defined by the pair of discs $P,Q \in \mathcal{R}$ for which it holds that for all $R,S \in \mathcal{R}$ we have $dist(r_c,s_c) \geq dist(p_c,q_c)$, where r_c, s_c, p_c and q_c are the center points of R,S,P and Q respectively. This pair of discs can be found using a closest pair algorithm on the center points of the discs [CLRS01, Section 33.4].

Next note that only pairs that have distance at most $d^+ + 2$ between their centers can be a possible closest pair and that any pair has at least distance $d^- + 2$ between their centers.

Since d^+ and d^- come from the same pair it holds that $d^+ = d^- + 4$. This is used to prove Lemma 3.

Lemma 3. *Every disc P can form a possible closest pair with at most a constant number of other discs.*

Proof. For every disc P with center point p_c only discs whose center points have at least distance $d^- + 2$ to p_c and at most distance $d^- + 6$ to p_c can form a possible closest pair with P. If a disc Q with center point q_c would have $dist(p_c, q_c) < d^- + 2$ then $dist^-(P, Q) < d^-$ which is in contradiction to the definition of d^-. If Q is such that $dist(p_c, q_c) > d^- + 6$ then P and Q can never form a possible closest pair.

If we construct circles around each center point of a disc with radius $\frac{1}{2}d^- + 1$ then these discs do not intersect each other. If two such discs were to intersect each other the center points would be a distance smaller then $d^- + 2$ apart which is impossible. Since only discs with center points with distance smaller then $d^- + 4$ to p_c can form a possible closest pair with P all these circles must be contained within the disc with radius $1\frac{1}{2}d^- + 7$ around p_c as illustrated in Figure 2.14. It easily follows that only there can only be a constant number of these circles without intersecting each other. Hence, there are only a constant number of regions that can form a possible closest pair for each disc P. □

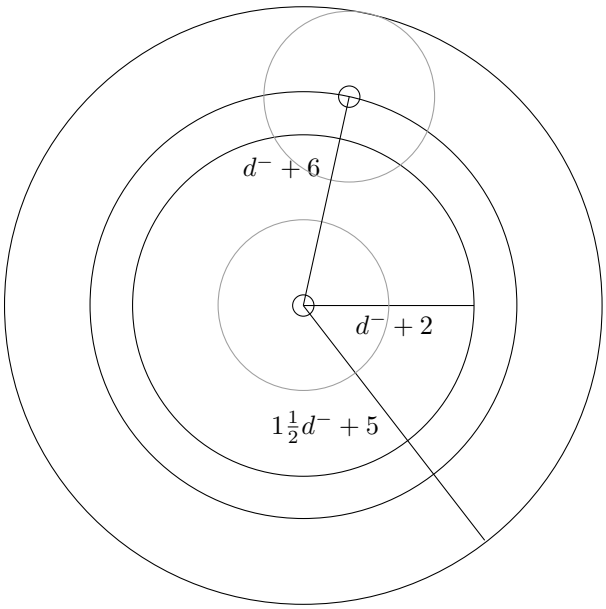


Figure 2.14: Only a constant number of circles with radius $\frac{1}{2}d^- + 1$ can fit in a disc with radius $1\frac{1}{2}d^- + 7$ around p_c.

From Lemma 3 we can conclude the following theorem.

Theorem 6. *Any set of disjoint unit discs has $O(n)$ possible closest pairs. Where n is the number of unit discs in the input.*

Finding pairs with a bound on minimal distance. We used the fact that all possible closest pairs have at most distance $d^+ + 2$ between their centers to show that there is only

a linear number of possible closest pairs. The same observation can be used to find a set of pairs of points that is at most linear in the size of the input set $\mathcal{R}$, but still contains all possible closest pairs. This algorithm finds all pairs of discs that have a distance less then $d^+ + 2$ between their center points.

To find these pairs of points we use a sweep line. The sweep line goes through the plane from left to right and stores the center points of regions of $\mathcal{R}$ that have an x-coordinate between the current position of the sweepline x_s and $x_s - (d^+ + 2)$. These points are stored in a data structure S which can report all points which have y-coordinate between a lower and upper bound in $O(\log k + m)$ time, where k is the number of elements stored and m the number of elements reported. Insertion and deletion on S can be done in $O(\log k)$ time. Whenever a the sweep line hits a new center point p a range query is used to find center points in S that may have distance less then $d^+ + 2$ from p. For each of the reported points, say a point q, the actual distance to p is checked and if $dist(p, q) \leq d^+ + 2$ it is reported. Then p is added to S and the sweep line can continue. Deleting points from S is done in a lazy way; a point is deleted if it is reported during a range query but its x-coordinate is already smaller then $x_s - d^+ + 2$. Because some points may be vertically aligned the sweep line just goes through the center points in lexicographical order. Algorithm 1 shows the pseudocode for this.

Algorithm 1: FindPairsWithMinimalDistance($\mathcal{R}$)

```
Q ← centerpoints of R in lexicographical order
while Q ≠ ∅ do
  p ← pop(Q)
  L ← RangeQuery(S, y_p − d⁺ − 2, y_p + d⁺ + 2)
  S ← S ∪ p
  for q ∈ L do
    if distance(p, q) ≤ d⁺ + 2 then
      (p, q) is reported
    else
      (p, q) is not possible closest pair
      if x_q < d⁺ + 2 then S ← S\{q}
```

Next, we show that Algorithm 1 indeed reports all pairs of points with distance at most $d^+ + 2$ between their center points.

Lemma 4. *Algorithm 1 reports exactly all pairs of regions with distance $d^+ + 2$ between their center points.*

Proof. The if-statement ensures that only pairs of center points are reported that have at most distance $d^+ + 2$ between them, so we only need to prove that all pairs of points with this property are reported.

Let p, q be two center points that have a distance of less then $d^+ + 2$ between them. Without loss of generality assume that q is hit by the sweep line first. Some time later p is hit by the sweep line, so $x_s = x_p$. Because $dist(p, q) \leq d^+ + 2$ and q was hit by the sweep line before p it holds that $x_s - d^+ + 2 \leq x_q \leq x_s$. This means q is still in S. Also from $dist(p, q) \leq d^+ + 2$ it follows that $y_p - (d^+ + 2) \leq y_q \leq y_p + d^+ + 2$, so the range query will report q and the pair (p, q) is reported by the algorithm. Since this holds for any two points

that have distance at most $d^+ + 2$ between their centers all these pairs of points are reported by the algorithm. □

Lemma 5. *Algorithm 1 runs in $O(n \log n)$ time where $n = |\mathcal{R}|$.*

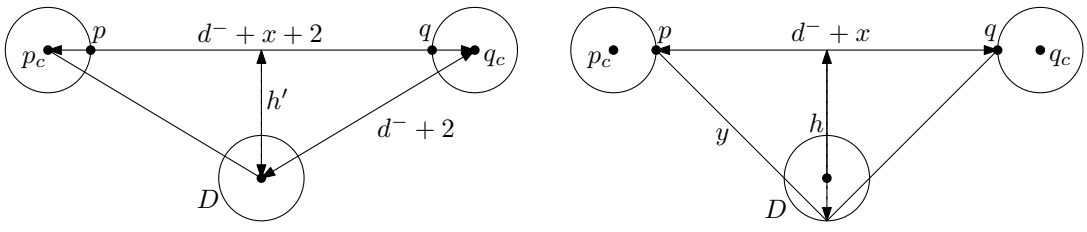
Proof. For every event point a range query is done. Because the range query only finds points in a $(2 * d^+ + 4) \times (d^+ + 2)$ and all points have at least distance $d^- + 2$ between them only a constant number of points is reported. Since there are at most n points in S the range query takes $O(\log n)$ time. The point p is then added to S which can also be done in $O(\log n)$ time. Lastly a check is done on each of the points reported by the range query, each operation in this check only takes $O(1)$ time and there is only a constant number of reported points so this takes a constant amount of time.

Since there are n events, handling all of them takes $O(n \log n)$ time. Sorting the event points also takes $O(n \log n)$ so the running time for Algorithm 1 is $O(n \log n)$. □

Finding possible closest pairs. For two discs with at least a distance $d^- + x + 2$ between their centers a simple placement can be used to show that this pair is indeed a possible closest pair. In this paragraph we will describe this simple placement and what x is.

Given two discs $P, Q \in \mathcal{R}$ with distance $d^- + x + 2$ between their centers p_c and q_c we can place a point $p \in P$ and $q \in Q$ such that $dist(p, q) = d^- + x$ as shown in Figure 2.15. Points in other discs are then placed as follows. Discs with center points below the line pq have their points placed as far down as possible and discs with center points above or on the line pq have their points placed as far up as possible.

By definition of d^- every two center points have at least distance $d^- + 2$ between them. Because we have placed all points below the line pq as low as possible the displacement from their center point is the same, so points below the line pq have at least distance $d^- + 2$ from each other. The same holds for points above or on the line pq (excluding p and q). A point above and a point below the line pq are displaced in opposite directions with respect to the center points of their discs so they also have at least distance $d^- + 2$ between them. So only the discs shown in Figure 2.15 can have a distance less then $d^- + 2$ between them. Of course not for each pair of discs there are discs above and below them as shown in Figure 2.15, however this is a worst case configuration. If a the disc D is placed somewhere else then y is always longer, because the distance between p and the center of D and q and the center of D has to be at least $d^- + 2$.



(a) h' depends on x (b) y depends on h

Figure 2.15: When using the placement described in this section y depends only on x

Using this placement, a maximal value for x can be determined, where p and q still form a closest pair. When we find this value for x we can use a plane sweep to find all pairs that

have less then $d^- + x + 2$ between their center points and report them as possible closest pairs.

Pythagoras gives:

$$h = 1 + \sqrt{(d^- + 2)^2 - (1/2(d^- + 2 + x))^2}$$

$$y = \sqrt{h^2 + (1/2(d^- + x))^2}$$

To ensure that (P, Q) indeed is a possible closest pair it must hold that $d^- + x \leq y$. For $d^- = 0$ this gives that any pair is closest pair in some placement when their minimal distance is smaller then $d^- + x$ for $x \leq 1.82$. For $d^- \geq 1$, x becomes greater then 2. However for a pair (P, Q) of regions for which $d^- + 2 < dist^-(P, Q) \leq d^- + x$ the placement described here may not have (p, q) as as closest pair. A pair (R, S) of regions that defines d^- has distance $d^- + 2$ between center points, so if r and s are on the same side of the line through p and q they are still distance $d^- + 2$ from each other. Therefor we only guarantee that (P, Q) is a possible closest pair if $dist^-(P, Q) \leq d^- + \min(x, 2)$.

To find the pairs or regions that have at most distance $d^- + x$ between them Algorithm 1 can be used, except there should be a test to see if $dist(p, q) < d^- + \min(x, 2)$.

Chapter 3

Exact Convex hull

The convex hull of a set S of points in $\mathbb{R}^2$ is defined as the intersection of all convex sets in which S is contained. For finite sets the convex hull is a polygon.

In the context of imprecise points we are interested in whether a given set of imprecise points can determine the vertices of the convex hull.

Exact Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: YES if there is a placement for $\mathcal{R}$ such that the points from regions in $\mathcal{C}$ are exactly the vertices of the convex hull of this placement, NO otherwise.

In this section we show that Exact Convex Hull is NP-hard when the regions are arbitrary line segments. The number of regions in $\mathcal{C}$ will be denoted by k during the rest of this section as well as in the sections on Superset Convex Hull (Chapter 4) and Subset Convex Hull (Chapter 5).

3.1 Arbitrary Line Segments

When the regions of $\mathcal{R}$ are arbitrary line segments which are allowed to have any size and orientation and to intersect each other, the exact convex hull problem is NP-hard. This can be proven using a reduction from 3-SAT, which is defined as follows.

3-SAT
Input: A 3-SAT formula ψ.
Output: YES if ψ is satisfiable, NO otherwise.

As with Planar 3-SAT which is described in Section 2.1, a 3-SAT formula consists of a disjunction of clauses where each clause is a conjunction of literals. Every literal is either a negated or non-negated occurrence of a boolean variable. In 3-SAT every clause consists of exactly three literals, but unlike with Planar 3-SAT we do not require the graph representation to be planar.

For a given formula ψ we will show how to construct a set of line segments $\mathcal{R}(\psi)$ with a subset $\mathcal{C}(\psi) \subseteq \mathcal{R}(\psi)$ such that Exact Convex Hull on these sets returns YES if ψ is satisfiable and NO otherwise. For the reduction one basic construct is used to represent a literal. Each of these constructs is placed inside a circle arc such that it is on the boundary of the convex hull, these circle arcs are shown in Figure 3.1a as grey areas. The number of circle

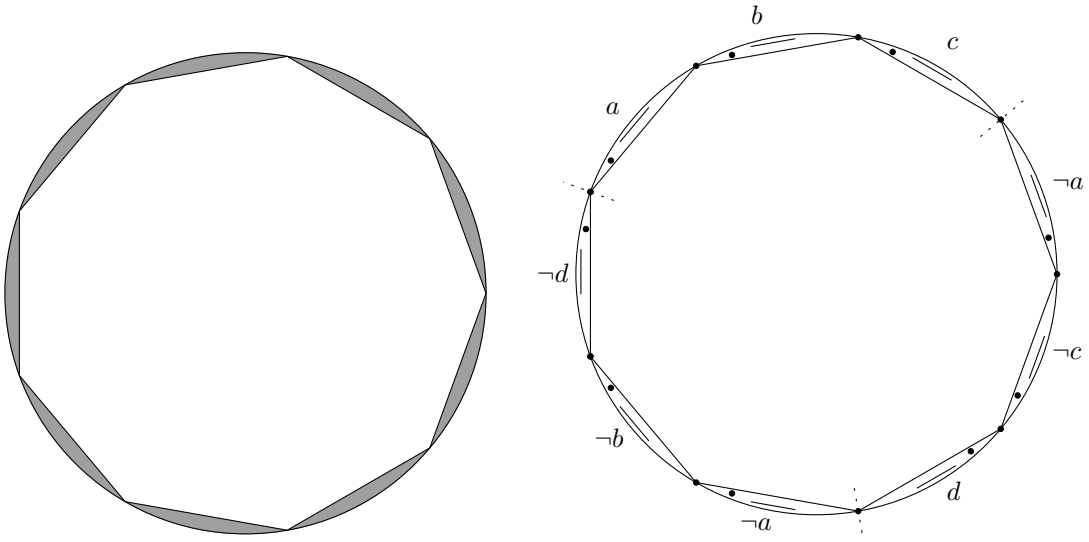
arcs is exactly the number of literals in the 3-SAT-formula. We place the literal constructs of one clause next to each other on the circle. The order in which these groups of literals belonging to one clause appear on the circle does not matter, neither does the order of the literals within a clause.

A literal construct for a non-negated literal is shown in Figure 3.2a. Note that this can easily be scaled to fit into a circle arc. For each literal in the 3-SAT formula such a construct is created and placed in one of the circle arcs. The leftmost and rightmost point, P and Q, are degenerate line segments (points) and part of $\mathcal{C}(\psi)$. Because they are on the boundary of the circle they are always guaranteed to be vertices of the convex hull. The point labeled with L is a degenerate line segment and is not in $\mathcal{C}(\psi)$ this point should not be a vertex on the convex hull. The line segment A is in $\mathcal{C}(\psi)$. The endpoints, a_t and a_f, denote true and false values for the variable. For a negated variable the point L is located under the a_f endpoint as shown in Figure 3.2b. The point L for a non-negated literal is placed such that a placement near the endpoint a_t *protects* L from becoming a vertex on the convex hull, whereas a placement near the a_f endpoint does not. For a negated literal a placement near the a_f endpoint protects L and a placement near the a_t endpoint does not.

