Abstract. We propose to study combinatorial and algorithmic aspects of geometric separation problems in the plane. In such a situation one is given a set of points, line segments or polygons in the plane and a set of separators such as lines, line segments, disks or polygons and the goal is to select a small subset of those separators such that every path between any two objects is intersected by at least one separator. We first look at several problems which arise when one is given a set of points and a set of unit disks in the plane and the goal is to separate the points using the minimum number of unit disks. We then focus on a separation problem involving only two points: Given a region bounded by a piecewise linear closed border, such as a fence, place few guards inside the fenced region such that wherever an intruder cuts through the fence, the closest guard is at most a distance one away. Lastly we restrict the separating objects to be lines and focus on algorithmic and combinatorial aspects which arise when we use them to pairwise separate a set of points, line segments or polygons in the plane.
4.1. Separating families on Arbitrary Sets 23
4.2. Separating Polygons using Lines 24
4.3. Separating Line Segments using Lines 29
References 31
1. Introduction and Organization

The main focus of this proposal are geometric separation problems in the plane. In such a situation one is given a set of points, line segments or polygons in the plane and a set of separators such as lines, line segments, disks or polygons, the goal is to select a small subset of those separators such that every path between any two objects is intersected by at least one separator.

The main motivation for studying these types of problems comes from sensor networks. Historically sensor networks tried to achieve full coverage of a region, i.e., each point in the region has to be within the sensing radius of at least one sensor. But recently, wireless sensors are being extensively used in applications to provide barriers as a defense mechanism against intruders at important buildings, estates, national borders etc. Monitoring the area of interest by this type of coverage is called barrier coverage. Such sensors are also being used to detect and track moving objects such as animals in national parks, enemies in a battlefield, forest fires, crop diseases etc. In such applications it might be prohibitively expensive to attain blanket coverage but sufficient to ensure that the object under consideration cannot travel too far before it is detected.

In Section 2 we outline several problems which arise when one is given a set of points and a set of unit disks in the plane and the goal is to separate the points using the minimum number of unit disks. We are interested in the approximation ratio of a simple recursive algorithm as well as the computational complexity of the problem. We hope that by investigating the complexity of the problem we can also settle computational hardness questions of related disk separation problems. We furthermore survey related research in the setting of line segments (instead of disks).

In Section 3 we propose to investigate a separation problem in the plane involving only two points: Given a region bounded by a piecewise linear closed border, such as a fence, place few guards inside the fenced region such that wherever an intruder cuts through the
fence, the closest guard is at most a distance one away. We present a preliminary result, namely an algorithm which finds the asymptotically optimal number of guards in case where the perimeter of the region is much larger than the number of border edges.

In Section 4 we restrict the separating objects to be lines and focus on algorithmic and combinatorial aspects which arise when we use them to pairwise separate a set of points, line segments or polygons in the plane. We proceed by first reviewing existing research in the algorithmic setting such as algorithms and computational complexity results for separating \( n \) points in the plane using the minimum or minimal number of lines. We then move to the combinatorial setting and survey existing work on the number of ways to partition \( n \) points in \( \mathbb{R}^d \) into two, possibly empty, sets using \( d - 1 \) dimensional hyperplanes. We propose to extend this line of research by enumerating the number of ways \( n \) points in the plane can be pairwise separated using \( k \) lines.

2. Separating Points using Disks and Line-Segments

Wireless sensors are being extensively used in applications to provide barriers as a defense mechanism against intruders at important buildings, estates, national borders etc. Monitoring the area of interest by this type of coverage is called barrier coverage [38]. Such sensors are also being used to detect and track moving objects such as animals in national parks, enemies in a battlefield, forest fires, crop diseases etc. In such applications it might be prohibitively expensive to attain full coverage but sufficient to ensure that the object under consideration cannot travel too far before it is detected. Such coverage is called barrier coverage [5, 49]. Inspired by such applications, we want to study the problem of isolating a set of points by a minimum-size subset of a given set of unit radius disks. A unit disk crudely models the region sensed by a sensor, and the work reported here readily generalizes to disks of arbitrary, different radii.

Problem 2.1 (Point Isolation). Given a set \( \mathcal{D} \) of \( n \) unit disks, and a set \( P \) of \( k \) points such that \( \mathcal{D} \) separates \( P \), that is, for any two points \( p, q \in P \), every path between \( p \) and \( q \)
intersects at least one disk in $\mathcal{D}$. The goal is to find a minimum cardinality subset of $\mathcal{D}$ that separates $P$. See Figure 1 for an illustration of this notion of separation.

There has been a lot of recent interest on geometric variants of well-known \textbf{NP}-hard combinatorial optimization problems and we would like to extend this line of research by investigating the Point Isolation Problem. For several variants of the geometric set cover problem, approximation algorithms have been designed [17, 3, 41] that improve upon the best guarantees for the combinatorial set cover problem. For the problem of covering points by the smallest subset of a given set of unit disks, there exist approximation algorithms that guarantee an $O(1)$-approximation and even a PTAS [11, 41]. These results hold even for disks of arbitrary radii. The Point Isolation Problem can be viewed as a set cover problem.
where the elements that need to be covered are not points, but paths. However, known
results only imply a trivial $O(n)$-approximation when viewed through this set cover lens.