For the rest of this section we will use the following formula ϕ as a running example.

$$\phi = (a \vee b \vee c) \wedge (\neg a \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg d)$$

The literal constructs of ϕ are shown in Figure 3.1b

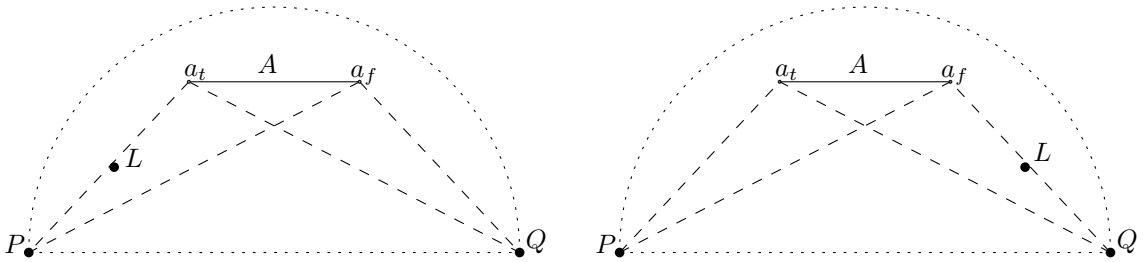


(a) Placement of literal constructs

(b) literal constructs of the 3-SAT formula ϕ

Figure 3.1

During the description of the literal construct we say that an endpoint of a line segment is *outside* the convex hull if it is outside the convex hull of the placement that is described in the context or it is a vertex of that convex hull. If an endpoint of a line segment is outside the convex hull then also a point can be placed on this line segment such that it is a vertex of the convex hull. An endpoint is said to be *inside* the convex hull if it is inside the convex hull of the placement that is described and not a vertex of this convex hull. For a line segment that

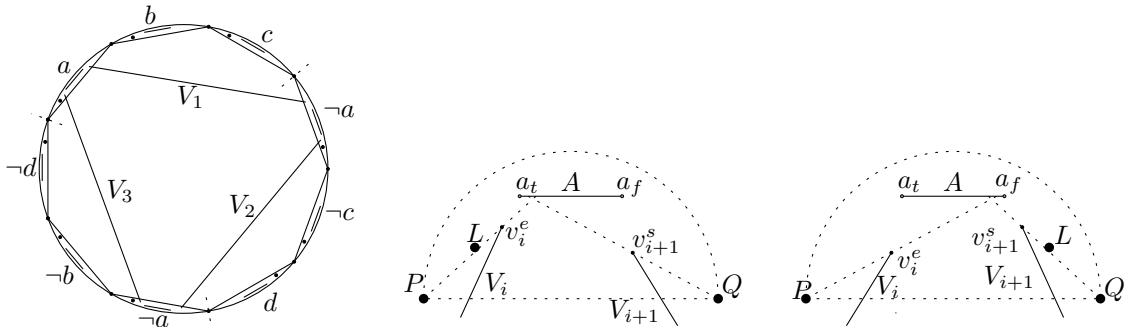


(a) non-negated literal (b) negated literal

Figure 3.2: literal constructs for reducing SAT to convex hull test

has both end points inside the convex hull no placement exists that makes the point from that line segment a vertex on the convex hull.

The line segment A in different literal constructs of the same variable should have a placement reflecting the same value. So either all points should be placed near the true endpoint or all points should be placed near the false endpoint. To ensure this we place line segments in a cyclic order between all literal constructs of the same variable as shown for the variable a of our example formule ϕ in Figure 3.3a. Let $V_1 \dots V_k$ denote the line segments of a variable and v_i^s and v_i^e the two endpoints of the line segment V_i. Then the endpoints of these these line segment in a literal construct are shown in Figures 3.3b and 3.3c. The lines Pv_i^e and Qv_{i+1}^s divide the line segment A into three parts. If a point a is placed in the leftmost part of A then the endpoint v_i^e of C will be below the lines Pa and aQ, so v_i^e is inside the convex hull. However v_{i+1}^s is above the line aQ, so it is outside the convex hull. If a is placed on the rightmost part of A, then v_{i+1}^s is inside the convex hull and v_i^e is outside. Lastly if a is placed in the middle part of A then both v_i^e and v_{i+1}^s are inside the convex hull. With this construction we can prove the following lemma.



(a) linesegments between literals of the variable a in ϕ (b) non-negated literal (c) negated literal

Figure 3.3

Lemma 6. *For literals of the same variable either the points on the line segments labeled with A are all placed near the true endpoint (a_t) or all near the false endpoint (a_f) in any placement such that the points of the line segments connecting literals of the same variable are vertices of the convex hull.*

Proof. Let there be k literals for a variable. Then there are k line segments connecting the literal constructs and in each literal construct at least one endpoint of these line segments is kept from being outside the convex hull. It is then easy to see that for no literal a point can be placed in the middle part of A since then there has to be a line segment for which both endpoints are inside the convex hull. A point on this line segment can never be a vertex of the convex hull. If one some literal construct the point on A is placed on the a_f endpoint and for the next construct in the cycle a points is placed near the a_t endpoint, then the line segment between these two constructs has both endpoints inside the convex hull. Therefore either all points of literal construct of the same variable are placed near the a_t endpoint or all points are placed near the a_f endpoint. □

In a satisfying assignment for a 3-SAT formula only one literal in each clause has to be true. In terms of our literal constructs that means that if for one of the literals in a clause the point L is protected from being a vertex on the convex hull by the placement of a point on A then the other two can also be protected. We do this using two line segments that have endpoints just above the point labeled with L in each line segment. For our example ϕ this is shown in Figure 3.5, which is shows all the line segments of $\mathcal{R}(\phi)$. We denote these line segments by W_1 and W_2. W_1 and W_2 are part of $\mathcal{C}(\psi)$ and should be outside the convex hull. As can be seen in Figure 3.4 these points can always be placed outside the convex hull since the endpoints are above the line Pa_t for a non-negated literal or Pa_f for a negated literal. With these line segments we can prove the following lemma.

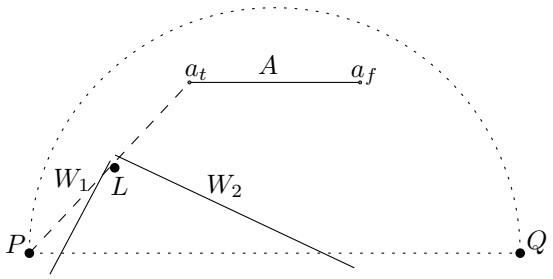


Figure 3.4: Literal construct with lines W_1 and W_1 which extend to other literal constructs of the same clause.

Lemma 7. *For every clause if either of the three points labeled with L is protected from being a vertex on the convex hull by the point $a \in A$ then then we can place points on W_1 and W_2 such that the other point L on the other two literals is also inside the convex hull.*

Proof. It is easy to see that in whichever literal the point L is protected by a a point on W_1 and a point on W_2 can be placed above L in the other two literals. □

Now that we have shown how to construct $\mathcal{R}(\psi)$ and $\mathcal{C}(\psi)$ and explained the basic properties of the line segments connecting literals of the same clause and of the same variable we can prove the following.

Lemma 8. *A SAT-formula ψ is satisfiable if and only if there is a placement of points for $\mathcal{R}(\psi)$ such that the points from regions in $\mathcal{C}(\psi)$ are exactly the vertices of the convex hull.*

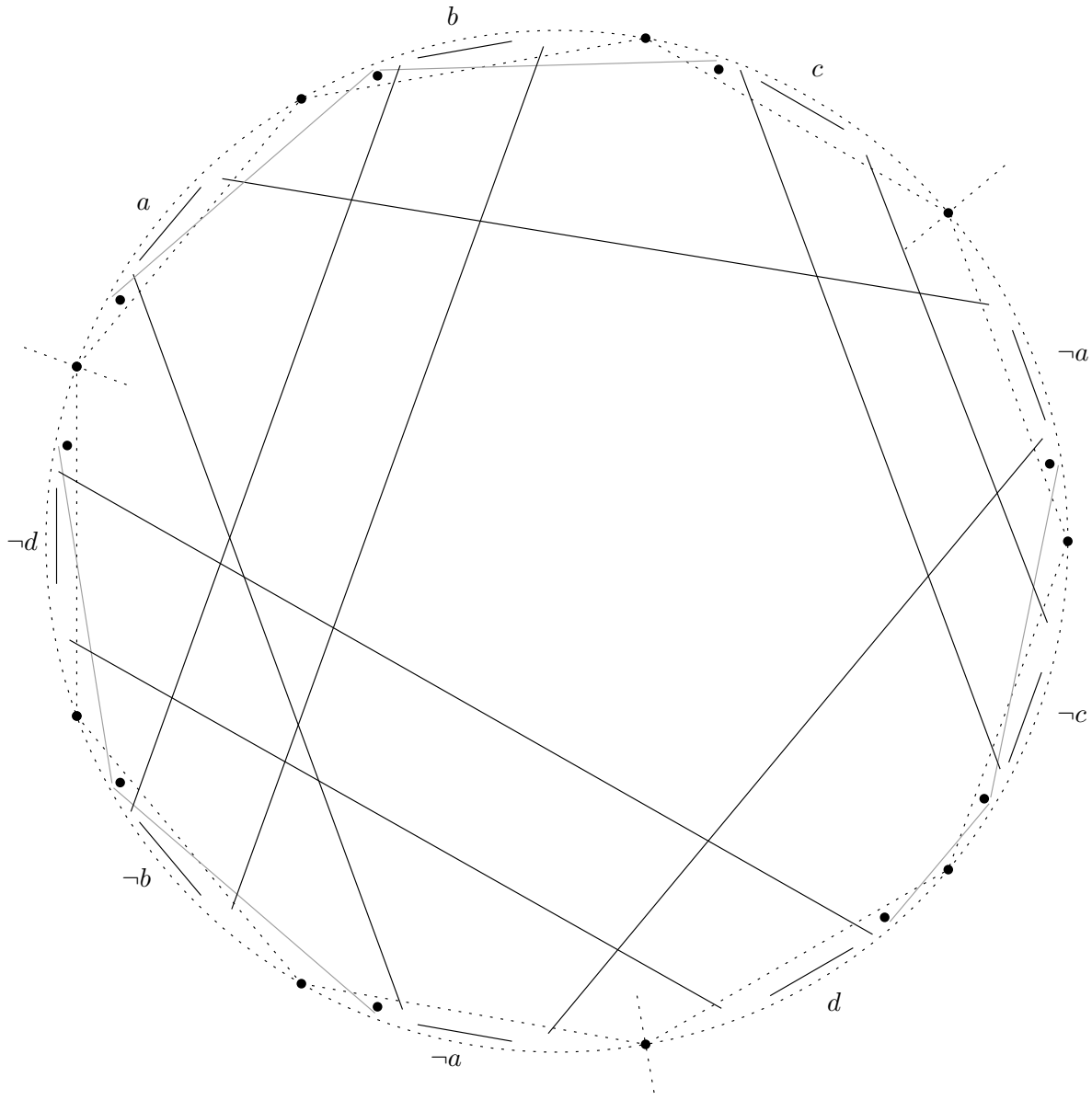


Figure 3.5: The set of line segments $\mathcal{R}(\phi)$, dotted lines are not part of the set and grey line segments connect literals of the same clause.

Proof. If ψ is satisfiable then there is a satisfying assignment S. For each literal the point on the line segment A can be placed either on a_t or a_f depending on the value of that variable in S. From Lemma 6 it follows that there is a placement possible for the points on the line segments connecting literals of the same variable such that each of them occurs on the convex hull. Because V is a satisfying assignment there is one literal for every clause that is true. For $\mathcal{R}(\psi)$ that means that for every clause there is one literal where L is inside the convex hull due to the placement of a point on A. It then follows from Lemma 7 that points on the clause connections can also be placed such that they are on the convex hull and points that should be inside the convex hull are inside.

If ψ is not satisfiable then for every possible assignment there is at least one clause for which all literals are false. In terms of the line segments that means that for every placement of points on the line segments labeled with A there is at least one clause for which in all literal constructs L is not inside the convex hull by the placement of the point on A. Then by Lemma 7 there is no placement possible such that all points from regions not in $\mathcal{C}(\psi)$ are not a vertex on the convex hull and points from regions in $\mathcal{C}(\psi)$ are vertices of the convex hull. □

Computing $\mathcal{R}(\psi)$ and $\mathcal{C}(\psi)$ can be done in polynomial time, because each literal construct has at most four connections each of which can easily be computed by giving all literals an identifier and connecting variables and clauses in the order induced by the identifier. From this we can then conclude the following theorem.

Theorem 7. *Given a set $\mathcal{R}$ of arbitrary line segments and a subset $\mathcal{C} \subseteq \mathcal{R}$,* Exact Convex Hull *is NP-hard on these sets.*

Chapter 4

Superset Convex hull

Instead of determining if there is a placement where the points from regions in $\mathcal{C}$ are exactly the vertices of the convex hull, we can also try to determine if there is a placement for which the points from $\mathcal{C}$ are a superset of the vertices of the convex hull.

Superset Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: YES if there is a placement of points such that the points from the regions in $\mathcal{C}$ are a superset of the vertices of the convex hull, NO otherwise.

In this section we will show that Superset Convex Hull is NP-hard for arbitrary line segments, but can be solved in $O(k^2 n \log n)$ time for disjoint vertical line segments of unit size.

4.1 Arbitrary line segments

Proving that Superset Convex Hull is NP-hard for arbitrary line segments will be done using a reduction from 3-SAT. The proof is very similar to that of Section 3.1. There we defined a literal construct which is connected to other literal constructs of the same clause and variable. For Superset Convex Hull we do not require every point from a region in $\mathcal{C}$ to be on the convex hull. So if we would use the same literal constructs, then the points on the line segments that connect literals of the same variable do not have to be vertices of the convex hull. Hence, Lemma 6 does not hold. However, now we do not require the points on line segments connecting literals of the same clause to be vertices of the convex hull. Therefore we use one construct for a variable and all literals are represented as points $L_1 \dots L_l$ in that construct as shown in Figure 4.1. As before a point placed at a_t corresponds to the variable being true and a point placed at a_f corresponds to the variable being false.

$L_1 \dots L_m$ are the non-negated literals and $L_{m+1} \dots L_l$ are the negated literals. They are placed on convex curves as indicated by the dashed lines in Figure 4.1. A point $a \in A$ is never above both curves, so it can never occur that a literal point from $L_1 \dots L_m$ is below the line pa and a literal point from $L_{m+1} \dots L_l$ is below the line aq.

Literals that are part of the same clause are again connected with line segments in the same way as in the reduction from 3-SAT to Exact Convex Hull; every clause is connected by two line segments which have endpoints just above the literal points of the clause. For the example formula ϕ the resulting set of line segments is shown in Figure 4.2. Let L_i represent

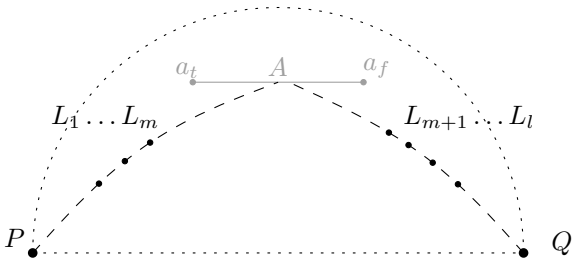


Figure 4.1: Variable gadgets for reducing 3-SAT to Superset Convex Hull on line segments

a non-negated literal which is connected to another literal of the same clause by the line segment V with endpoint v just above L_i. Then the position of v is such that L_i is under the polyline avP for any point $a \in A$. Hence, regardless of where a is, L_i will not become a vertex of the convex hull if a point on V is placed at v. The point v is also close enough to L_i such that it does not protect any other literal point from being on the convex hull when a is placed to the right of the arc containing L_i. This implies that the only way to keep a point L_i from becoming a vertex of the convex hull is by placing a point above it on a line segment that connects it to other literals of the same clause or by placing the point a far enough to the a_t endpoint. For a point representing a negated literal the same holds.

Now we can show that the set of regions $\mathcal{R}(\psi)$ derived from a 3-SAT formula ψ allows a placement such that only points from regions in $\mathcal{C}(\psi)$ are vertices of the convex hull if and only if the corresponding 3-SAT formula is satisfiable.

Lemma 9. *A 3-SAT formula ψ is satisfiable if and only if there is a placement for $\mathcal{R}(\psi)$ such that only points from regions in $\mathcal{C}(\psi)$ are vertices on the convex hull of this placement.*

Proof. If ψ is satisfiable then let V_s be a satisfying assignment. For each variable A a point is placed at a_t if the variable is true in V_s and at a_f if the variable is false. Because this is a satisfying assignment in every clause at least one literal will be inside the convex hull due to a placement of the variable points. The other literals in the clause can then be kept inside the convex hull by placing the points on the line segments of that clause just above the literal points.

If ψ is not satisfiable then for any valuation V there is a clause c in which all literals are false. If the points are placed on the variable line segments at the a_t or a_f endpoint according to V then all literals of c will still be vertices of the convex hull. Each clause line segment can only keep one literal of c from becoming a vertex of the convex hull, but there are only two line segments and three points. At least one literal will still be a vertex of the convex hull. Placing points on different places on the variable line segments then at a_t or a_f does not help either since it is never possible to keep both a negated and non-negated literal from being a vertex of the convex hull only by the point on a variable line segment. Hence no placement will have only points from $\mathcal{C}$ as vertices of the convex hull if ψ is not satisfiable. □

Since it is also easy to see that we can construct $\mathcal{R}(\psi)$ and $\mathcal{C}(\psi)$ in polynomial time we can conclude with the following theorem.