2.1. Related Work in the Setting of Line Segments. A Point Isolation variant is
studied in [2] where one is given a set $D$ of line segments in the plane and two points $a$ and
$b$ in different cells of the induced arrangement and the authors investigate the following 3
problems:

**Problem 2.2.** *(2-Cells-Connection Problem)* Compute the minimum number of segments
one needs to remove so that there is a path connecting $a$ and $b$ that does not intersect any of
the remaining segments.

**Problem 2.3.** *(All-Cells-Connection Problem)* Compute the minimum number of segments
one needs to remove so that the arrangement induced by the remaining segments has a single
cell.

**Problem 2.4.** *(2-Cells-Separation Problem)* Compute the minimum number of segments one
needs to retain so that any path connecting $a$ to $b$ intersects some of the retained segments.

The study of those problems is motivated by sensor network questions. If each line segment
corresponds to a sensor Problem 2.2 asks for the minimum number of sensors that must be
turned off so that an intruder can walk freely from $a$ to $b$ while Problem 2.3 generalizes this
question by asking for the minimum number of sensors which must be turned off so that an
intruder can walk freely between any points contained in the faces of the arrangement. In
the same light, Problem 2.4 asks for the minimum number of sensors to be turned on so that
any intruder walking from $a$ to $b$ gets registered.

Problem 2.2 is shown to be NP-hard by a reduction from MAX-2-SAT. For the reduction,
the authors distinguish between heavy and light line segments. Given a 2-CNF formula $F$
consisting of $m$ clauses, heavy line segments are $cm$ copies of a single line segment, while
light line segments are just single line segments. The factor $c$ is chosen such that an optimal
solution for Problem 2.2 never removes a heavy segment. The high-level idea of the reduction
uses curved instead of straight segments. As shown in Figure 3, given a CNF formula $F$, a rectangle $\mathcal{D}$ is built out of heavy segments which contains a point $a$ in the lower and a point $b$ in the upper corner. For each variable $x_i$ of $F$ a short vertical heavy segment is added to the lower half of $\mathcal{D}$. From this segment $k_i$ horizontal light segments, denoted by $R_i$, go to the right and $k_i$ horizontal light segments, denoted by $L_i$ go to the left, where $k_i$ corresponds to the number of occurrences of $x_i$ in $F$. The idea is that an optimal $a$-$b$ path has to choose for each $x_i$ whether it crosses all segments in $L_i$. Thus, setting $x_i$ to true, or all segments in $R_i$, thereby setting $x_i$ to false. For a clause consisting of two literals $l_i$ and $l_j$, depending on whether $l_i$, $l_j$ are positive or negative literals respectively, one of the curved segments in $L_i/R_i$ and one segment in $L_j/R_j$ get prolonged such that they cross inside the upper part of the rectangle $\mathcal{D}$ as shown in Figure 3. The prolongation is done in such a way that an optimal $a$-$b$ path crosses exactly one of the prolongations for each clause. Thus, any optimal $a$-$b$ path crosses $2m$ line segments in the lower part of $\mathcal{D}$ thereby encoding a variable assignment which satisfies $\text{opt}(F)$ many clauses. Therefore, in the upper part of $\mathcal{D}$ only $m - \text{opt}(F)$ additional segment crossings are counted, since $\text{opt}(F)$ of the $m$ segments in the upper part were already crossed in the lower part of $\mathcal{D}$.

Using a more careful construction, one can replace the curved segments by straight line segments and NP-hardness of Problem 2.2 thus follows.

Figure 3. Example reduction from MAX-2-SAT to Problem 2.2.
Since the reduction is approximation preserving and MAX-2-SAT is APX-hard this implies that Problem 2.2 is also APX-hard. A problem is APX-hard if the existence of a PTAS for the problem implies the existence of a PTAS for all problems in the APX class, i.e. for the class for which polynomial time constant factor approximation algorithms exist. Note that Arora et al. showed in [4] that this would imply \( P = NP \).

In [53] Problem 2.2 is carefully shown to be NP-hard even when all the line segments are restricted to have unit length but this reduction is not approximation preserving.

Problem 2.3 is shown to be NP-hard by a reduction from the feedback vertex set problem (FVS), where one is looking for the smallest subset of vertices of a given graph whose removal leaves the graph without cycles. As shown in [54], this problem is NP-hard even on planar graphs. In order to reduce the planar FVS problem to Problem 2.3, the edges of a planar graph \( G \) are subdivided so that the resulting graph \( G' \) is bipartite (and planar). Observe that the optimal solution of the FVS problem on \( G \) and \( G' \) have the same size. Furthermore, it is well known that any planar bipartite graph is the intersection graph of horizontal and vertical segments, where no two segments intersect at an interior point. Thus, for a set \( D \) of line segments whose corresponding intersection graph is \( G' \), removing \( k \) segments of \( D \) connects all cells in the arrangement of \( D \) if and only if \( G \) has a feedback vertex set of size \( k \).

For Problem 2.4 an \( O(n^2 + nk) \) algorithm is presented where \( k \) is the number of pairs of segments that intersect. For a set of line segments \( D \) the algorithm first builds the intersection graph \( G = (D, E) \), with \( E = \{ \{D, D'\} : D \cap D' \neq \emptyset, D, D' \in D \} \). Without loss of generality assume that the line segment \( \overline{ab} \) connecting the points \( a \) and \( b \) is vertical. For a polygonal path \( \gamma \), \( N(\gamma; a, b) \) denotes the number of times \( \gamma \) intersects \( \overline{ab} \) from left to right minus the number of times \( \gamma \) intersects \( \overline{ab} \) from right to left.