Theorem 8. *Given a set $\mathcal{R}$ of arbitrary line segments and a subset $\mathcal{C} \subseteq \mathcal{R}$, then* Superset Convex Hull *is NP-hard for these sets.*

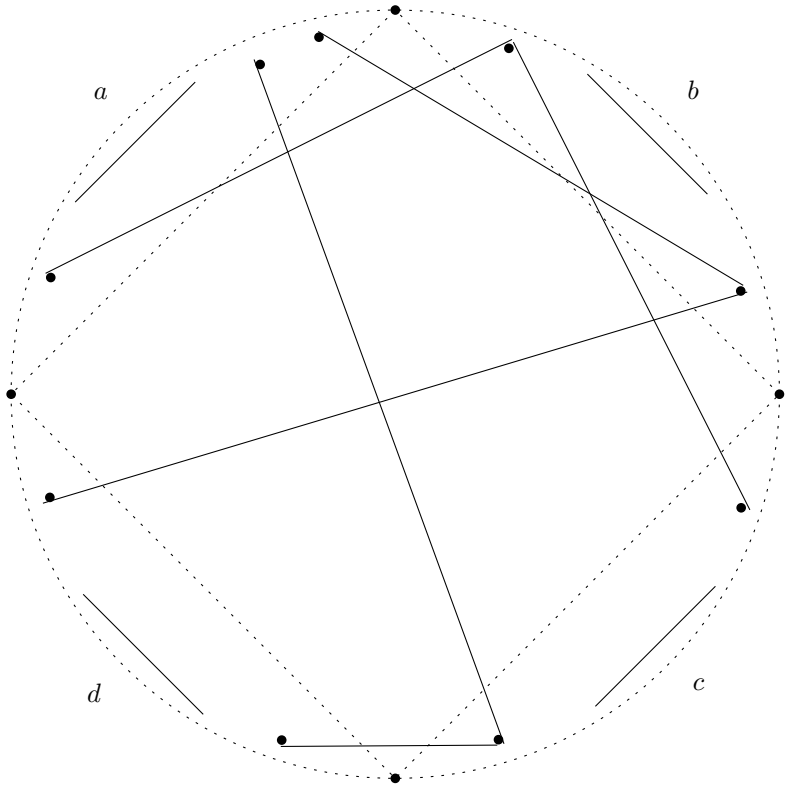


Figure 4.2: Set of line segments $\mathcal{R}(\phi)$ corresponding to the example formula ϕ

4.2 Disjoint parallel line segments of unit length

For disjoint parallel line segments of unit size we give an algorithm that solves Superset Convex Hull. We first reduce it to a different problem called Maximal Convex Hull. If we view possible convex hulls as sets of points then they form a partial order under inclusion. This partial order has some maximal elements. A *maximal convex hull* CH_{max} of a given set $\mathcal{S}$ of regions is a possible convex hull of $\mathcal{S}$ with the property that there is no other possible convex hull CH of $\mathcal{S}$ such that $CH \supset CH_{max}$. Maximal Convex Hull is then defined as follows.

Maximal Convex Hull
Input: A set $\mathcal{R}$ of regions and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: YES if there is a placement of points in $\mathcal{C}$ such that the resulting convex hull CH_{max} is a maximal convex hull of $\mathcal{C}$ and every region of $\mathcal{R}\backslash\mathcal{C}$ has a point inside CH_{max}.

Next we show that Maximal Convex Hull and Superset Convex Hull are equivalent for convex regions.

Lemma 10. *Let $\mathcal{R}$ be a set of disjoint convex regions and $\mathcal{C} \subseteq \mathcal{R}$. Then* Maximal Convex Hull *will return YES if and only if* Superset Convex Hull *would return YES.*

Proof. If Maximal Convex Hull returns YES with $\mathcal{R}$ and $\mathcal{C}$ as input, then there exists a placement for $\mathcal{C}$ with a maximal convex hull CH_{max} such that each region of $\mathcal{R}$ has a

point inside CH_{max}. Then by placing all points of regions of $\mathcal{R}$ inside CH_{max}, it becomes a valid convex hull for a placement of points for $\mathcal{R}$ where only points of $\mathcal{C}$ are vertices. Hence, Superset Convex Hull also returns YES.

If Superset Convex Hull returns YES with $\mathcal{R}$ and $\mathcal{C}$ as input, then there exists a placement V for $\mathcal{R}$ such that the convex hull CH has only points of $\mathcal{C}$ as vertices. CH is not necessarily a maximal convex hull for the regions of $\mathcal{C}$. By definition there exists a maximal convex hull CH_{max} for regions of $\mathcal{C}$ that is a superset of CH. CH_{max} then still has only points from regions in $\mathcal{C}$ as vertices and all points of V are inside CH_{max}. So CH_{max} is a solution to Maximal Convex Hull. □

From now on we assume, without loss of generality, that the input line segments are vertical. In a maximal convex hull on vertical line segments all vertices except the leftmost and rightmost vertex have to be an endpoint of a line segment. If that is not the case then the convex hull can easily be made larger by simply moving the point up or down towards an endpoint. The leftmost and rightmost vertex are located somewhere on the leftmost and rightmost line segment. For now we will assume there is one unique leftmost line segment S_l and one unique rightmost line segment S_r. At the end of this section we argue that Maximal Convex Hull can still be solved if the leftmost or rightmost line segment is not unique.

Let $\mathcal{C}'$ denote the set $\mathcal{C}\backslash\{S_l, S_r\}$. We can look for a solution to the maximal convex hull problem by first looking at different placements for $\mathcal{C}'$. As said we only need to consider placements where points are placed at the endpoints of line segments. Also we are only interested in maximal convex hulls. Solving Maximal Convex Hull is then done as described in Algorithm 2. FindPlacement is an algorithm that returns YES if there is a placement for S_l and S_r given a convex hull on $\mathcal{C}'$, such that all regions of $\mathcal{R}$ intersect the resulting convex hull, and NO otherwise. How FindPlacement works will be explained later. For now we assume that FindPlacement(CH_{max}, S_l, S_r) returns the correct result.

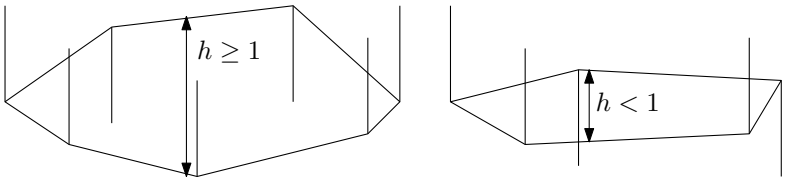
Algorithm 2: FindMaxHull$(\mathcal{R}, \mathcal{C})$

```
C' ← C\{S_l, S_r}
\\ Only consider placements at endpoints of line segments in C'
forall Maximal convex hulls CH_max on C' do
  if FindPlacement(CH_max, S_l, S_r) = YES then
    return YES
return NO
```

First we show that Algorithm 2 returns YES if and only if there is a maximal convex hull such that all regions of $\mathcal{R}$ intersect it by showing that every maximal convex hull on $\mathcal{C}$ consists of a maximal convex hull on $\mathcal{C}'$ and points on S_l and S_r.

Lemma 11. *Every maximal convex hull on $\mathcal{C}$ is the convex hull of a point $s_l \in S_l$, a point $s_r \in S_r$ and a maximal convex hull on $\mathcal{C}'$ where points are only placed at endpoints of regions in $\mathcal{C}'$*

Proof. Let CH_{max} be a maximal convex hull on $\mathcal{C}$. Then there is a placement V which has CH_{max} as its convex hull and only contains endpoints for the line segments in $\mathcal{C}'$. Let s_l be the placement of a point on S_l in V, and let s_r be the placement of a point on S_r in V. Let V' be the placement of points for $\mathcal{C}'$ in V. The convex hull $CH(V')$ of V' may not be maximal for $\mathcal{C}'$, but by definition there is also a maximal convex hull $CH_{C'}$ which contains $CH(V')$.



(a) Fat convex hull (b) Thin convex hull

Figure 4.3: Fat and thin convex hulls, the height is denoted by h

Then $CH_{C'}$ must be contained in CH_{max}, otherwise CH_{max} would not be a maximal convex hull. Then because $CH_{C'}$ contains $CH_{V'}$ the convex hull of $CH_{C'}$, s_l and s_r is CH_{max}. □

The running time of FindMaxHull now depends on three things; the number of maximal convex hulls on $\mathcal{C}'$, the time it takes to compute all maximal convex hulls on $\mathcal{C}'$, and the running time of FindPlacement.

First we look at the total number of maximal convex hulls on $\mathcal{C}'$ which only have points placed at the endpoints of regions in $\mathcal{C}'$. We look at two different types of convex hulls which can be distinguished by their height. We define the height of a convex hull as the maximum length of a vertical line segment between the upper and lower boundary of the convex hull as illustrated in Figure 4.3. A *fat convex hull* is a maximal convex hull which has a height of at least one, whereas a *thin convex hull* has a height of less then 1. Because we are using unit size line segments this means that in a fat convex hull at least one line segment is completely contained inside the convex hull and in a thin convex hull no line segment is contained in it, see Figure 4.3.

Before computing bounds on the number of fat and thin convex hulls we first define functions l_t and l_b. Let $S_1 \dots S_m$ be the line segments of $\mathcal{C}'$ then $l_t(S_i, S_j)$ is the line between the upper endpoint of S_i and the upper endpoint of S_j. Similarly $l_b(S_i, S_j)$ is the line segment between the two lower endpoints. Now we can determine how many fat and thin convex hulls there can be on $\mathcal{C}'$.

Lemma 12. *There are $O(k^2)$ fat convex hulls on $\mathcal{C}'$ when placing points only at the endpoints of regions in $\mathcal{C}'$. Moreover, this bound is tight in the worst case*

Proof. In a fat convex hull there is at least one line segment which is completely inside the convex hull. Because it is a maximal convex hull all line segments that are completely contained in the convex hull are consecutive. Let $S_1 \dots S_m$ be the line segments of $\mathcal{C}'$ from left to right. Suppose the line segments $S_i \dots S_j$ are the only line segments that are completely inside the convex hull. Then there is a unique placement for $S_{i+1} \dots S_{j-1}$ because each of these line segments can have only one endpoint outside the convex hull of the endpoints of S_i and S_j as shown in Figure 4.4.

For the segments $S_1 \dots S_{i-1}$ we first make the following observation. Let S_a and S_{a+1} be two consecutive line segments in $S_1 \dots S_{i-1}$. Then either the line $l_t(S_a, S_i)$ is above S_{a+1} or the line $l_b(S_a, S_i)$ is below the S_{a+1}. Because we know S_i is completely inside the convex hull either placing a point at the upper endpoint of S_a or at the lower endpoint will cause S_{a+1} to be completely inside the convex hull. For example if the line between the upper endpoints of S_a and S_i is above S_{a+1} then if we place s_a at the upper endpoint of S_a then we must place s_{a+1} at the lower endpoint of S_{a+1} to get a maximal convex hull. In that case

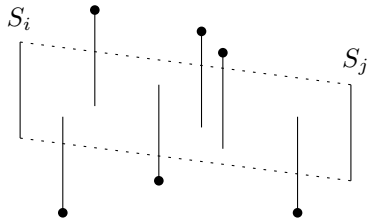


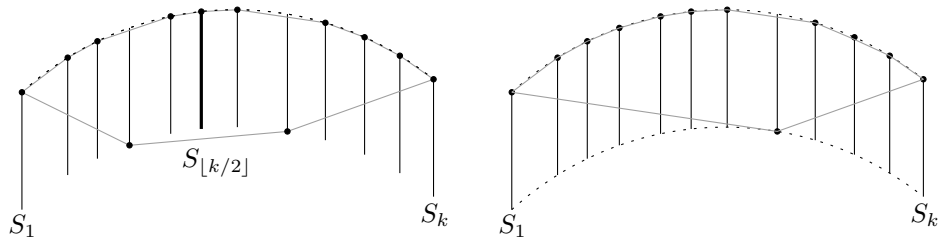
Figure 4.4: Unique placement of middle part of a fat convex hull

S_{a+1} is completely inside the convex hull contradicting our assumption that S_i is the first line segment completely inside the convex hull. Therefore the point on S_1 can only be placed at one endpoint. This same arguments holds for $S_2 \dots S_{i-2}$, but for S_{i-1} it does not hold since there is no line segment between S_{i-1} and S_i. For this line segment a point can be placed at either endpoint. So for the left part, $S_1 \dots S_{i-1}$, there are at most two placements possible. Similarly there are only two placements for the right part, $S_{j+1} \dots S_m$. This means that there are at most four placements that lead to a fat convex hull where S_i is the first line segment inside the convex hull and S_j is the last. Since there are $O(k^2)$ different ways to choose i and j there are also $O(k^2)$ fat convex hulls.

The set of line segments in Figure 4.5a shows that there can be $\Omega(k^2)$ fat convex hulls. In this set of line segments the upper endpoints of all line segments form a convex chain. In a fat convex hull at least one endpoint of a line segment is inside the convex hull. It can also be on the boundary as long as it is not a vertex. If we look at the middle line segment $S_{\lfloor k/2 \rfloor}$, then the lower endpoint of this line segment is inside the convex hull if for a line segment in $S_1 \dots S_{\lfloor k/2 \rfloor - 1}$ a point s placed on the lower endpoint and for a line segment in $S_{\lfloor k/2 \rfloor + 1} \dots S_k$ a point is placed on the lower endpoint. There are $\Omega(k^2)$ choices for these two line segments and each of them gives a different fat convex hull. Therefore there are $\Omega(k^2)$ fat convex hulls. □

Lemma 13. *There are $O(k)$ thin convex hulls on $\mathcal{C}'$ when placing points only at the endpoints of regions in $\mathcal{C}'$ and this bound is tight in the worst case.*

Proof. In a thin convex hull no line segment is completely inside the convex hull. Let $S_1 \dots S_m$ be the line segments of $\mathcal{C}'$ then for any line segment S_i, where $3 \leq i \leq m$, it holds that either $l_t(S_1, S_i)$ is above S_2 or $l_b(S_1, S_i)$ is below S_2. If $l_t(S_1, S_i)$ is above S_2 then we cannot place s_1 and s_i at the upper endpoints of S_1 and S_i, since then S_2 would be completely inside the convex hull. Because we are only looking for thin convex hull this placement is not allowed. Similarly, if $l_b(S_1, S_i)$ is below S_2 then we cannot place s_1 and s_i at their lower endpoints. This means that if we place s_1 at the upper endpoint of S_1 then for some regions $\mathcal{S}_t \subseteq \{S_3 \dots S_m\}$ we cannot place a point at the upper endpoint whereas if we place s_1 at the lower endpoint we cannot place a point at the lower endpoint of the other regions of $S_3 \dots S_m$. We denote this second set of line segments by $\mathcal{S}_b \subseteq \{S_3 \dots S_m\}$. If we place s_1 at the upper endpoint of S_1 then for the line segments $\mathcal{S}_t \cup \{S_2\}$ we are still free to place points and either endpoint. However the convex hull of $\mathcal{S}_t \cup \{S_2\}$ should also be thin since it is a subset of the convex hull on $\mathcal{C}'$. Now let x denote the number of regions in $\mathcal{S}_t \cup \{S_2\}$ then the number of regions in $\mathcal{S}_b \cup \{S_2\}$ is $m - x$. If we denote the number of possible placements for m line segments by $T(m)$ then $T(m)$ is defined by the following recurrence.



(a) a fat convex hull (b) a thin convex hull

Figure 4.5: Examples to show bounds on number fat and thin convex hulls are tight

$$T(m) = \begin{cases} T(m-x) + T(x) & \text{if } m > 0 \\ 2 & \text{if } m = 1 \end{cases}$$

Solving this recurrence we get that $T(k) = 2k$. Hence, there are $O(k)$ thin convex hulls.

The same set of line segments that we used to show that there are $\Omega(k^2)$ fat convex hulls can also be used to show there are $\Omega(k)$ thin convex hulls. If we place the point on one line segment down and the rest up then it is easy to see that no line segment has both endpoints inside the convex hull as illustrated in Figure 4.5b. Because there are k line segments there are $\Omega(k)$ different thin convex hulls. □

The next problem is that of computing these maximal convex hulls on $\mathcal{C}'$. For both the fat and thin convex hulls an algorithm to compute them follows easily from the proof of Lemmas 12 and 13. Therefore we only describe the algorithms shortly while showing the running time.

Lemma 14. *We can compute all maximal convex hulls on the set $\mathcal{C}'$ of line segments where points are only placed at the endpoints of the line segments in $\mathcal{C}'$ in $O(k^3)$ time.*

Proof. For the fat convex hulls we simply try any two line segments S_i and S_j and then compute the left middle and right part. Since for every region the correct placement of a point only depends on a constant number of line segments this takes $O(1)$ time for each point. The convex hull of this placement can then be computed in $O(k)$ time since the line segments can be sorted at the start of the algorithm. There are $O(k^2)$ choices for S_i and S_j, so all fat convex hulls can be computed in $O(k^3)$ time.

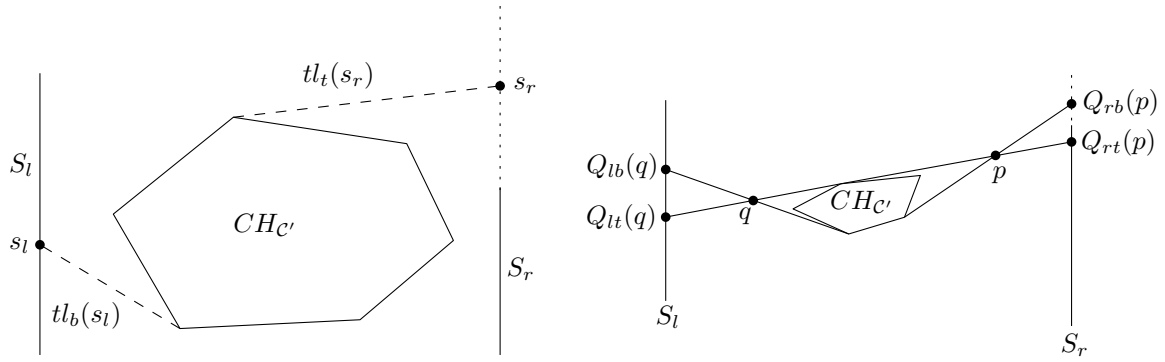
For the thin convex hulls we can simply place a point on the upper endpoint of the first line segment S_1 and then recurse on $\mathcal{S}_t \cup \{S_2\}$. We then place a point on the lower endpoint of S_1 and recurse on $\mathcal{S}_b \cup \{S_2\}$. This means that on a set of k line segments we need $O(k)$ time to determine for all line segment if we can place a point already or if it should be placed in the recursion. Also when all points are placed the convex hull has to be computed which takes $O(k)$ time. Let $S(k)$ denote the time required to find all thin convex hulls for k line segments. Then $S(k)$ is defined by the following recurrence.

$$S(k) = \begin{cases} O(k) + S(k-x) + S(x) & \text{if } k > 0 \\ O(k) & \text{if } k = 1 \end{cases}$$

This recurrence solves to $O(k^2)$, so we can find all thin convex hull on a set of k line segments in $O(k^2)$ time. □

Now we discuss how FindPlacement works and what the running time is. Let $CH_{\mathcal{C}}$ denote a maximal convex hull on $\mathcal{C}$ which is the convex hull of $CH_{\mathcal{C}'}$, a point $s_l \in S_l$ and $s_r \in S_r$. Let s be a point that is either on the vertical line through S_l or the vertical line through S_r. Then we define $tl'_t(s)$ as the line that is tangent to $CH_{\mathcal{C}'}$ such that $CH_{\mathcal{C}'}$ is below the line, and that passes through the point s. Then $tl_t(s)$ is the line segment of $tl'_t(s)$ from s to the nearest point on $CH_{\mathcal{C}'}$. Similarly $tl'_b(s)$ is the tangent line that contains s and is below $CH_{\mathcal{C}'}$. Again, $tl_b(s)$ is the line segment from s to the nearest point on $CH_{\mathcal{C}'}$. These tangent lines are illustrated in Figure 4.6a.

We have to find a placement for S_l and S_r such that all upper endpoints of line segments in $\mathcal{R}\backslash\mathcal{C}$ are above or on the lower boundary of $CH_{\mathcal{C}}$ and all lower endpoints of $\mathcal{R}\backslash\mathcal{C}$ are below or on the upper boundary. Let R'_t be the set of upper endpoints of regions in $\mathcal{R}\backslash\mathcal{C}$ and R'_b the set of lower endpoints. Then not all points of R'_t and R'_b are interesting. Points of R'_t that are in $CH_{\mathcal{C}'}$ or straight above are guaranteed to be above the lower boundary regardless of the placement for S_l and S_r. Therefore we define R_t to be the subset of R'_t which contains only points that are not in $CH_{\mathcal{C}'}$ or straight above it. R_b is defined similarly. Lastly we define functions Q_{lt}, Q_{lb}, Q_{rt} and Q_{rb}. Let $p_t \in R_t$, then $Q_{lt}(p_t)$ is a point on the vertical line through S_l such that $p_b \in tl_t(Q_{lt}(p_b))$. In other words $Q_{lt}(p_b)$ is the point such that the tangent line to the upper chain through this point passes through p_b. This means that placing a point s_l on S_l lower than $Q_{lt}(p_b)$ will put p_b above $tl_t(s_l)$ whereas placing s_l higher will put p_l below $tl_t(Q_{lt})$. Note that $Q_{lt}(p_b)$ is not defined if p_b is to the right of $CH_{\mathcal{C}'}$, but in that case we can simply define $Q_{ly}(p_b)$ as the point with y-coordinate ∞. The other three functions are defined in a similar manner as illustrated in Figure 4.6b.



(a) tl_t and tl_b (b) Q_{lt}, Q_{lb}, Q_{rt} and Q_{rb}

Figure 4.6: Illustration of several functions

A maximal convex hull $CH_{\mathcal{C}'}$ on $\mathcal{C}'$ is always between S_l and S_r. However we can distinguish two cases which are dealt with separately by FindPlacement. Let s_{lt} and s_{rt} denote the upper end points of S_l and S_r respectively and s_{lb} and s_{rb} denote the lower endpoints.

In the first case $CH_{\mathcal{C}'}$ has some part outside the quadrangle $s_{lt}s_{rt}s_{rb}s_{lb}$ or on the upper or lower boundary of the quadrangle. For now we assume that the top part is outside this polygon. In this case we know the upper boundary of the convex hull on $\mathcal{C}$ will consist of $tl_t(s_l)$, a part of the upper boundary of $CH_{\mathcal{C}'}$ and $tl_t(s_r)$ as shown in Figure 4.7a. Hence, lower endpoints that should be below the upper boundary of the $CH_{\mathcal{C}}$ can be divided into two sets, R_{bl} and R_{br}. All points of R_{bl} can only be below the upper boundary by placing s_l high enough and the points in R_{br} can only be below the upper boundary by placing s_r high

enough. We simply compute what the lowest possible placement is for s_l and s_r such that all lower endpoints are below upper boundary of $CH_{\mathcal{C}}$. Then we test if all points of R_t are above the lower boundary of $CH_{\mathcal{C}}$. If this is not the case, then no solution is possible with this $CH_{\mathcal{C}'}$. Algorithm 3 describes this is in a little more detail.

Algorithm 3: FindPlacementCase1(CH_{max}, S_l, S_r)

```
q ← a topmost point of CH_C'
R_ll ← points of R_b right of q
R_lr ← points of R_b left of q
s_l ← lower endpoint of S_l
forall p ∈ R_bl do
  if Q_lt(p) is below s_l then s_l ← Q_lt(p)
s_r ← lower endpoint of S_r
forall p ∈ R_br do
  if Q_rt(p) is below s_r then s_r ← Q_rt(p)
if s_r is above S_r or s_l is above S_l then
  return NO
else
  lb ← lower boundary of convex hull of CH_C', s_r and s_l
  forall p ∈ R_t do
    if p is under lb then return NO
return YES
```

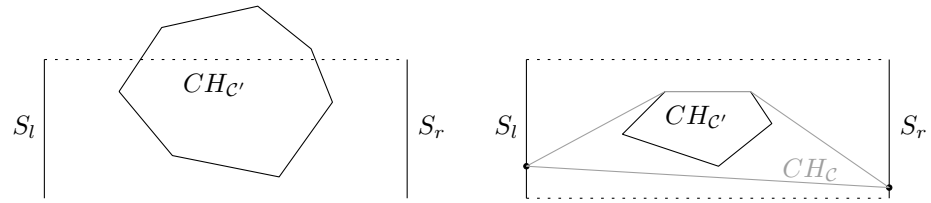
Now we prove that FindPlacementCase1 returns the correct answer and that it runs in $O(n \log k)$ time.

Lemma 15. FindPlacementCase1 *returns YES if and only if there is a placement for S_l and S_r such that $CH_{\mathcal{C}}$ has a non-empty intersection with each region of $\mathcal{R}$.*

Proof. If FindPlacementCase1 finds a solution then s_l is below or on $Q_{lt}(p_{bl})$ for all points $p_{bl} \in R_{bl}$, and s_r is below or on $Q_{rt}(p_{br})$ for all points $p_{br} \in R_{br}$. If $R_b \supset R_{bl} \cup R_{br}$ then the algorithm returns NO, since a point that is between the two points will never be below the upper boundary of the convex hull. Hence, every point of R_b is either in R_{bl} or R_{br}. This implies that all points of R_b are below the upper boundary of the convex hull. The algorithm also tests if all points of R_t are below the convex hull so the solution that is found is correct. If FindPlacementCase1 does not find a solution then either the placement found for s_l or s_r was above S_l or S_r in which case there is no placement on S_l or S_r such that all points of R_{br} and R_{bl} are below the upper boundary of the convex hull. The other case when the algorithm returns NO is when the placement that is found based on R_{br} and R_{bl} still has one or more points from R_t below the lower boundary of the convex hull. This can only be solved by moving either s_l or s_r down. In either case a point from R_b will go above the convex hull, hence there is no correct placement. □

Lemma 16. FindPlacementCase1 *runs in $O(n \log k)$ time.*

Proof. For a point $p \in R_t \cup R_b$ the point $Q_{lt}(p)$ can be computed in $O(\log k)$ time using a binary search on the points of the upper boundary of $CH_{\mathcal{C}'}$ to find the tangent point. Since there are $O(n)$ points in R_b finding a placement for s_l and s_r can be done in $O(n \log k)$ time.



(a) $CH_{\mathcal{C}'}$ has a part above or on the line between upper endpoints of S_l and S_r.

(b) $CH_{\mathcal{C}'}$ is completely between S_l and S_r. The lower boundary of $CH_{\mathcal{C}}$ is a direct connection and upper boundary an indirect connection.

Figure 4.7: Different cases for FindPlacement

The lower boundary lb of $CH_{\mathcal{C}}$ then consists of $O(k)$ line segments, so for each point in R_t it takes $O(\log k)$ time to determine if it is above or below lb. Hence FindPlacementCase1 takes $O(n \log k)$ time. □

If a lower part of $CH_{\mathcal{C}'}$ is below the quadrangle $s_{lt}s_{rt}s_{rb}s_{lb}$ then a similar algorithm can be used. If $CH_{\mathcal{C}'}$ sticks out at the top as well as the bottom then either of the two algorithms can be used. For the rest of this section we only consider the top part of $CH_{\mathcal{C}'}$ to be above the line $s_{lt}s_{rt}$.

In the other case $CH_{\mathcal{C}'}$ is inside the quadrangle $s_{lt}s_{rt}s_{rb}s_{lb}$. The upper boundary of $CH_{\mathcal{C}}$ can be a straight line between s_l and s_r that does not intersect the interior of $CH_{\mathcal{C}'}$ in which case we say the upper boundary is a *direct connection* between s_l and s_r. The upper boundary of $CH_{\mathcal{C}}$ can also be a polyline consisting of $tl_t(s_l)$, a polyline along the upper boundary of $CH_{\mathcal{C}'}$ and $tl_t(s_r)$ as illustrated in Figure 4.7b. In this case we say that the upper boundary is an *indirect connection* between s_l and s_r.

Observe that for any choice of points on S_l and S_r the convex hull $CH_{\mathcal{C}}$ can not have a direct connection as the upper and lower boundary. Either both the upper and lower boundary are indirect connections or one of the two is a direct connection. We can then determine if there is a suitable placement for s_l and s_r by assuming one of these three choices for the upper and lower boundary and trying to find a placement for s_l and s_r accordingly.

Upper and lower boundary are both indirect connections. We can solve this case by assuming that for any point $s_l \in S_l$ we have that s_l is connected to the lower and upper boundary of $CH_{\mathcal{C}'}$ with $tl_t(s_l)$ and $tl_b(s_l)$. We make the same assumption for any point $s_r \in S_r$. Each point $p_t \in R_t$ should be above the lower boundary of $CH_{\mathcal{C}}$. This is the case when either s_l is placed below or on $Q_{lb}(p_t)$ or s_r is placed below or on $Q_{rb}(p_t)$. Symmetrically each point $p_b \in R_b$ is below the upper boundary of $CH_{\mathcal{C}}$ if s_l is placed above or on $Q_{lt}(p_b)$ or s_r is placed above or on $Q_{rt}(p_b)$. We can then find a suitable placement by first placing s_l at the upper endpoint of S_l and s_r at the lower endpoint of S_r. If there is still a point $p_b \in R_b$ such that neither s_l is above or on $Q_{rt}(p_b)$ nor s_r is above or on $Q_{lt}(p_b)$ then either of the two will have to move up. However because s_l is already at its highest point, s_r is moved up to $Q_{rt}(p_b)$. Because of this it is possible that another point $p_t \in R_t$ now has $Q_{lb}(p_t)$ above s_l and $Q_{rb}(p_t)$ above s_r. Then either s_r or s_l has to be moved down. However, s_r cannot be moved down so s_l should be moved down. This process then continues until either a suitable placement is found, s_l reaches the lower endpoint of S_l, or s_r reaches the lower endpoint of S_r.

To do this more efficiently we define lists $T_{S_l}, T_{S_r}, B_{S_l}$ and B_{S_r} as follows. The lists T_{S_l} and T_{S_r} both contain all points of R_t. In T_{S_l} the points are ordered such that for two points $p_1, p_2 \in R_t$ it holds that p_1 occurs before p_2 in T_{S_l} if and only if $Q_{lb}(p_1)$ is above $Q_{lb}(p_2)$. When moving s_l down along S_l the points of R_t get inside the convex hull in the order of T_{S_l}. In T_{S_r} the points are sorted such that they go outside the convex hull in the order they are listed when moving s_r up. The lists B_{S_l} and B_{S_r} both contain all points of R_b and are ordered similarly. The list B_{S_l} has the points sorted to correspond to moving s_l up and B_{S_r} has the points sorted to correspond with moving s_r down. Using these lists FindPlacement deals with this case as described in Algorithm 4.

Algorithm 4: FindPlacementCase21(CH_{max}, S_l, S_r)

```
Compute T_{S_l}[1...l], T_{S_r}[1...l], B_{S_l}[1...m], B_{S_r}[1...m]
i_1 ← 1; i_2 ← 1; i_3 ← 1; i_4 ← 1
s_l ← upper endpoint of S_l
s_r ← lower endpoint of S_r
R'_t ← points of R_t below tl_b(s_l) or tl_b(s_r)
R'_b ← points of R_b above tl_t(s_l) or tl_t(s_r)
while R'_t ≠ ∅ or R'_b ≠ ∅ do
  while R'_t ≠ ∅ do
    i_1 ← i_1 + 1
    R'_t ← R'_t \ {T_{S_l}[i_1]}
  s_l ← Q_lb(T_{S_l}[i_1])
  while Q_lt(B_{S_l}[i_2]) is above s_l do
    R'_b ← R'_l ∪ {B_{S_l}[i_2]}
    i_2 ← i_2 + 1
  while R'_b ≠ ∅ do
    Symmetric to when R'_t is not empty
if s_l ∈ S_l and s_r ∈ S_r then return YES else return NO
```

Next, we prove that this algorithm finds a placement for S_l and S_r such that all line segments of $\mathcal{R}$ have some part inside $CH_\mathcal{C}$ if there is a such a placement for which $CH_\mathcal{C}$ has two indirect connections. Also we show that FindPlacementCase1 never returns an placement such that not every region of $\mathcal{R}$ has a point inside $CH_\mathcal{C}$.

Lemma 17. *If there is a placement for S_l and S_r such that $CH_\mathcal{C}$ has a non-empty intersection with every region of $\mathcal{R}$ and the upper and lower boundary of $CH_\mathcal{C}$ are both indirect connections then* FindPlacementCase21 *returns YES.*

Proof. This is equivalent to proving that if the algorithm reports no solution then there is no solution with two indirect connections. The algorithm keeps invariant that any placement of s_l higher then its current position or s_r lower then its current position is not a solution. At the start of the algorithm this is trivially true. Then every time we move s_l there is a region $P \in \mathcal{R} \backslash \mathcal{C}$ completely below $tl_b(s_l)$ or $tl_b(s_l)$. From our invariant we know that any placement of s_r lower then its current position does not lead to a solution hence we can only move s_l down. Then s_l is moved down only as far as needed to get a part of P inside the convex hull. It then follows that any placement of s_l higher then its current position does not lead to a solution. Similarly moving s_r keeps the invariant valid.

It then follows that if s_l is below the lower endpoint of S_l then no placement of s_l on S_l leads to a solution. Hence, there is no solution. The same holds when s_r is above the upper endpoint of S_r □

Lemma 18. *If* FindPlacementCase21 *returns YES then there is a placement of s_l and s_r such that $CH_{\mathcal{C}}$ has a non-empty intersection with each region of $\mathcal{R}$.*

Proof. The algorithm returns YES only if it finds a placement of s_l and s_r such that R'_t and R'_b are empty and the points s_l and s_r are on S_l and S_r. This means that for the final placement of s_l and s_r all lower endpoints of regions in R_b are below either $tl_t(s_l)$ or $tl_t(s_r)$ and all up endpoints in R_t are above $tl_b(s_l)$ or $tl_b(s_r)$. If the placement does have two indirect connections as we assumed then obviously all regions of $\mathcal{R}$ have a non-empty intersection with $CH_{\mathcal{C}}$. It is also possible that $CH_{\mathcal{C}}$ has one direct connection, but then still the line segments $tl_t(s_l), tl_t(s_r), tl_b(s_l)$ and $tl_b(s_r)$ are inside the convex hull. Then the points of R_t must still be above the lower boundary of the convex hull and the points from R_b must still be below the upper boundary. Hence, also in this case the resulting convex hull has a non-empty intersection with each region of $\mathcal{R}$. □

Lastly we prove that FindPlacementCase21 runs in $O(n \log n)$ time.