The algorithm computes a shortest path tree \( T_v \) for each \( v \in D \). For a spanning tree \( T \) and an edge \( e \), \( \tau(e, T) \) denotes the cycle obtained by concatenating the edge \( e \) with the path in \( T \) connecting both endpoints of \( e \) and \( \gamma(e, T) \) denotes the corresponding polygonal curve.
Figure 4. The set of line segments (left), with its corresponding intersection graph (middle) and a closed polygonal path $\gamma(\pi)$ for the closed walk $\pi = s_2s_1s_4s_6s_7s_2$ (right).

Computing the set

$$P = \{(r, e) \in D \times E(G) : e \in E(G) \setminus E(T_r) \land N(\gamma(e, T_r); a, b) \neq 0\}$$

and choosing

$$(r^*, e^*) = \arg \min_{(r, e) \in P} |\tau(e, T_r)|,$$

the segments in $\tau(e, T_r)$ are returned by the algorithm. Letting $k$ denote the number of line segment intersections (i.e., $k = |E(G)|$), the algorithm can be implemented in a straightforward way in time $O(kn^2)$. A more careful construction yields an $O(nk + n^2)$ time algorithm.

A related problem is that one wants to build a set of line segments (from scratch) which separates two given regions in a polygonal environment and has minimal total length. This has been shown in [35] to be solvable in polynomial time.

2.2. Other Related Work. Sankararaman et al. [49] investigate a notion of coverage which they call weak coverage. Given a region $\mathcal{R}$ of interest (which they take to be a square in the plane) and a set $\mathcal{D}$ of unit disks (sensors), the region is said to be $k$-weakly covered if each connected component of $\mathcal{R} - \bigcup_{D \in \mathcal{D}} D$ has diameter at most $k$. They consider the situation when a given set $\mathcal{D}$ of unit disks completely covers $\mathcal{R}$, and address the problem of partitioning $\mathcal{D}$ into as many subsets as possible so that $\mathcal{R}$ is $k$-weakly covered by every subset.
The work of [2] was extended in [12] to include an exact algorithm for solving the two-point separation problem on unit disks. Furthermore, they show that separating $k$ points with the minimum number of unit circles is $\textbf{NP}$-hard, but they do not investigate the situation where unit disks are given, i.e., they don’t investigate the complexity of the Point Isolation problem.

Berg and Kirkpatrick [6] consider a problem that loosely resembles the two-point separation problem of [12]: Given a set of unit disks and two points $s$ and $t$, find a path from $s$ to $t$ that intersects the smallest number of disks.

2.3. An Approximation Algorithm for the Point Isolation Problem. In [26] a polynomial time $O(1)$-approximation algorithm for the Point Isolation problem is presented. The algorithm works by calling recSep($P$), where recSep($Q$), for any $Q \subseteq P$ is the following recursive procedure, illustrated in Figure 5:

1. If $|Q| \leq 1$, return $\emptyset$.
2. For every pair of points $s, t \in Q$, invoke the algorithm of [12] to find a minimum cardinality subset $B_{s,t} \subseteq D$ such that $B_{s,t}$ separates $s$ and $t$.
3. Let $B$ denote the minimum size subset $B_{s,t}$ over all pairs $s$ and $t$ considered.
4. Let $Q_1$ and $Q_2$ be the partition of $Q$ into two subsets such that each subset corresponds to points in the same face induced by $B$. It is easy to see that $B$ consists of exactly one bounded region and thus it indeed partitions $Q$ into exactly two blocks.
5. Return $B \cup \text{recSep}(Q_1) \cup \text{recSep}(Q_2)$.

**Problem to Investigate 1.** What is the approximation factor of the above algorithm?

**Problem to Investigate 2.** Is it possible to obtain a $\textbf{PTAS}$ for the Point Isolation problem?

2.4. Complexity of Point Isolation and Related Problems. Besides the algorithmic aspects of the Point Isolation problem we would also like to investigate its Computational Complexity.

**Problem to Investigate 3.** Is the Point Isolation Problem $\textbf{NP}$-complete?
We would furthermore like to investigate the following related problem when restricted to unit disk graphs (see Definition 1).

**Problem 2.5 (Multiterminal Cut Problem).** Given a graph $G = (V, E)$ and a set $S \subseteq V$ of $k$ terminals, find the minimum cardinality set $E' \subseteq E$ such that in $G' = (V, E \setminus E')$ there is no path between any two nodes in $S$, see Figure 6 for an example.

In [18] it was shown that the Multiterminal Cut Problem is NP-hard on planar graphs if $k$ is not fixed. Furthermore, it is shown to be MAXSNP-hard for any fixed $k \geq 3$. This implies that it is APX-hard and thus no PTAS exists unless $P = NP$.

On the other hand, when $S$ only contains two vertices this problem is known as the min-cut (max-flow) problem for which many efficient algorithms are known.

An easy 2-approximation algorithm for Problem 2.5 is presented in [18] which greedily chooses a minimum cut $C_i$ separating the terminal $s_i$ from all other terminals in $S$, which takes one max-flow computation. Denoting by $A$ an optimal solution for Problem 2.5 and by $A_i$ a cut in $A$ separating $s_i$ from $S \setminus \{s_i\}$, it follows that each edge in $A$ is contained in two of the cuts $A_i$ and $A_j$ for some $1 \leq i < j \leq k$. This holds since in any optimal solution there are exactly $k$ connected components and each edge of $A$ is incident to two of the cut
components. Thus,

$$\sum_{i=1}^{k} w(A_i) = 2w(A)$$

and since $w(C_i) \leq w(A_i)$ the 2-approximation of the greedy algorithms follows. Discarding the heaviest of the $k$ cuts improves the approximation factor to $2 - 2/k$, since

$$w(C) \leq (1 - 1/k) \sum_{i=1}^{k} w(C_i) \leq (1 - 1/k) \sum_{i=1}^{k} w(A_i) = 2(1 - 1/k)w(A).$$

\textbf{Definition 1.} A unit disk graph is the intersection graph of a family of unit disks in the Euclidean plane. Each disk is represented as a vertex and two vertices are connected by an edge if and only if the corresponding disks intersect (see Figure 7 for an example.)
Figure 8. An example of the All-Cells Separation question in the setting of Unit Disks. Left: A set $\mathcal{D}$ of unit disks, right, a maximum cardinality subset $\mathcal{D}' \subseteq \mathcal{D}$ of the disks such that $\mathcal{D}'$ induce an arrangement consisting of only a single cell.

Problem to Investigate 4. Is the Multiterminal Cut Problem $\text{NP}$-hard on Unit Disk Graphs?

Problem to Investigate 5. How well can the Multiterminal Cut Problem be approximated on Unit Disk Graphs?

Moving back to the problems studied in [2], we are interested in investigating the All-Cells Separation problem (Problem 2.3) in the setting of Unit Disks as shown in Figure 8 and ask the following questions:

Problem to Investigate 6. Given a set of unit disks embedded in the plane, find a minimum cardinality subset s.t. the remaining disks induce an arrangement consisting of only a single cell. Is this problem $\text{NP}$-hard?

3. Placing Guards inside a Fenced Region

In this section we propose to investigate a separating problem involving only two points: Given a region bounded by a piecewise linear closed border, such as a fence, place few guards inside the fenced region such that wherever an intruder cuts through the fence, the closest guard is at most a distance one away. We present a preliminary result, namely an algorithm which finds an asymptotically optimal number of guards in case where the perimeter of the
Figure 9. A polygon (dotted) containing a geodesic disk centered at $v$, whose interior is depicted in gray and its boundary is drawn in black.

region is much larger than the number of border edges.

We motivate these studies in the context of Barrier Coverage (see for example [7],[14],[15],[37],[39],[48],[50]). In a typical Barrier Coverage problem the goal is to place few sensors or guards to detect any intruder into a given region. We would like to extend this line of research by investigating the boundary guarding problem, which we now define more formally:

For two points $u$ and $v$ in a simple polygon $P$, the geodesic distance, denoted by $d(u,v)$, is the length of the shortest path between $u$ and $v$ inside $P$. A geodesic disk $D$ of radius $r$ centered at a point $v \in P$ is the set of all points in $P$ whose geodesic distance to $v$ is at most $r$. We refer to geodesic disks of radius 1 as geodesic unit disks. The boundary of $D$, denoted by $\partial D$, contains all points of $P$ which are either exactly at distance $r$ from $v$ or which are at distance at most $r$ from $v$ but contained on the polygon boundary $\partial P$ (see Fig. 9).

Having introduced the concept of a Geodesic Unit disk, the Boundary Coverage Problem can be reformulated as:

**Problem to Investigate 7** (Geodesic Boundary Coverage). Given a simple polygon, cover its boundary using the minimum number of Geodesic Unit disks.

A collection of geodesic disks covers a polygon boundary $\partial P$, if each point of $\partial P$ is contained in at least one disk. We would like to consider the setting where the centers of
the disks can be placed anywhere inside the polygon. A version which might be of separate interest is the restriction that the centers have to lie on $\partial P$.

**Problem to Investigate 8** (Geodesic Boundary Coverage). What is the Computational Complexity of the Boundary Coverage Problem in Simple Polygons?

On the other hand, it follows from Theorem 7 of [55] that this problem is $\text{NP}$-hard in polygons with holes.

3.1. **Related Work.** Several papers ([25],[31],[34],[36],[51],[56]) study full coverage of geometric regions with Euclidean disks. For an overview of optimal coverings of squares and triangles with few disks see Chapter 1.7 of [9].

In the context of Barrier Coverage, [12] computes an exact minimal barrier consisting of Euclidean unit disks which separates two points in the plane. Extending the problem to $k$ points, an $O(1)$-approximation algorithm was presented in [26] and $\text{NP}$-hardness was shown in [45]. The same separation problem but using segments instead of disks was addressed in [2].

Covering a simple polygon with the smallest geodesic disk has been studied in [46] and an output sensitive algorithm for computing an arrangement of geodesic disks is presented in [8].

3.2. **Two Greedy Algorithms.** One natural algorithm is to contiguously cover the polygon boundary, in each step extending the currently covered portion maximally but contiguously. This algorithm would result in at most a 2-approximation even for convex polygons. This can be seen by a rectangle of length $n$ and height $\epsilon > 0$. It can be covered with $n/(2\sqrt{1 - \epsilon^2}/4)$ many geodesic unit disks (by centering them on the median line at height $\epsilon/2$). On the other hand, CONTIGUOUSGREEDY centers disks in steps of 2 on the boundary, thus after finishing one side of the rectangle, each disk introduced a small uncovered hole on the other side. CONTIGUOUSGREEDY covers those holes by placing another $n/2$ disks contiguously.
on the other side of the polygon, resulting in a total of $n$ disks needed; see Figure 10 for an illustration of it.