Lemma 19. FindPlacementCase21 *runs in $O(n \log n)$ time.*

Proof. First we compute the lists $T_{S_l}, T_{S_r}, B_{S_l}$ and B_{S_r}. If we precompute all the values of Q_{lt}, Q_{lb}, Q_{rt} and Q_{rb} for points in R_t and R_b then we can sort them in $O(n \log n)$ time. Then determining if a point from R_t is below the convex hull or if a point from R_b is above the convex hull can be done in $O(1)$ time using the lists so computing R'_t and R'_b takes $O(n)$ time. In each of the while loops at least one of the indices of the lists is increased, so we can make $O(n)$ passes through the loops. The other operations in the loops each take just constant time. The running time of FindPlacementCase21 is $O(n \log n)$. □

Either upper of lower boundary is a direct connection. Since the approach is symmetric when the lower or upper boundary is a direct connection we only describe it for the upper boundary being a direct connection.

First we assume that the upper boundary is a direct connection, so the upper boundary will be a straight line $s_l s_r$ which is above $CH_{\mathcal{C}'}$. We should ensure that all points of R_b are below $s_l s_r$ and all points of R_t are above the lower boundary of $CH_{\mathcal{C}}$. To ensure all points of R_t are above the lower boundary we would like to place s_l and s_r as low as possible. For this reason we will only look at placements where the line $s_l s_r$ is tangent to CH_t, which is the convex hull of R_b and $CH_{\mathcal{C}'}$. Then placing either s_l or s_r lower will cause the line $s_l s_r$ to intersect the interior of $CH_{\mathcal{C}'}$ or have a point of R_b above it.

We can compute for every placement of s_l the corresponding placement of s_r. We will denote the placement of s_r that corresponds to s_l as $Y_l(s_l)$ and the placement of s_l that corresponds to s_r as $Y_r(s_r)$. Note that $Y_l(s_l)$ may also be a point on the vertical line through S_r, since the tangent line to CH_t may go below S_r. Now for R_t we can make lists T_{S_l} and T_{S_r} as before and start by placing s_l at the upper endpoint and compute where s_r is. Then if there is still a point $p_t \in R_t$ below the lower boundary of $CH_{\mathcal{C}}$ we move s_l down until p_t is inside the lower boundary, then we can compute the new placement for s_r and go through T_{S_r} to find see if there are any new points of R_t that are below the lower boundary.

We continue this until either no points of R_t are still below the lower boundary or until either s_l reaches the lower endpoint of S_l or s_r reaches the upper endpoint of S_r. In the first case a solution for Maximal Convex Hull is found. In the second case no solution is possible where the top boundary is a direct connection. This approach is described in Algorithm 5.

Algorithm 5: FindPlacementCase22(CH_{max}, S_l, S_r)

Compute $T_{S_l}[1 \dots l]$ and $T_{S_r}[1 \dots l]$
$i_1 \leftarrow 1; i_2 \leftarrow 1$
$CH_t \leftarrow$ convex hull of $CH_{\mathcal{C}'}$ and R_l
$S_l \leftarrow$ top endpoint of S_l
if $Y_l(s_l)$ is above S_r **then return** NO
if $Y_l(s_l)$ is below S_r **then**
  $s_r \leftarrow$ lower endpoint of S_r
  $s_l \leftarrow T_r(s_r)$
$R'_t \leftarrow$ points in R_t that are below convex hull of $CH_{\mathcal{C}'}$, s_l and s_r
while $R'_t \neq \emptyset$ **do**
  $i_1 \leftarrow i_1 + 1$
  $R'_t \leftarrow R'_t \backslash T_{S_l}[i_1]$
  $S_l \leftarrow Q_{lb}(T_{S_l}[i_1])$
  $S_r \leftarrow Y_l(s_l)$
  while $Q_{rb}(T_{S_r}[i_2])$ is below S_r **do**
    $R'_t \leftarrow R'_t \cup \{T_{S_r}[i_2]\}$
    $i_2 \leftarrow i_2 + 1$
if $s_l \in S_l$ and $s_r \in S_r$ **then return** YES **else return** NO

For FindPlacementCase22 we also have to prove that it finds a solution if there is one with a direct connection and never returns an invalid solution.

Lemma 20. FindPlacementCase22 *returns YES if and only if there is a placement for s_l and s_r such that $CH_{\mathcal{C}}$ has a non-empty intersection with each region of $\mathcal{R}$.*

Proof. If FindPlacementCase22 returns YES then a placement was found such that no lower endpoints of regions in $\mathcal{R}\backslash\mathcal{C}$ are above the upper boundary of $CH_{\mathcal{C}}$ and no upper endpoints are below the lower boundary. This implies that for this placement $CH_{\mathcal{C}}$ has a non-empty intersection with every region of $\mathcal{R}$.

Suppose there is a placement of s_l and s_r such that the line $s_l s_r$ is not tangent to CH_t but above it, such that $CH_{\mathcal{C}}$ has a non-empty intersection with every region of $\mathcal{R}$. Then we can move s_l or s_r down to make the lower boundary lower. Doing this will not place any points of R_t that are above the lower boundary of the $CH_{\mathcal{C}}$ below it. This way s_l and s_r can be moved down until $s_l s_r$ is tangent to CH_t. Therefore we only have to consider placements such that $s_l s_r$ is tangent to CH_t.

Similarly as with FindPlacement21 we keep invariant that any placement of s_r below its current position cannot be a placement where the upper boundary is a direct connection and $CH_{\mathcal{C}}$ has a non-empty intersection with all regions of $\mathcal{R}$. Every placement of s_l above its current position does not lead to a solution either. It is easy to see that this holds for the initial placement of s_l and s_r. Then for every point $p_t \in R'_t$ we can only move s_l down

to get it inside the convex hull. Then s_r must be moved accordingly to keep $s_l s_r$ tangent to CH_t. Because placing s_l between its old and new position would put p_t below the convex hull the invariant again holds. Hence, if FindPlacementCase21 returns NO, then s_l was placed below S_l or s_r was placed above S_r, in either case the invariant gives us that there is no correct placement for which the upper boundary of $CH_{\mathcal{C}}$ is a direct connection. □

The running time of FindPlacementCase22 is as follows.

Lemma 21. FindPlacementCase22 *runs in* $O(n \log n)$ *time.*

Proof. FindPlacementCase22 computes the convex hull of $CH_{\mathcal{C}'}$ and R_b which takes $O(n \log n)$ time. The lists can again be computed in $O(n \log n)$ time and by the same reasoning as for FindPlacementCase21 we only go through the loops $O(n)$ times. However in this case not all operations can be done in constant time. Computing $Y_l(s_l)$ can take more then constant time. The upper boundary of CH_t has at most n points so $Y_l(s_l)$ can be computed in $O(\log n)$ time. Hence, FindPlacementCase22 runs in $O(n \log n)$ time. □

Next we show that FindPlacement returns YES if and only if there is a placement for s_l and s_r such that the convex hull of s_l, s_r and $CH_{\mathcal{C}'}$ intersects every region in $\mathcal{R}$.

Lemma 22. FindPlacement *returns YES if and only if there is a placement for* s_l *and* s_r *such that the convex hull of* s_l *and* s_r *and the given convex hull* $CH_{\mathcal{C}'}$ *intersects every region in* $\mathcal{R}$.

Proof. The convex hull $CH_{\mathcal{C}'}$ can either be inside the quadrangle $s_{lt}s_{rt}s_{rb}s_{lb}$ or outside. If it is inside the polygon then it follows from Lemma 15 that FindPlacement returns YES if and only if a correct placement exists.

If $CH_{\mathcal{C}'}$ is completely inside $s_{lt}s_{rt}s_{rb}s_{lb}$ we again distinguish two cases. If there is a placement with a convex hull that has an indirect connection as the upper and lower bound then by Lemma 17 the algorithm FindPlacementCase21 will return YES. Hence, also FindPlacement returns YES. If there is a solution of the second kind, so either the upper or lower boundary of $CH_{\mathcal{C}}$ is a direct connection, then by Lemma 20 the algorithm FindPlacementCase22 will return YES. Lastly if there is no placement such that $CH_{\mathcal{C}}$ has a non-empty intersection with all regions of $\mathcal{R}$, then by Lemmas 18 and 20 the algorithm FindPlacement will return NO. □

Next we look at the running time of the algorithms described here. FindPlacement is split into three cases, so we will use the running times of these separate algorithms to determine the running time for FindPlacement.

Lemma 23. FindPlacement *runs in* $O(n \log n)$.

Proof. This follows directly from Lemmas 16, 19 and 21. □

So far we have assumed that S_l and S_r are the unique leftmost and rightmost line segments. However it is possible that for example there are multiple leftmost line segments. Because the regions are disjoint they have some vertical order. We are only interested in maximal convex hulls, but it is easy to see that in case of multiple leftmost line segments there is only one placement that can lead to a maximal convex hull. The topmost line segment will have a point placed at its upper endpoint and the bottommost line segment will have a point at

its lower endpoint. If you place them anywhere else the convex hull can be made bigger by moving them towards these endpoints. The problem is then made easier since we only have to look at different placements for S_r.

From Lemmas 11, 22 and 23 we can conclude the following theorem.

Theorem 9. *Given a set $\mathcal{R}$ of n disjoint parallel line segments of unit length and a subset $\mathcal{C} \subseteq \mathcal{R}$ containing k line segments, then* FindMaxHull *computes the correct result for* Maximal Convex Hull *in $O(k^2 n \log n)$ time.*

Due to Lemma 10 we then get the following corollary.

Corollary 4. *Given a set $\mathcal{R}$ of n disjoint parallel line segments of unit length and a subset $\mathcal{C} \subseteq \mathcal{R}$ containing k line segments,* Superset Convex Hull *can be solved in $O(k^2 n \log n)$ time for these sets.*

Chapter 5

Subset Convex hull

A different variant of the convex hull problem on imprecise points is that of determining if there is a placement where the points from $\mathcal{C}$ are a subset of the vertices of the convex hull.

Subset Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: YES if there is a placement of points such that the points from regions in $\mathcal{C}$ are a subset of the vertices of the convex hull, NO otherwise.

This problem can be reduced to a slightly different problem, which will be called the Minimal Convex hull problem. Analogous to a maximal convex hull a *minimal convex hull* is a minimal element of the partial order of all possible convex hulls under inclusion. For a minimal convex hull it holds that no other convex hull is contained in it.

In the Minimal convex hull problem the goal will be to find a minimal convex hull where all regions of $\mathcal{C}$ have some part outside it or contribute a vertex.

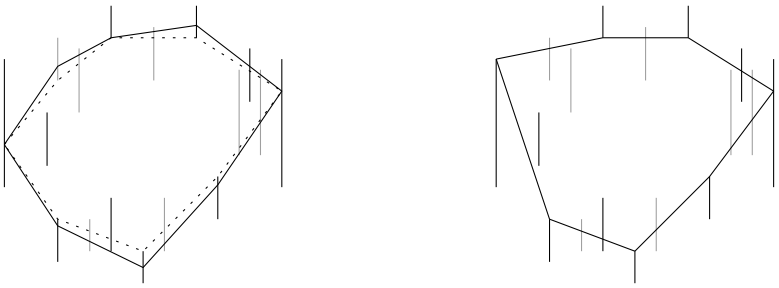
Minimal Convex Hull
Input: A set of regions $\mathcal{R}$ and a subset $\mathcal{C} \subseteq \mathcal{R}$.
Output: YES if there is a placement of points such that its convex hull CH is minimal and for every region $P \in \mathcal{C}$, either $P \backslash CH \neq \emptyset$ or the point from P is a vertex of the convex hull, NO otherwise.

Figure 5.1a illustrates the difference between a minimal and a non-minimal convex hull. The vertical line segments are the regions of $\mathcal{R}$ and the grey line segments are the ones in $\mathcal{C}$. The solid polygon shows a convex hull for some placement of $\mathcal{R}$. However, this convex hull is not minimal since the dotted polygon is also a possible convex hull and is contained within the solid polygon. The solid polygon is not a solution to Minimal Convex Hull whereas the dotted polygon is.

Figure 5.1b illustrates a different minimal convex hull which also is not a solution to Minimal Convex Hull because not all regions of $\mathcal{C}$ have some part outside the convex hull.

Lemma 24. *When the input regions are convex and disjoint there is a solution for* Subset Convex Hull *if and only if there is a solution to* Minimal Convex Hull.

Proof. Let the placement V be a solution for Subset Convex Hull and $V_{\mathcal{C}} \subseteq V$ be the points from regions in $\mathcal{C}$. On the convex hull of V, $CH(V)$, all points from $V_{\mathcal{C}}$ are vertices. This convex hull does not have to be minimal, but by definition there is always a minimal



(a) non-minimal convex hull

(b) a line segment from $\mathcal{C}$ is inside the convex hull

Figure 5.1: Convex hulls that are not solutions to Minimal Convex Hull

convex hull CH_{min} that is included in $CH(V)$. Each vertex p of $CH(V)$ is then either still a vertex of CH_{min} or it is not in CH_{min}.

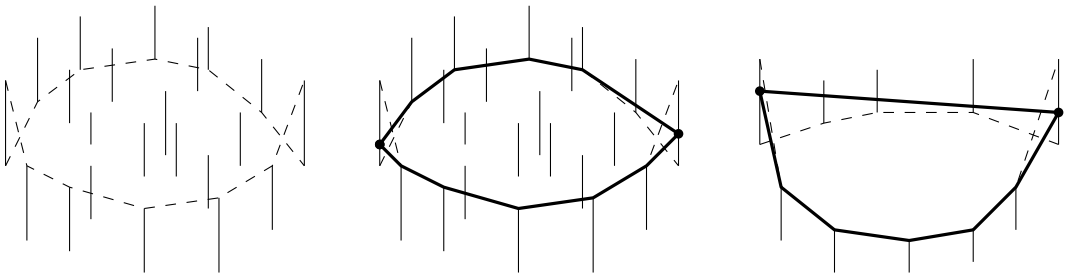
Given a solution to Minimal Convex Hull it is also possible to construct a solution to Subset Convex Hull. Because every region $P \in \mathcal{C}$ that does not yet contribute a vertex to the convex hull has some part outside the convex hull and these regions are convex it is possible to place the point of P an arbitrarily small distance outside the convex hull. Because the regions are disjoint this point can always be placed close enough to the convex hull such that it does not cause another vertex to be inside the convex hull. This way all points of regions in $\mathcal{C}$ become vertices of the convex hull and so the result is a solution to Subset Convex Hull. It is also possible that for an edge of the boundary of the minimal convex hull there are multiple regions of $\mathcal{C}$ that intersect it. Then points in these regions should be placed on a convex chain that is close enough to the edge such that replacing the edge by the chain still yields a convex polygon. □

Minimal convex hulls have the property that for each vertex, this vertex is the only point of intersection of the convex hull and the region for which the vertex is a placement. We prove this in the following lemma.

Lemma 25. *A convex hull CH is minimal if and only if there is a region P for each vertex p such that $CH \cap P = \{p\}$.*

Proof. Let CH be a minimal convex hull and S be the placement of points that has CH as its convex hull. Then let p be a vertex of CH and P the region that has p as its placement. Now assume that $P \cap CH \supset p$ so the intersection of P and CH contains more points than just p. Then we can choose for P a point $q \in P \cap CH$ different from p. Then for a placement where q is added and p is removed the new convex hull CH' is smaller then CH, because p is not in CH'. Therefore p is the only point in the intersection of P and CH if CH is a minimal convex hull.

Let CH be a convex hull where for each vertex p from a region P it holds that $CH \cap P = \{p\}$. Then assume that CH is not minimal and there is a convex hull CH_{min} which is a subset of CH. Because CH_{min} must contain a point of each region and only contains points that are in CH every vertex p of CH must also be part of CH_{min}. It then follows that CH_{min} contains at least all points of CH since it contains all vertices of CH, thus CH_{min} cannot be a subset of CH, so CH must be minimal. □



(a) bottom and top chains

(b) convex hull by choosing s_l and s_r

(c) s_l and s_r should be connected directly

Figure 5.2: bottom and top chains and convex hulls construct by connecting points on the left and rightmost line segment to each other through the chains

In the next sections it will be shown how to compute a solution for Minimal Convex Hull for a set of disjoint parallel line segments or a set of disjoint axis aligned rectangles. In these sections we will only explain how to find a minimal convex hull that is a solution to Minimal Convex Hull without detailing the placement that leads to this convex hull. However, such a placement is easily constructed since every region will intersect the minimal convex hull.

5.1 Disjoint parallel line segments

In this section we will discuss an algorithm to solve the minimal convex hull problem for parallel line segments.

Mukhopadhyay et al. introduce the notions of a bottom and top chain on a set of vertical line segments [MKGB08]. The bottom chain is the lower boundary of the convex hull of all upper endpoints of the line segments, whereas the top chain is the upper boundary of the convex hull of lower endpoints. In Figure 5.2a the top and bottom chains are indicated with dotted lines for a set of vertical line segments.

Let S_l and S_r be the leftmost and rightmost line segment. For now we assume there is only one leftmost line segment and one rightmost, so they are uniquely defined. We define $tl'_t(s)$ for a point s that is either on the vertical line through S_l or on the vertical line through S_r as the line that is tangent to the top chain and goes through s. We then define $tl_t(s)$ to be the segment of the line $tl'_t(s)$ from s to the chain. The line $tl_b(s)$ is defined symmetrically for the bottom chain.

If we take two points $s_l \in S_l$ and $s_r \in S_r$ then we can construct a convex hull $CH(s_l, s_r)$ as follows. In general s_l and s_r are connected to the chains with tangent lines $tl_t(s_l), tl_b(s_l), tl_t(s_r)$ and $tl_b(s_r)$ as shown in Figure 5.2b. However it may be the case that the $tl_t(s_l)$ and $tl_t(s_r)$ to the top chain cross or connect to the same point and make a left angle when going from s_l to s_r. In this case, instead of connecting s_l and s_r to the top chain, they should be connected to each other using a single line segment, as shown in Figure 5.2c. The same applies if $tl_b(s_l)$ and $tl_b(s_r)$ cross or make a right angle when going from s_l to s_r.

An important property of the convex hulls that are constructed like this is that they are all minimal convex hulls.

Lemma 26. *Each convex polygon $CH(s_l, s_r)$ constructed from a choice of points $s_l \in S_l$ and $s_r \in S_r$ is a minimal convex hull.*

Proof. The vertices of the upper boundary of $CH(s_l, s_r)$ are vertices of the top chain and s_l and s_r. The vertices of the chain are by definition the lower endpoints of line segments, so they are the only points of those line segments that are on the convex hull. Since S_l and S_r are the unique leftmost and rightmost line segment the convex hull only intersects S_l in s_l and S_r in s_r. By similar reasoning also the vertices on the lower boundary of $CH(s_l, s_r)$ are the only points of their line segments on the convex hull.

Then it follows from Lemma 25 that $CH(s_l, s_r)$ is a minimal convex hull. □

We can also show that every minimal convex hull is equal to $CH(s_l, s_r)$ for some $s_l \in S_l$ and $s_r \in S_r$.

Lemma 27. *For every minimal convex hull CH_{min} there are points $s_l \in S_l$ and $s_r \in S_r$ such that $CH_{min} = CH(s_l, s_r)$.*

Proof. Let's assume there is a minimal convex hull CH_{min} for which no points s_l and s_r exist such that $CH(s_l, s_r) = CH_{min}$. From Lemma 25 and the fact that S_l and S_r are unique left and rightmost line segments it follows that CH_{min} contains exactly one point s_l on S_l and one point s_r on S_r.

If $CH(s_l, s_r) \neq CH_{min}$ then because both are minimal CH_{min} must contain a point that is not in $CH(s_l, s_r)$. Because $CH(s_l, s_r)$ is convex there is a vertex p of CH_{min} that is not in $CH(s_l, s_r)$. Let's assume that p is above $CH(s_l, s_r)$, the case when p is below $CH(s_l, s_r)$ is analogous. Because p is above $CH(s_l, s_r)$ it is also above the top chain, which implies it is not the lower endpoint of its region P. It then follows from Lemma 25 that CH_{min} is not a minimal convex hull, so $CH(s_l, s_r)$ must be equal to CH_{min}. □

From Lemma 27 it follows that to find a solution to Minimal Convex Hull we only have to look at different placements of s_l and s_r and the top and bottom chains. In particular we want to find a placement of s_l and s_r such that each region in $\mathcal{C}$ is partly outside $CH(s_l, s_r)$ or contribute a vertex to the convex hull.

Because $CH(s_l, s_r)$ only depends on s_l and s_r and these can be anywhere on a single line segment we can represent all convex hulls in a rectangle in $\mathbb{R}^2$ with y-coordinates for s_l on one axis and y-coordinates for s_r on the other. This rectangle will be called the *solution space* and is denoted by S. For each region $P \in \mathcal{C}$ we can determine the region S_P of the solution space where P is partly outside $CH(s_l, s_r)$ or is a vertex of it. The intersection of all these regions then represents the solutions where every region of $\mathcal{C}$ has a part outside $CH(s_l, s_r)$.

First we will describe what constraints a region $P \in \mathcal{C}$ imposes on s_l and s_r and how we can translate that to a region in the solution space.

The constraints that a region $P \in \mathcal{C}$ imposes on s_l and s_r are based on the fact that either the upper endpoint p_u should be above the convex hull or the lower endpoint p_l should be below the convex hull. Let $s_{tl}(p_u)$ be the point on the vertical line through S_l such that p_u is on $tl_t(s_{tl}(p_l))$. The point $s_{tr}(p_u)$ is a point on the vertical line through S_r such that p_u is on $tl_t(s_{tr}(p_u))$. These lines are illustrated in Figure 5.3 Now the following conditions hold for p_u if and only if it is above the upper boundary of the convex hull.

(u_1) s_l is below $s_{tl}(p_u)$
(u_2) s_r is below $s_{tr}(p_u)$

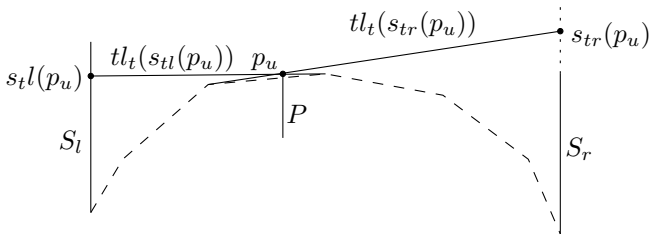


Figure 5.3: The lines $tl_t(s_{tl}(p_u))$ and $tr_t(s_{tr}(p_u))$ are tangent to the chain and go through p_u

(u_3) p_u is above the line $s_l s_r$
(u_4) p_u is above the top chain or a vertex on it

There are two exceptional cases where not all of these requirements are properly defined, more precisely in some cases $s_{tl}(p_u)$ and $s_{tr}(p_u)$ are not properly defined. This is the case when the region P is either S_l or S_r. If that is the case, then we can simply ignore them when making these requirements. By construction if there is a unique S_l and S_r the points s_l or s_r are always a vertex on $CH(s_l, s_r)$. The other case is when the upper endpoint p_u is under the top chain. In that case there is no point $s_{tl}(p_u)$. Again this is no problem since in this case u_4 does not hold, so p_u is not above the convex hull.

Now we can prove that p_u is above the $CH(s_l, s_r)$ for given points s_l and s_r if and only if the conditions u_1 to u_4 hold.

Lemma 28. *p_u is above $CH(s_l, s_r)$ if and only if the conditions u_1 to u_4 hold.*

Proof. First to prove that if one of the conditions does not hold, then p_u is below the top connection.

- If u_1 does not hold, then $tl_t(s_l)$ is above p_u or goes through it. Hence, p_u cannot be above the convex hull.
- If u_2 does not hold, then $tl_t(s_r)$ is above p_u or goes through it. Hence, p_u cannot be above the convex hull.
- If u_3 does not hold, then p_u is below or on the line $s_l s_r$, since both s_l and s_r are part of the convex hull, the entire line $s_l s_r$ is inside the convex hull. Since p_u is below or on the line it can never be above the convex hull.
- If u_4 does not hold, then p_u cannot be above the upper boundary of the convex hull or a vertex on it since the upper boundary cannot be below the top chain.

Next, we prove that whenever p_u is below or on the upper boundary of the convex hull, but not a vertex, one of the conditions does not hold. If the upper boundary is just the line $s_l s_r$ and p_u is below or on this line obviously u_3 does not hold.

If the upper boundary consists of a part of the top chain and lines through s_l and s_r that are tangent to the top chain, then p_u can be in any of the following three areas. Each leads to at least one the conditions u_1 to u_4 not holding.

- If p_u is below or on the top chain, but not a vertex of it then clearly u_4 does not hold
- If p_u is between the top chain and $tl_t(s_l)$ or on $tl_t(s_l)$. Then s_l is above or on $s_{tl}(p_u)$ because p_u is below or on $tl_t(s_{tl}(p_u))$.

- Symmetrically p_u can be between the top chain and $tl_t(s_r)$ or on $tl_t(s_r)$. By the same reasoning as in the previous case u_2 does not hold in this case.

We can conclude that p_u is above the top chain if and only if u_1 to u_4 hold. □

For the lower endpoint $p_l \in P$ similar conditions, l_1 to l_4, can be given. These hold whenever p_l is below $CH(s_l, s_r)$ or p_l is a vertex on the lower boundary of $CH(s_l, s_r)$. The point $s_{bl}(p_l)$ is defined as the point on the vertical line through S_l such that p_l is on the line $tl_b(s_{bl}(p_l))$. The point $s_{br}(p_l)$ is on the vertical line through S_r and defined similarly.

(l_1) s_l is above $s_{bl}(p_l)$
(l_2) s_r is above $s_{br}(p_l)$
(l_3) p_l is below the line $s_l s_r$
(l_4) p_l is below the bottom chain or a vertex on it

Lemma 29. *p_l is below the bottom connection from s_l and s_r if and only if the conditions l_1 to l_4 hold.*

Proof. This is symmetric to the proof of Lemma 28. □

For each region $P \in \mathcal{C}$ either the upper endpoint or the lower endpoint must be outside the convex hull or be a vertex on it. Let $u_1(P)$ to $u_4(P)$ and $l_1(P)$ to $l_4(P)$ denote the constraints u_1 to u_4 and l_1 to l_4 respectively for a region P. Then either $u_1(P)$ to $u_4(P)$ or $l_1(P)$ to $l_4(P)$ should be true. These restrictions on s_l and s_r can be formulated as in Formula 5.1.

$$\bigwedge_{P \in \mathcal{C}} \left(\bigwedge_{i=1}^{4} u_i(P) \vee \bigwedge_{i=1}^{4} l_i(P) \right) \tag{5.1}$$

Lemma 30. *s_l and s_r define a minimal convex hull $CH(s_l, s_r)$ where each region of $\mathcal{C}$ has some part outside the convex hull or contributes a vertex to it if and only if Formula 5.1 is true.*

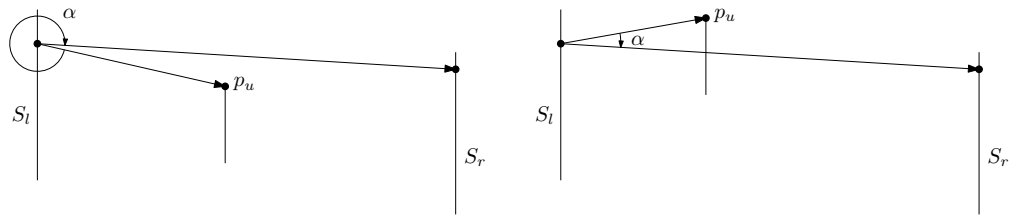
Proof. By Lemmas 28 and 29 if any region $P \in \mathcal{C}$ is completely inside $CH(s_l, s_r)$ and does not contribute a vertex to it, then both $\bigwedge_{i=1}^{4} u_i(P)$ and $\bigwedge_{i=1}^{4} l_i(P)$ are false. So indeed Formula 5.1 is also false.

If Formula 5.1 is false then for some P both $\bigwedge_{i=1}^{4} u_i(P)$ and $\bigwedge_{i=1}^{4} l_i(P)$ do not hold then by Lemmas 28 and 29 the line segment P is completely inside the $CH(s_l, s_r)$ and does not contribute a vertex to it. □

Each of these conditions $u_1(P)$ to $u_3(P)$ and $l_1(P)$ to $l_3(P)$ can be represented in an inequality in the y coordinates of s_l and s_r as follows.

For $u_1(P)$ and $u_2(P)$ the inequalities are easy since we only have to compare the y-coordinates. For $u_1(P)$ the inequality is $y_{s_l} < y_{s_{tl}(p_u)}$ and for $u_2(P)$ the inequality is $y_{s_r} < y_{s_{tr}(p_u)}$.

For u_3 the inequality is somewhat more difficult since it depends on both s_l and s_r. The point p_u is above the line $s_l s_r$ if and only if the angle between the vectors $\overrightarrow{s_l p_u}$ and $\overrightarrow{s_l s_r}$ is between 0 and π as illustrated in Figure 5.4. If we view these as three dimensional vectors where the z component is 0 we can take the crossproduct $\overrightarrow{s_l p_u} \times \overrightarrow{s_l s_r}$ of the two vectors. The resulting vector has the form $(0, 0, z)$. If z is positive then the angle between $\overrightarrow{s_l p_u}$ and $\overrightarrow{s_l s_r}$ is



(a) p_u below $s_l s_r$, $\pi < \alpha < 2\pi$ (b) p_u above $s_l s_r$, $0 < \alpha < \pi$

Figure 5.4: The angle α from the vector $\overrightarrow{s_l p_u}$ to $\overrightarrow{s_l s_r}$ is related to p_u being above or below $s_l s_r$

between 0 and π. If z is negative then the angle is between π and 2π. Lastly if z is 0 then then angle is either 0 or π. This means we want the z component of the cross product to be more then 0. The constraint is then as follows, where constants are indicated by X or Y, whereas variables are indicated by x or y.

$$(X_{s_r} - X_{s_l})(Y_{p_u} - y_{s_l}) - (X_{p_u} - X_{s_l})(y_{s_r} - y_{s_l}) > 0 \tag{5.2}$$

Equation 5.2 is a linear inequality in y_{s_l} and y_{s_r}. The inequality for l_3 is similar.

For u_4 and l_4 we do not give an equation since they do not depend on s_l or s_r. Computing whether or not they are true can be done by going through all line segments ordered by x-coordinate while following the top and bottom chains.

Each of these inequalities defines a region in the solution space. Let $u'_1(P)$ to $u'_4(P)$ and $l'_1(P)$ to $l'_4(P)$ be the regions of the solution space where $u_1(P)$ to $u_4(P)$ and $l_1(P)$ to $l_4(P)$ hold respectively. The regions in the solution space where all regions of $\mathcal{C}$ are partly outside the convex hull or contribute a vertex to it are then defined by Formula 5.3. The regions are combined as described by Algorithm 6. First for each region $P \in \mathcal{C}$ the region of the solution space is computed where at least one endpoint is outside the convex hull. For a region P this is $\bigcap_{i=1}^{4} u'_i(P) \cup \bigcap_{i=1}^{4} l'_i(P)$. Then these regions in the solution space are combined using a divide and conquer approach where the intersection of a set $\mathcal{A}$ of regions is computed by splitting $\mathcal{A}$ into $\mathcal{A}_1$ and $\mathcal{A}_2$ and computing the intersection of regions in $\mathcal{A}_1$ and $\mathcal{A}_2$ first and then combining those using a map overlay algorithm. For storing the regions in the solutions space doubly connected edge lists are used [BCKO08, Chapter 2]. Note that because we are only interested in the solution space, lines that go outside the solution space are cut off so we only have to deal with line segments and not lines.

$$\bigcap_{P \in \mathcal{C}} \left(\bigcap_{i=1}^{4} u'_i(P) \cup \bigcap_{i=1}^{4} l'_i(P)\right) \tag{5.3}$$

Lemma 31. *Algorithm 6 runs in $O(n \log n + k^2)$ time.*

Proof. For each region in $\mathcal{C}$ for the upper endpoint p_u the tangent points for $tl_t(s_{tl}(p_u))$ and $tl_t(s_{tr}(p_u))$ should be found and it should be checked if p_u is above or below the top chain. Testing if p_u is above or below the top chain can be done using a binary search on the x-coordinates of the points on the chain to find the segment directly above or below p_u. Since there are $O(n)$ points on the chain this takes $O(\log n)$ time. To find the tangent points also a binary search can be done on the vertices of the chain. When inspecting a point the direction

Algorithm 6: FindMinHull($\mathcal{R}, \mathcal{C}$)

```
A ← ∅
forall P ∈ C do
  uspace ← ℝ²
  lspace ← ℝ²
  for i ∈ [1...3] do
    u'_i ← Region where u_i(P) holds
    uspace ← uspace ∩ u'_i
    l'_i ← Region where l_i(P) holds
    lspace ← lspace ∩ l'_i
  if p_u is below top chain then
    uspace ← ∅
  if p_l is above bottom chain then
    lspace ← ∅
  A ← A ∪ {uspace ∪ lspace}
solspace ← ComputeIntersection(A)
if solspace = ∅ then
  return There is no solution
else
  return a point in solspace
```

of the search is determined by which of the neighboring points is above or below the line through p_u and the point that the search is currently inspecting. For example if the previous point is above the line then the search continues in that direction and if both neighboring points are below the line, then the current point is the tangent point. Again because there are $O(n)$ vertices on the chain this takes $O(\log n)$ time.