Another natural greedy approach is to cover the largest amount of uncovered boundary at each step. This algorithm results in an approximation ratio of $\Omega(\log n)$, i.e., it is unbounded w.r.t. $|OPT|$. An example where this greedy rule performs badly is illustrated in Fig. 11(a). The parts of the boundary denoted by $F_1, \ldots, F_k$ are dense folding as shown in Fig. 11(b) where the boundary length of $F_1$ is twice that of $F_2$, four times that of $F_3$, and so on. The global greedy algorithm first covers the two $F_1$ sections on opposite sides of the boundary (illustrated by $D_1$ in Fig. 11(a)), then the two $F_2$ sections continuing in this way until the two $F_k$ sections are covered, thereby having used $k$ disks to cover the foldings, (plus some constant number of disks to cover the rest of $\partial P$). Notice that when the height of the polygon is arbitrary close to 2, the number of foldings can be made arbitrary large, while $OPT$ only uses a constant number of disks to cover $\partial P$. 

![Figure 10. Illustration of a convex polygon for which a contiguous greedy algorithm yields at most a 2-approximation. The optimal solution, shown at the top, uses four disks while the approximate solution uses seven disks. This ratio can be made arbitrarily close to 2 by making the rectangle sufficiently long and narrow.](image-url)
Neither of the two greedy algorithms (nor any combination of them) result in a constant factor approximation when the polygon is allowed to have holes.

3.3. Covering Large Perimeters. If the polygon perimeter $L$ is significantly larger than $n$, i.e., $L \geq n^{1+\delta}$, with $\delta > 0$, the problem becomes easier and we propose a simple linear time algorithm which achieves an approximation ratio which goes to one as $L/n$ goes to infinity. For this, we decompose $\partial P$ into long and short portions, based on the length of the corresponding medial axis. The medial axis is the set of points in $P$ which have more than one closest point on $\partial P$. It forms a tree whose edges are either line segments or parabolic arcs and it can be computed in linear time [16]. For a line segment edge, the closest points to the boundary are a subset of two polygon edges; for a parabolic edge, the closest boundary points are a polygon vertex and a subset of a polygon edge. The idea of the algorithm is to identify long edges of the medial axis (of length at least some constant $c > 2$), and to cover the corresponding polygon boundary section (referred to as corridors) almost optimally using only a constant number of disks more than $OPT$ uses to cover the corridor. It is easy to see that each corridor stemming from a parabolic arc can be covered with at most two more disks than $OPT$ uses, by centering disks at distance 2 from each other on the corresponding polygon boundary segment and one disk on the corresponding polygon.
vertex. Each corridor consisting of a pair of polygon boundary segments can be covered by greedily centering disks on the corresponding medial axis as long as each disk contains corridor portions of length more than two; if the length becomes two or less, greedily center the disks on corridor segments in steps of two. Observe that also in this case, the number of disks needed to cover a corridor is at most two more than $OPT$ uses and their centers can be computed in time linear in their number. This holds since there is at most one point where the covering changes from centering disks on the medial axis to centering disks on $\partial P$. The rest of the polygon, i.e., the short portions, can be covered greedily by centering $O(n)$ disks on $\partial P$.

Let $D$ be the set of all disks placed by the algorithm, $D_L \subseteq D$ the disks covering the corridors and $D_S \subseteq D$ the $O(n)$ disks covering the short portion of $\partial P$. Since the number of edges in the medial axis is $O(n)$ (see [16]) and the procedure for covering the long corridors uses at most two more disks than $OPT$ for each corridor, $|D_L| \leq |OPT| + O(n)$. It therefore holds that $|D| = |D_L| + |D_S| \leq |OPT| + O(n)$. Next, we are going to show that $|OPT| = \Omega(L)$. For this we write $|OPT| = |D| - O(n)$. Since any disk in $D_L$ covers corridor boundary lengths of at most 4 and $L \geq n^{1+\delta}$ this implies that $|D| = \Omega(L)$. Furthermore, it is easy to see that the disks of $OPT$ which contain a polygon vertex cover at most an $O(n)$ portion of $\partial P$ implying that $|OPT| = \Omega(L)$. Therefore, the approximation ratio can be written as

$$\frac{|D|}{|OPT|} \leq 1 + \frac{O(n)}{|OPT|} = 1 + \frac{O(n)}{\Omega(L)} = 1 + O\left(\frac{n}{L}\right) = 1 + O\left(n^{-\delta}\right),$$

which indeed goes to one as $n$ goes to infinity.

3.4. Covering with Euclidean Disks. Moving away from the geodesic metric, one can ask the same covering questions for Euclidean disks:

**Problem to Investigate 9** (Simple Euclidean Boundary Coverage). Given a simple polygon, cover its boundary using the minimum number of Euclidean Unit disks.
Problem to Investigate 10 (Euclidean Boundary Coverage). Given a polygon, possibly with holes, cover its boundary using the minimum number of Euclidean Unit disks.

It seems that the fact that the boundary line segments stem from a polygon does not provide much information to the Euclidean disk and thus a related, more general problem to look at is the following:

Problem to Investigate 11 (Line Segment Coverage). Given a set of line segments cover them using the minimum number of Euclidean Unit disks.