Once these points are known, $s_{tl}(p_u)$ and $s_{tr}(p_u)$ are easily computed and the requirements can be translated into regions in the solution space. For each region P computing the area in which at least one endpoint of P is outside the convex hull or a vertex of it can be done in $O(1)$ time the, since only a constant number of lines define the region. So computing $\mathcal{A}$ takes $O(n \log n)$ time.

ComputeIntersection then computes the intersection of all regions in $\mathcal{A}$. This is done by computing the arrangement of the line segments of the constraints. This takes $O(k^2)$ time using a topological sweep as described by Edelsbrunner and Guibas [EG89]. Then, for each cell in the arrangement we compute in how many regions of $\mathcal{A}$ it is by traversing the faces of the arrangement. When going from one face to the next we cross only one line of the arrangement so we only have to check for one region if the next face is contained in this region or not. Since this traversal is the same as traversing the dual graph of the arrangement this can also be done in $O(k^2)$ time. The total running time of Algorithm 6 is then $O(n \log n + k^2)$. □

So far we have assumed that S_l and S_r are the unique leftmost and rightmost line segments. However it is possible that there are multiple line segments that are vertically aligned that all have the same x-coordinate. In that case the problem just becomes easier since we still require for a minimal convex hull that vertices are the only points of their region that intersect the convex hull. If there are several leftmost line segments then the only placements of points leading to a minimal convex hull have the point on the top line segment at its lower endpoint

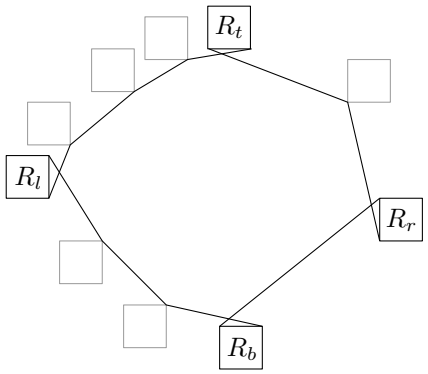


Figure 5.5: Chains between the extreme regions

and on the bottom line segment at its top endpoint. All minimal convex hulls then correspond to a placement of s_r only.

5.2 Disjoint axis aligned rectangles

The method for solving the minimal convex hull problem for disjoint parallel line segments can be adapted to work for disjoint axis aligned rectangles.

For rectangles there is an *extreme region* in all four direcions. R_l is the region with the leftmost right edge, R_t is the region with the topmost bottom edge, R_r is the region with rightmost left edge and R_b is the region with bottommost top edge.

First consider the general case where each of these rectangles is unique and they are *nicely positioned*. This means that R_t is completely above R_l, R_r and R_b, R_l is completely to the left of R_t, R_b and R_r and R_r and R_b have similar positions. It then follows from Lemma 25 that points on these rectangles can only be placed on one edge. For example for R_l this is the right edge, placing it anywhere else would lead to the intersection of the convex hull with R_l containing more then a single point. These edges are the *extreme edges* and are denoted by S_l, S_t, S_r and S_b.

Between these extreme edges, chains can be defined based on the corners of regions as illustrated in Figure 5.5. The top left chain between S_l and S_t is defined as the top left part of the convex hull of lower right endpoints. For no convex hull, the top left boundary is below the top left chain, because then a point of this chain and the region it belongs too would lie completely outside the convex hull. The top right, bottom left and bottom right chains are defined similarly.

Using these chains we can define a convex hull based on placements of points on the four extreme edges as we did for line segments. $CH(s_l, s_t, s_r, s_b)$ is the convex hull created by connecting each two consecutive extreme points to each other using the chains. For example s_l and s_t are connected by a single line segment if the whole top left chain is below this line segment. Otherwise the top left boundary of the convex hull consists of a line segment from s_l to the top left chain that is tangent to the chain, then a part of the top left chain and then a line segment from s_t to the chain that is also tangent to the chain. If a connection between two extreme points is just a single line segment we call it a *direct connection* if it consists of a part of the chain and line segments connecting the extreme points to it, that is called a *chain connection*.

Using the chains and the extreme edges we prove the following lemmas.

Lemma 32. *For a set of regions where the extreme regions are unique and nicely positioned, $CH(s_l, s_t, s_r, s_b)$ is a minimal convex hull for every choice of points s_l, s_t, s_r, s_b on the extreme edges.*

Proof. Lets assume there is a convex hull CH_{min} that is a subset of $CH(s_l, s_t, s_r, s_b)$. Then CH_{min} contains at least s_l, s_t, s_r and s_b because these are the only points of the extreme rectangles in $CH(s_l, s_t, s_r, s_b)$. The other vertices of $CH(s_l, s_t, s_r, s_b)$ are vertices of the chain, but by definition of the chains and the construction of $CH(s_l, s_t, s_r, s_b)$ these vertices are then the only points of the corresponding regions that are contained in $CH(s_l, s_t, s_r, s_b)$. Hence all vertices of $CH(s_l, s_t, s_r, s_b)$ are also contained in CH_{min} and CH_{min} cannot be a subset of $CH(s_l, s_t, s_r, s_b)$. We then conclude that every convex hull constructed from a choice of points on the extreme edges must be minimal. □

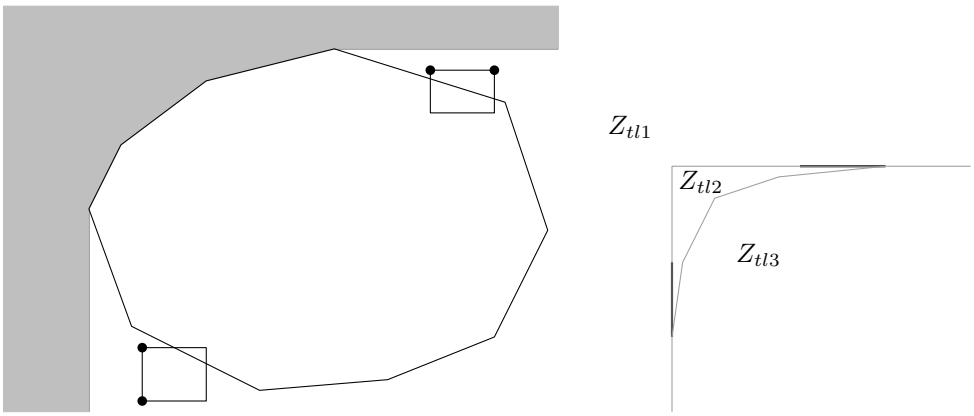
Lemma 33. *For a set of regions where the extreme regions are unique and nicely positioned every minimal convex hull CH_{min} is equal to $CH(s_l, s_t, s_r, s_b)$ for some choice of points s_l, s_t, s_r and s_b on the extreme edges.*

Proof. Because we are assuming that the extreme rectangles are nicely positioned the points from them have to be vertices of the convex hull, namely the leftmost, topmost, bottommost and rightmost point of the convex hull. Then by Lemma 25 there is only one point of the extreme rectangles on the convex hull. These are the only candidates for s_l, s_t, s_r and s_b. Let CH be the convex hull $CH(s_l, s_t, s_r, s_b)$. Then for CH_{min} to be different from CH it has to have a vertex outside of CH, because CH is minimal (see Lemma 32).

Without loss of generality assume that this vertex p is on the top left boundary of CH_{min}. Since p is outside CH, it must also be above the top left chain. Then by definition of the top left chain p is not the lower right endpoint of its region P. Then because p is on the top left boundary the lower right endpoint of P must be inside the convex hull. From Lemma 25 it then follows that CH_{min} is not a minimal convex hull. □

To find a good placement for the extreme points we would like to use an adapted version of the approach used for line segments. We make constraints for the corner points of $\mathcal{C}$ which can be combined to hold if and only if there is a convex hull with all regions of $\mathcal{C}$ having some part outside this convex hull.

Recall that for line segments, a line segment has some part outside the convex hull if either the top endpoint is above the convex hull or the bottom endpoint is below the convex hull. With rectangles at least one of the corners should be outside the convex hull. We say that a top left corner of a rectangle is *outside* the top left boundary of the convex hull if it is either above or to the left of the top left boundary as indicated by the grey region in Figure 5.6a. Vertices of the top left boundary are also considered to be outside the top left boundary. Using this notion of outside a boundary, a rectangle has a corner outside the convex hull if and only the top left corner is outside the top left boundary, the top right corner is outside the top right boundary, the bottom left corner is outside the bottom left boundary or the bottom right corner is outside the bottom right corner. It is easy to see that this is equivalent to saying that at least one corner is outside the convex hull. If the top left corner is outside the convex hull but not outside the top left boundary, then the bottom left or top right corner must also be outside the convex hull as illustrated by the two rectangles in Figure 5.6a.



(a) Illustration of the area *outside* the top left boundary of a convex hull

(b) Top left corners are split into three sets based on their position

Figure 5.6

Based on the top left chain we can divide top left corners into three sets. These sets are labeled Z_{tl1}, Z_{tl2} and Z_{tl3} in Figure 5.6b. Let S'_l be the vertical line through S_l and S'_t the horizontal line through S_t. Then the sets Z_{tl1}, Z_{tl2} and Z_{tl3} are defined as follows; points in Z_{tl1} are above or on S'_t or to the left of or on S'_l; points in Z_{tl2} are below S'_t, to the right of S'_l and above the chain, but also the vertices of the chain are in Z_{tl2}; points in Z_{tl3} are below S'_t, to the right of S'_l and below or on the chain, however the vertices of the chain are not in Z_{tl3}.

Clearly points in Z_{tl1} are always outside the top left boundary and points in Z_{tl3} are never outside the top left boundary. Whether or not a point in Z_{tl2} is inside or outside the top left boundary of the convex hull depends on the choice of s_l and s_t. Testing in which set a point is can easily be done using a binary search on the x-coordinates of the points in the chain. For each point p_{tl} in Z_{tl2} we define constraints on s_l and s_t that hold if and only if p_{tl} is outside the top left boundary of $CH(s_l, s_t, s_r, s_b)$. These constraints make use of two lines, l_{ltl} and l_{ttl}, which are defined as follows. For a region P, the line $l_{ltl}(P)$ is tangent to the top left chain and intersects the top left corner p_{tl} of P. The line l_{ltl} crosses the vertical line through S_l, p_{tl} and the chain in that order. $l_{ttl}(P)$ is defined similarly except it crosses the horizontal line through S_t, p_{tl} and the chain in that order. The constraints are then tl_1, tl_2 and tl_3.

(tl_1) s_l is below where l_{ltl} crosses the vertical line through S_l
(tl_2) s_t is to the right of where l_{ttl} crosses the horizontal line through S_t.
(tl_3) p_{tl} is above the line $s_l s_t$

As with vertical line segments the lines l_{ltl} and l_{ttl} are not well defined when p_{tl} is a vertex of the top left chain. If p_{tl} is a vertex of the top left chain then l_{ltl} should not just intersect p_{tl} on the chain, but also the next point on the chain closer to s_t. Similarly l_{ttl} should not just hit p_{tl} on the chain but also the next point towards s_l on the chain.

The requirements tl_1, tl_2 and tl_3 can then be turned into inequalities that hold only if the requirements hold using the cross product of two vectors (see Section 5.1). These inequalities have only y_{s_l} and x_{s_t} as variables. For example tl_3 holds if and only if the following inequality holds. Note that capitol letters, X and Y, indicate that this coordinate is fixed, while lower case letters, x and y, indicate the coordinate is variable.

$$(x_{s_t} - X_{s_l})(Y_{p_{tl}} - y_{s_l}) - (X_{p_{tl}} - y_{s_l})(Y_{s_t} - y_{s_l}) > 0$$

The variables here are x_{s_t} and y_{s_l} so this inequality is not linear, but we will show later that this is not a problem. Similar requirements and equations can also be made for the top right (tr_1 to tr_3), bottom left (bl_1 to bl_3) and bottom right (br_1 to br_3) corners of a rectangle.

These constraints can be combined as follows to hold if and only if at least one corner is outside $CH(s_l, s_t, s_r, s_b)$.

$$p_{tl} \in Z_{tl1} \vee (p_{tl} \notin Z_{tl3} \wedge \bigwedge_{i=1}^{3} tl_i) \vee p_{tr} \in Z_{tr1} \vee (p_{tr} \notin Z_{tr3} \wedge \bigwedge_{i=1}^{3} tr_i) \vee$$
$$p_{bl} \in Z_{bl1} \vee (p_{bl} \notin Z_{bl3} \wedge \bigwedge_{i=1}^{3} bl_i) \vee p_{br} \in Z_{br1} \vee (p_{br} \notin Z_{br3} \wedge \bigwedge_{i=1}^{3} br_i)$$

The minimal convex hull now depends on four variables, s_l, s_t, s_r and s_b. We can represent all minimal convex hulls as points in a 4-dimensional object which we call the *solution space*. Each point in the solution space represents a minimal convex hull. Each equation of a requirement then holds in a region of the solution space. These regions are defined by the equations we define for each requirement.

For a rectangle $P \in \mathcal{C}$ we can combine these regions of the solutions space. Let tl'_1 to tl'_3 be the regions where tl_1 to tl_3 hold, tr'_1 to tr'_3 where tr_1 to tr_3 holds, etcetera. Let Z'_{tl1} either be $\mathbb{R}^4$ if p_{tl} is in Z_{tl1} or $\emptyset$ if p_{tl} is not in Z_{tl1} and let Z'_{tl2} and Z'_{tl3} be defined similarly. For the other corners and regions we use a similar definition. The region $R(P)$ of the solution space which contains only points that represent convex hulls where P has a corner outside of them or is a vertex of the boundary is then defined as follows.

$$R(P) = Z'_{tl1} \cup (Z'_{tl2} \cap \bigcap_{i=1}^{3} tl'_i) \cup Z'_{tr1} \cup (Z'_{tr2} \cap \bigcap_{i=1}^{3} tr'_i) \cup$$
$$Z'_{bl1} \cup (Z'_{bl2} \cap \bigcap_{i=1}^{3} bl'_i) \vee Z'_{br1} \cup (Z'_{br2} \cap \bigcap_{i=1}^{3} br'_i)$$

Then a solution to Minimal Convex Hull can be found by taking the intersection of $R(P)$ over all regions $P \in \mathcal{C}$.

To find the intersection of these regions we compute the arrangement of the surfaces that define the regions. This arrangement has $O(k^4)$ cells, because there are $O(k)$ surfaces and the solution space has four dimensions. It should then be checked for each cell if it is contained in all regions or not. Computing this arrangement can be done using techniques described by Chazelle et al. [CEGS91]. The time required for this is dependant on the complexity of the decomposition of these cells into constant complexity cells. Koltun has shown that the complexity the vertical decomposition of an arrangement of k surfaces in four dimensions has a complexity of $O(n^{4+\varepsilon})$ for any $\varepsilon > 0$ [Kol04]. This implies that we can compute the cells in $O(k^{4+\varepsilon})$ time. As with line segments we can traverse the cells of this arrangement to test in $O(k^{4+\varepsilon})$ to determine for each cell how many rectangles of $\mathcal{C}$ contribute a vertex or are partly outside the convex hull.

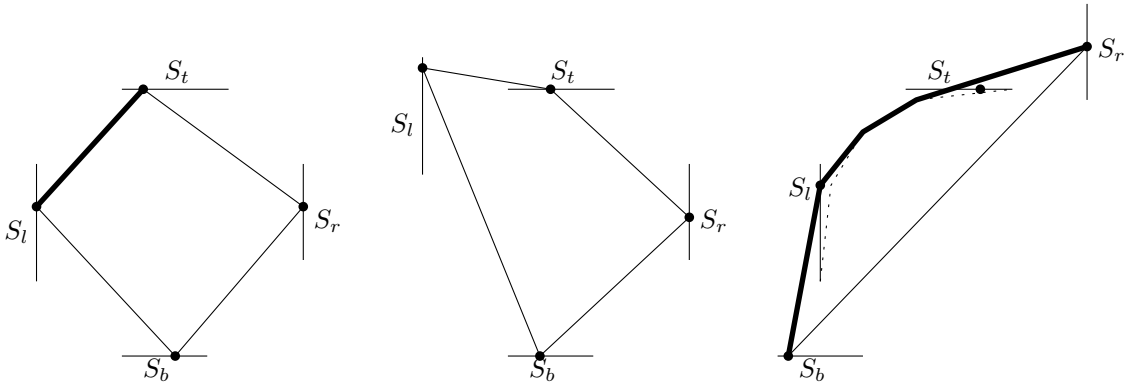
We can then use this to show that Minimal Convex Hull can be solved in $O(n \log n + k^{4+\varepsilon})$ time for $\varepsilon > 0$.