As a preliminary result we have

Theorem 3.1. One can compute in $O(n)$ time a 7-approximation for the number of Euclidean unit disks needed to cover a set $S$ of $n$ line segments.

To see this, we define $S_L \subseteq S$ to consist of long segments, having length at least some constant $L$. For each such segment we construct a hippodrome of width 2 and cover it using $4L$ unit disks.

To cover the short segments contained in $S_S = S \setminus S_L$, we construct a $\sqrt{2}$-square grid and we observe that a unit disk centered in the middle of the square fully covers it. Furthermore, it is easy to see that any unit disk intersects at most 7 grid cells. Thus simply counting the number of cells intersected by elements of $S_S$ yields a 7-approximation.

Problem to Investigate 12 (Point Coverage). Given a set of points in the plane, cover them using the minimum number of Euclidean Unit disks.

In [27] a $2^{d-1}[\sqrt{d}]^d$ approximation algorithm for covering $n$ points in $\mathbb{R}^d$ using Euclidean unit balls is presented which runs in time $O(dn + n \log s)$, with $s$ the number of hyper-squares in an optimal solution. A PTAS was presented in [30] but its running time of $n^{O(\epsilon^{-2})}$ renders the algorithm impractical even for relatively large values of $\epsilon$. 
We would like to tackle the Point Coverage Problem using a local search algorithm. We are especially interested in investigating a question involving the following local optimality definition:

**Definition 2** \((k\text{-optimal})\). A set \(D\) of unit disks is \(k\text{-optimal}\), if no \(k\)-disks can be replaced by \(k-1\) disks and still cover the points set \(P\).

**Problem to Investigate 13** (Local Search for Point Cover). If a set of disks is \(k\)-optimal for a set \(P\) of points, what can be said about the number of disks w.r.t. \(OPT\)?

It is easy to see that 2-optimality does not provide any bound on \(D/OPT\).

**Problem to Investigate 14** (3-Optimality). If a set of disks is 3-optimal for a set \(P\) of points, what can be said about the number of disks. w.r.t. \(OPT\)?

4. **Separating Objects with Lines**

Pairwise separating, i.e., *shattering*, objects in the plane using lines in such a way that each object is contained in its own cell is well studied in the computational geometry community because of its potential applications to manufacturing, constructive solid geometry and statistical classification (see [24]). Several papers ([13],[24],[43]) focus on algorithmic and complexity theoretical aspects of separating points in the plane with lines. In [24] it is shown that shattering \(n\) points in \(\mathbb{R}^d\) using \(n-1\) parallel hyperplanes can be done in time \(O(dn \log n)\) by first computing the normal vector \(\vec{n}\) of the \(d-1\) dimensional hyperplane to which all points uniquely project. The separating hyperplanes can then be found by sweeping the line containing \(\vec{n}\).

If we don’t restrict ourselves to parallel hyperplanes, then the problem of finding the minimum number of hyperplanes which shatter \(n\) points is \(NP\)-hard, even in \(\mathbb{R}^2\). This is shown in [24] by a reduction from the following \(NP\)-hard problem ([40]): Given a set of rational points in \(\mathbb{R}^2\) and an integer \(k\), is there a set of \(k\) straight lines, such that each point lies on at least one line?
In [24] it is further shown that shattering $n$ points in the plane with the minimum number of lines remains $NP$-complete when the lines are restricted to be either horizontal or vertical by a reduction from 3-SAT.

The study of combinatorial aspects was initiated in [29] where Hardig showed that there are

$$\sum_{i=0}^{d} \binom{n-1}{i}$$

ways to partition $n$ points in $\mathbb{R}^d$ into two (possibly empty) sets using a $d - 1$ dimensional hyperplane, which for $d = 2$ evaluates to $\binom{n}{2} + 1$. Furthermore, in [22], the minimum number of hyperplanes required to pairwise separate $n$ points is studied in both the general and convex cases.

We would like to continue this line of research by investigating the number of ways to separate $n$ points in convex position in the plane with $k$ lines.

Since there are generally uncountably many lines introducing the same separation, we define the following notion of linear bipartitions. For a set $S$ of points in the plane, a linear bipartition of $S$ is a set \{U, S \setminus U\} consisting of two disjoint nonempty subsets of $S$ which respectively are fully contained in the two open half-planes bounded by some line.

A set $\mathcal{P}$ of linear bipartitions is called a linear separating family for $S$ if for every distinct elements $p, q \in S$ there is a linear bipartition \{U, S \setminus U\} in $\mathcal{P}$ such that $p \in U$ and $q \in S \setminus U$. Furthermore, $\mathcal{P}$ is called minimal, if no proper subfamily of $\mathcal{P}$ separates $S$.