Lemma 34. Minimal Convex Hull *can be solved in* $O(n \log n + k^{4+\varepsilon})$ *time for any* $\varepsilon > 0$*, if the extreme rectangles are unique and nicely positioned.*

Proof. Similarly as for line segments (see Lemma 31) the equation that defines the surface for each requirement can be computed in $O(1)$ time after all the tangent points have been computed and for each corner it is checked if it is outside the corresponding chain which takes $O(n \log n)$ time. Computing $R(P)$ for a rectangle $P \in \mathcal{C}$ then takes $O(1)$ time, since the number of surfaces involved is only constant.

As described above computing the intersection of all $R(P)$ can be done by computing the arrangement of surfaces and computing for each cell if it is a subset of $R(P)$ for all $P \in \mathcal{C}$. This can be done in $O(k^{4+\varepsilon})$ for any $\varepsilon > 0$.

Thus the total time required to compute the solution to Minimal Convex Hull is $O(n \log n + k^{4+\varepsilon})$. □



(a) Only the connection between s_l and s_t

(b) Top left part is a single point

(c) Top left part consists of multiple line segments

Figure 5.7: The top left boundary of the convex hull can consist of different line segments

Unfortunately the extreme rectangles are not always nicely positioned, unique or distinct. If for example S_l is not unique because there is more then one square with the leftmost right edge then as was the case with line segments this makes the problem easier. To get a minimal convex hull the lowest leftmost right edge must have a point placed at the top of its edge whereas the highest leftmost right edge must have a point placed on its lower endpoint. The convex hull then only depends on the placement of three points.

It can also be that extreme squares are not distinct, for example S_l and S_t may be from the same square. In that case the top left chain is simply a point and this point can first be considered as being placed on S_l and if that does not lead to a solution it can be placed on S_t. Again the convex hull then only depends on three points.

Lastly it is possible that the extreme rectangles are not nicely positioned. For example R_l may only be partially below R_t as illustrated in Figure 5.7b. In these cases not every choice of points on the extreme edges leads to a minimal convex hull. In Figure 5.7b the convex hull can be made smaller by moving S_l to the left. However it still holds that every minimal convex hull is defined by a choice of points on the extreme edges and the chains.

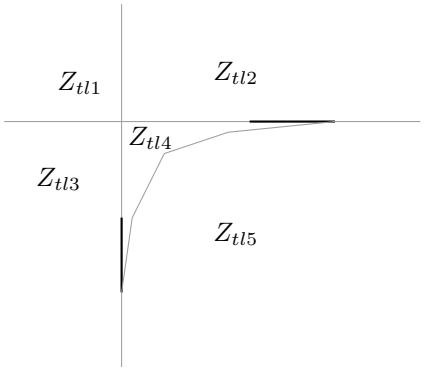


Figure 5.8: Top left corners are divided into sets based on their position

The constraints that we use when the extreme rectangles are nicely positioned heavily rely on the fact that the upper boundary of the convex hull is the connection between s_l and s_t. If the extreme regions are not so nicely positioned the top left part of the convex hull may consist of multiple line segments or of a single point as shown in Figure 5.7b and 5.7c.

To deal with this we look at two different cases, namely if the top left boundary of the convex hull contains part of the top left chain and if it consists of only direct connections between extreme point.

First we assume that the top left boundary is a chain connection. We divide the top left corners of regions in $\mathcal{C}$ into five sets, Z_{tl1} to Z_{tl5}, based on their location as indicated in Figure 5.8. Let S'_l be the vertical line through S_l and S'_t the horizontal line through S_t. Then the points in Z_{tl1} are above or on S'_t and to the left of or on S'_l; points in Z_{tl2} are above S'_t and to the right of S'_l; points in Z_{tl3} are below S'_t and to the left of or on S'_l; points in Z_{tl4} are below S'_t and to the right of S'_l and above the chain, but the vertices of the chain are also in Z_{tl4}; points in Z_{tl5} are below S'_t, to the left of S'_l and below or on the chain, except the vertices of the chain.

Because we assumed that the top left boundary of the convex hull is a chain connection points in Z_{tl1} are clearly outside the top left boundary. Points in Z_{tl5} are never outside the top left boundary. For points in Z_{tl4} we have two of the same constraints we had before, namely tl_1 and tl_2, where l_{ltl} and l_{ttl} are defined in the same way. However as shown in Figure 5.7c it is also possible that a tangent line from the chain to s_r is part of the top left boundary of the convex hull. To ensure that a point p_{tl} is not under this line we also require that s_r is under the line l_{ttl}. Similarly we require that s_b is to the right of the line l_{ltl}. For points in Z_{tl2} and Z_{tl3} we have similar constraints on s_b and s_r. Let l_{rtl} and l_{btl} be lines tangent to the top left chain that cross p_{tl}, and the horizontal line through S_t and the vertical line through S_l respectively. The new constraints are then as follows.

(tl_4) s_b is to the right of where l_{ltl} crosses the horizontal line through S_b
(tl_5) s_r is below where l_{ttl} crosses the vertical line through S_r.
(tl_6) s_b is to the right of where l_{btl} crosses the horizontal line through S_b.
(tl_7) s_r is below where l_{rtl} crosses the vertical line through S_r

Let $Z'_{tl1}(P)$ be $\mathbb{R}^4$ if the top left corner p_{tl} of P is in $Z_{tl1}(P)$ and empty otherwise, $Z'_{tl2}, Z'_{tl3}, Z'_{tl4}$ and Z'_{tl5} are defined similarly. Let tl'_4 to tl'_7 denote the regions of the solution space where tl_4 to tl_7 hold respectively. The region $R_{ctl}(P)$ of the solution space where the top

left corner is outside the top left boundary of the convex hull of $CH(s_l, s_t, s_r, s_b)$ under the assumption that this is a chain connection is defined as follows. Note that $R_{ctr}(P), R_{cbr}(P)$ and $R_{cbl}(P)$ are defined similarly.

$$R_{ctl}(P) = Z'_{tl1} \cup (Z'_{tl2} \cap tl'_7) \cup (Z'_{tl3} \cap tl'_6) \cup (Z'_{tl4} \cap tl'_1 \cap tl'_2 \cap tl'_4 \cap tl'_5)$$

If the top left boundary of the convex hull consists of direct connections between the extreme points another problem arises. In the case of nicely positioned extreme rectangles it is sufficient to say that the top left corner has to be above the line $s_l s_t$. In the more general case however this is not always sufficient since the top left connection may not consist only of the line segment $s_l s_t$. Instead we will replace tl_3 with a set of requirements that hold if and only if a top left corner is outside the convex hull of s_l, s_t, s_r and s_b. If these points occur in clockwise order on the boundary of the convex hull then a top left corner p_{tl} is outside the convex hull if and only if it is to the left of at least one of the vectors $\overrightarrow{s_l s_t}, \overrightarrow{s_t s_r}, \overrightarrow{s_r s_b}$ or $\overrightarrow{s_b s_l}$.

The order of the extreme points on the convex hull of the extreme points is not always the same, but we can treat each ordering as a separate case. The solution space is then split into regions where in each region the ordering is the same. This splitting can also be done using constraints on the extreme points. For four extreme points q_1, q_2, q_3 and q_4 they appear in that order on the boundary of the convex hull if and only if the following requirements hold.

- q_2 is to the left or on of $\overrightarrow{q_1 q_3}$
- q_4 is to the left or on of $\overrightarrow{q_3 q_1}$
- q_1 is to the left or on of $\overrightarrow{q_4 q_2}$
- q_3 is to the left or on of $\overrightarrow{q_2 q_4}$

It is also possible that only three points are vertices on the boundary of the convex hull and the fourth point is in the interior of the convex hull. q_4 is in the interior of the convex hull of q_1, q_2 and q_3 with the points occurring on the boundary in that order if and only if the following requirements hold.

- q_4 is to the right of $\overrightarrow{q_1 q_2}$
- q_4 is to the right of $\overrightarrow{q_2 q_3}$
- q_4 is to the right of $\overrightarrow{q_3 q_1}$

Because for each of these regions of the solution space the ordering of points on the convex hull of extreme points is fixed we can compute where in the solution space a top left corner p_{tl} is outside the convex hull. Let $R_{dtl}(P)$ be the region of the solution space where the constraints hold that guarantee that the top left corner of P is outside the convex hull of extreme points and R_{dtr}, R_{dbr} and R_{dbl} defined similarly for the top right, bottom right and bottom left corner respectively. Then the region $R(P)$ of the solution space where P has some corner outside the convex hull is defined as follows.

$$R(P) = (R_{ctl}(P) \cap R_{dtl}(P)) \cup (R_{ctr}(P) \cap R_{dtr}(P)) \cup (R_{cbr}(P) \cap R_{dbr}(P)) \cup (R_{cbl}(P) \cap R_{dbl}(P))$$

Similar requirements are used for the other corners and they can be turned into regions of the solution space and combined in the same way as for nicely positioned extreme rectangles. There are only a constant number of possible orderings for the extreme points on the convex hull. Hence the running time does not change.

We can then conclude with the following theorem.

Theorem 10. *Given a set $\mathcal{R}$ of n disjoint axis aligned rectangles and a subset $\mathcal{C} \subseteq \mathcal{R}$ containing k rectangles, then* Minimal Convex Hull *on $\mathcal{R}$ and $\mathcal{C}$ can be solved in $O(n^{4+\varepsilon})$, for any $\varepsilon > 0$.*

We then conclude the following corollary from Theorem 10 and Lemma 24.

Corollary 5. *Given a set $\mathcal{R}$ of n disjoint axis aligned rectangles and a subset $\mathcal{C} \subseteq \mathcal{R}$ containing k rectangles, then* Subset Convex Hull *on $\mathcal{R}$ and $\mathcal{C}$ can be solved in $O(n^{4+\varepsilon})$, for any $\varepsilon > 0$.*

5.3 Disjoint unit discs

Although the number of extreme points for a disc is not constant like for squares (corners) or line segments (end points) Lemma 25 also holds for discs.

Although Lemma 25 limits the placements of points that are vertices on the convex hull to the boundary of their region this still leaves infinitely many placements for many points and does not tell us anything about which regions should be considered to contribute vertices on the convex hull.

Although on a set of discs there can be a lot of different possible convex hulls for different placements the convex hull CH_{cp} on the center points of these discs is unique. We can show that only discs whose center point contributes a vertex to CH_{cp} may contribute a vertex to a minimal convex hull on the discs.

Lemma 35. *Only discs whose center point is a vertex on CH_{cp} can contribute a vertex to a minimal convex hull.*

Proof. Let D be a disc that does not contribute a vertex to CH_{cp}. For any two discs P and Q whose center points occur consecutively on CH_{cp} for any placement of points $p \in P$ and $q \in Q$ there is always a part of D that is below the line pq as illustrated in Figure 5.9. Because in a minimal convex hull lines the boundaries do not cross there must be a region of D inside any minimal convex hull, so any region D that does not contribute a vertex to CH_{cp} can not contribute a vertex to a minimal convex hull. □

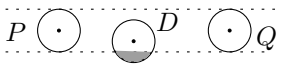
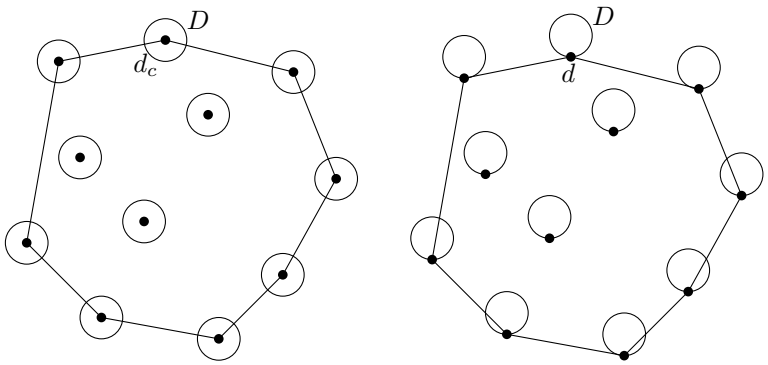


Figure 5.9: Only discs whose center point is a vertex on CH_{cp} can contribute a vertex to a minimal convex hull

The converse of this can also be proven, namely that only discs that can contribute a vertex to a minimal convex hull can contribute a vertex to CH_{cp}.

Lemma 36. *For every disc D that contributes a vertex to CH_{cp} there is minimal convex hull to which D contributes a vertex.*

Proof. Let D be a disc that contributes a vertex to CH_{cp}. Then without loss of generality assume that the center point d_c of D is the unique topmost point of CH_{cp}. All other points of CH_{cp} have a smaller y-coordinate then d_c.



(a) CH_{cp} (b) CH

Figure 5.10: Different convex hulls to illustrate that any region that contributes a vertex to the convex hull of center points also contribute a vertex to some minimal convex hull.

Next we define CH to be the convex hull which is the same as CH_{cp} but moved down by 1 as illustrated in Figure 5.10. CH still contains at least one point of each region so it is a valid convex hull. Also it intersects only in a single point d with D. Although CH does not have to be a minimal convex hull by definition there is also a minimal convex hull CH_{min} which in contained in CH. Then because every point in CH has a lower y-coordinate then d also every point in CH_{min} has a lower y-coordinate then d. It then follows that d is a vertex of CH_{min}.

□

Chapter 6

Conclusion

We have shown that it is NP-hard to determine if a pair of imprecise points is a possible closest pair when the input is a set of disjoint parallel line segments of unit length, a set of disjoint axis aligned squares of unit size or a set of disjoint unit discs. Most other types of regions are generalizations of these types. Hence the possible closest pair problem is NP-hard for the most commonly used types or regions. A question that is still open is whether the possible closest pair problem is also NP-complete.

For the convex hull on imprecise points we defined three different structures, the exact, superset and subset convex hull. We have shown that the exact convex hull problem is NP-hard when regions are line segments. It would be interesting to see if this is also the case for disjoint parallel line segments of unit length, since both the subset and superset convex hull problems can be solved in polynomial time for these regions.

We have given two results on the superset convex hull problem. For arbitrarily line segments it is NP-hard, whereas for disjoint parallel line segments of unit length we can solve it in polynomial time. This leaves a very interesting gap of which combination of restrictions on the line segments enables the polynomial solution.

The subset convex hull problems seems somewhat easier than the exact and superset convex hull. We have given polynomial time algorithms that solve the subset convex hull problem for sets of disjoint parallel line segments and for sets of disjoint axis aligned rectangles. There are many open questions for the subset convex hull. Most notably if the subset convex hull problem is NP-hard for arbitrary line segments as is the case for the exact and superset convex hull, and whether there is a polynomial algorithm that can solve the subset convex hull problem when the regions are disjoint unit discs.

Bibliography

[BCKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[Cab07] Sergio Cabello. Approximation algorithms for spreading points. *Journal of Algorithms*, 62(2):49–73, 2007.

[CEGS91] Bernard Chazelle, Herbert Edelsbrunner, Leonidas Guibas, and Micha Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theoretical Computer Science*, 84(1):77–105, 1991.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* The MIT Press, 2001.

[EG89] Herbert Edelsbrunner and Leonidas Guibas. Topologically sweeping an arrangement. *Journal of Computer and System Sciences*, 38(1):165–194, 1989.

[FKP02] Jirí Fiala, Jan Kratochvíl, and Andrzej Proskurowski. Geometric systems of disjoint representatives. In *GD '02: Revised Papers from the 10th International Symposium on Graph Drawing*, pages 110–117, London, UK, 2002. Springer-Verlag.

[GSS90] Leonidas Guibas, David Salesin, and Jorge Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. In *SIGAL '90: Proceedings of the international symposium on Algorithms*, pages 261–270, New York, NY, USA, 1990. Springer-Verlag.

[Kol04] Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *Journal of the ACM*, 51(5):699–730, 2004.

[LK07] Maarten Löffler and Marc van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. In Frank K. H. A. Dehne, Jrg-Rdiger Sack, and Norbert Zeh, editors, *WADS*, volume 4619 of *Lecture Notes in Computer Science*, pages 446–457. Springer, 2007.

[LK08] Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 2008.

[MKGB08] Asish Mukhopadhyay, Chanchal Kumar, Eugene Greene, and Binay Bhattacharya. On intersecting a set of parallel line segments with a convex polygon of minimum area. *Information Processing Letters*, 105(2):58–64, 2008.

[NT01] Takayuki Nagai and Nobuki Tokura. Tight error bounds of geometric problems on convex objects with imprecise coordinates. In *JCDCG '00: Revised Papers from the Japanese Conference on Discrete and Computational Geometry*, pages 252–263, London, UK, 2001. Springer-Verlag.

[SSG89] David Salesin, Jorge Stolfi, and Leonidas Guibas. Epsilon geometry: building robust algorithms from imprecise computations. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry*, pages 208–217, New York, NY, USA, 1989. ACM.