**Example 1.** Given a set $S = \{p_1, \ldots, p_6\} \subseteq \mathbb{R}^2$ of points in convex position, the set $\mathcal{P} = \{P_1, \ldots, P_4\}$ defined below and illustrated in Figure 12 is an example of a minimal
linear separating family for $S$.

\[
P_1 = \{(p_1, p_2, p_3, p_4), \{p_5, p_6\}\}, \quad P_2 = \{(p_1, p_2, p_3, p_6), \{p_4, p_5\}\}
\]

\[
P_3 = \{(p_1, p_3, p_4, p_5, p_6), \{p_2\}\}, \quad P_4 = \{(p_1, p_4, p_5, p_6), \{p_2, p_3\}\}
\]

In [24] a tight lower bound of $\Omega(n^{1/d})$ on the number of hyperplane needed to shatter a point set in $\mathbb{R}^d$ is given. This lower bound can be seen, since an arrangement of $r$ hyperplanes in $\mathbb{R}^d$ can have at most $\sum_{i=0}^{r} \binom{d}{i}$ cells (see [20]) and the claimed bound follows by solving it for $r$. Tightness follows since one can place one point in each cell of the hyperplane arrangement. For $d \leq 4$ exact values were computed, but no closed form solution exists for $d \geq 5$, since this requires finding the root of polynomials of degree bigger than 4. Considering an upper bound, $n - 1$ hyperplanes of dimension $d - 1$ always suffice to shatter a set of point in $\mathbb{R}^d$ as can be seen by induction on $n$. Given an $n + 1$ point set $S$ and a set $\mathcal{D}$ of hyperplanes shattering $S$ we can look at a hyperplane $h \in \mathcal{D}$ for which not all the points lie on one side of $h$. Thus, $h$ separates $S$ into $S_1$ of size $n_1$ and $S_2$ of size $n_2$, with $n_1 + n_2 = n + 1$. Furthermore, $\mathcal{D}$ clearly shatters both $S_1$ and $S_2$. By the induction hypothesis $S_1$ can be shattered by $n_1 - 1$ hyperplanes and $S_2$ by $n_2 - 1$ hyperplanes. Thus, $S$ can be shattered by $n_1 - 1 + n_2 - 1 + 1 = n$ hyperplanes and the claim follows.

If the $n$ points in $S$ are in convex position in $\mathbb{R}^d$, it is shown in [22] that there exist point configurations for which $\lceil \frac{n-1}{d} \rceil$ hyperplanes are required for shattering. The extremal setting is achieved by letting $S$ be the vertex set of a cyclic polytope.\footnote{A cyclic polytope is a convex polytope obtained from the convex hull of $n$ distinct points on a rational normal curve $(x, x^2, \ldots, x^d)$ in $\mathbb{R}^d$.} Furthermore, if $d$ is even and $n \geq 2$ it can be shown that $\lceil \frac{n}{d} \rceil$ hyperplanes are required.

While the counting linear separating families of arbitrary size does not seem to allow a closed formula, enumerating linear separating families of fixed sizes might turn out more
promising. We say that a linear separating family for $S$ is of maximum (minimum) size if its cardinality is the largest (smallest) among all linear separating families for $S$.

**Problem to Investigate 15.** What is the number of linear separating families of minimum size?

**Problem to Investigate 16.** What is the number of linear separating families of maximum size?

4.1. **Separating families on Arbitrary Sets.** Dropping the linear requirement from the bipartitions, one can study the problem on general sets.

For this we define a *bipartition* of a set $S$ as either \{S\} or an unordered pair \{U,V\} of nonempty subsets of $S$ such that $U \cap V = \emptyset$ and $U \cup V = S$. Note that we allow \{S\} as a bipartition, because it corresponds to the case where the ground set $S$ is divided into $S$ and $\emptyset$. As before, a collection of bipartitions of $S$ is a *separating family* for $S$ if every two elements in $S$ are separated by some bipartition in the collection, that is, they are contained in different components of some bipartition. A separating family for $S$ is *minimal* if no proper subfamily is a separating family for $S$.

**Example 2.** Let $S = \{1, 2, 3, 4\}$ and let $P_1, P_2, Q_1, Q_2, Q_3$ be the bipartitions given as

\[
P_1 = \{\{1, 2\}, \{3, 4\}\}, \quad Q_1 = \{\{1\}, \{2, 3, 4\}\},
\]

\[
P_2 = \{\{1, 3\}, \{2, 4\}\}, \quad Q_2 = \{\{1, 2\}, \{3, 4\}\},
\]

\[
Q_3 = \{\{1, 2, 3\}, \{4\}\}.
\]

The family of bipartitions $\{P_1, P_2\}$ is a minimal separating family of minimum size for $S$, while $\{Q_1, Q_2, Q_3\}$ is a minimal separating family of maximum size. Here the size of a separating family denotes its cardinality.

The concept of separating families appears in the following search problem. Suppose that we are given a finite set $S$ and a collection $\{P_1, \ldots, P_m\}$ of bipartitions of $S$. For an unknown
element \( x \) in \( S \), we choose a bipartition \( P_i \) and we are allowed to ask which component of \( P_i \) contains \( x \), thereby narrowing down the range containing \( x \). The goal is to locate the unknown element \( x \) by asking a series of such questions. One can easily observe that for every element in \( S \) there exists a series of questions which leads to finding it if and only if \( \{P_1, \ldots, P_m\} \) is a separating family for \( S \). Rényi [47] initiated the study of the search problem described above, although he didn’t employ bipartitions but subsets of \( S \) as questions. Since then, many authors have studied combinatorial problems related to finding the minimum size of a separating family under various constraints (see [1] [32] [33] for a survey).

In this setting we obtained the following preliminary results using a bijection from the set of all minimal separating families of maximum size for \( S \) to the set of all spanning trees on \( S \).

**Theorem 4.1.** The number of minimal separating families of maximum size for an \( n \)-element set is \( n^{n-2} \).

In [52] we were furthermore able to prove the following enumeration result:

**Theorem 4.2.** The number \( \tau_{n,k} \) of separating families of size \( k \) for an \( n \)-element set with \( 2 \leq n \) and \( 1 \leq k \leq 2^{n-1} \) is

\[
\tau_{n,k} = \frac{(n-1)!}{k!} \sum_{i=1}^{k} (-1)^{k-i} \left[ \begin{array}{c} k \\ i \end{array} \right] \left( 2^i - 1 \right) \left( n-1 \right).
\]

4.2. **Separating Polygons using Lines.** Although it was shown in [24] that finding the minimum number of lines which shatter a point set in \( \mathbb{R}^2 \) is NP-hard, finding the minimal number of lines shattering a set of polygons can be done in polynomial time. Here minimal means, that if one removes a line then the remaining lines do not shatter the polygons anymore.

**Problem 4.3.** Given a set \( S \) of simple polygons in \( \mathbb{R}^2 \) with a total of \( n \) vertices, compute a set of \( n-1 \) or fewer lines which shatter \( S \) or determine that no such set exists.

As shown in Figure 13, not every collection of polygons is shatterable by lines.
In [23] two different algorithms which solve Problem 4.3 are presented. The first algorithm replaces each polygon by its convex hull, since a reflex vertex cannot intersect a shattering line. Furthermore, a plane sweep algorithm can be used to check if two of the input polygons intersect which implies that the arrangement cannot be shattered. The remaining operations use a so-called visibility graph (see Figure 14) which has a node for each polygon node and has an edge between vertices \( u \) and \( v \) if they can see each other, that is, if the line segment \( uv \) does not intersect the interior of any obstacle in \( S \). The visibility graph of \( S \) is computed and the candidate lines \( D \), i.e. the supporting lines through visibility edges which don’t intersect any object are determined. If we further restrict candidate lines to be internal double tangents between two objects, it follows that \( |D| \leq \min(|E|, |S|^2) \). If \( S \) is shatterable, then it is shatterable by lines in \( D \). We define a cluster to be a subset of \( S \) consisting of at least two polygons that are contained in the same cell. We add a candidate line \( l \) with angle \( \alpha \) from the negative \( y \)-axis to the solution set \( D \) if it intersects the convex hull of at least one cluster. We can test if \( l \) intersects the convex hull of a cluster by consulting the visibility profile \( VP(\alpha) \) of the convex hulls of the current clusters. \( VP(\alpha) \) consists of non-overlapping intervals in sorted order corresponding to the portions of convex hulls visible from infinity, looking in direction \( \alpha \). The key task is to efficiently maintain \( VP(\alpha) \) as \( \alpha \) varies from 0 to \( \pi \). Changes to \( VP(\alpha) \) only occur at angles corresponding to visibility edges \( E \). Thus, one can sort the edges in \( E \) by increasing slope, and radially swipe the arrangement. A change occurs if an edge \( e \) is contained in a tangent line between two visible clusters. When we encounter such an edge corresponding to a candidate line \( l \) by testing in \( O(\log n) \) time if \( l \)
stabs at least one cluster. If so, we add $l$ to $D$ and update $VP(\alpha)$ by breaking all clusters stabbed by $l$. The visibility graph can be computed in time $O(|E| + |V| \log |V|)$ and the total running time is $O(|E| + |V| \log |V| + |S|^2 \log |S|)$.

A second algorithm also replaces each polygon by its convex hull and then builds the visibility graph of the arrangement, but it works in a divide and conquer fashion. Although its running time is inferior to the previous algorithm, it has the advantage of being extendable to arbitrary dimensions. This algorithm searches for lines which stab the edges in the visibility graph, by looking at the dual space. Here, a point $p = (a, b) \in \mathbb{R}^2$ gets mapped to the line $p^* : y = ax - b$ in the dual plane and a non-vertical line $l : y = mx - c$ gets mapped to the point $l^* = (m, c)$. For example, the dual of a line segment is a double wedge where the intersection point of the two wedge lines corresponds to the supporting line of the line segment. An important characteristic of the dual transformation is that it preserves incidence relations, i.e. a point $p$ lies above (below) a line $l$ if and only if $l^*$ lies above (below) $p^*$. Thus, a line $l$ stabs a line segment $s$, if $l^*$ lies inside the double wedge $s^*$. We calculate the set $D^*$ of the duals of the candidate shattering lines $D$ and pre-process it in order to handle triangle range queries. In a triangle range query [28], a triangle is specified and the answer consists of all the points contained in the query triangle. Thus, such queries can be used to determine whether $l$ stabs $e$. Furthermore, if $e$ is an edge of the convex hull of the collection $S$ of the $k$ polygons, then the line through $e$ is guaranteed not to be a candidate line. Thus, if we find a candidate line $l$ that stabs $e$, we add $l$ to the solution set $D$ and remove all the other edges stabbed by $l$, we decompose $G$ into two connected components $G_1$ and $G_2$. For each $G_i$ of size at least 2 we recursively choose an edge on the convex hull of the obstacles defining the subgraph $G_i$ and search for a stabbing line. If no such line exists, we can conclude that the arrangement is not shatterable. However, the set $D$ provides the maximum partial shattering possible. The running time of the algorithm depends on the triangular range query time but is inferior to the first algorithm even with the best known triangular query data structure.
Since solving Problem 4.3 using the minimum number of lines is an NP-optimization problem, already an approximative answer is of interest. Formulating Problem 4.3 as a minimum set cover problem yields a solution which is at most a factor of $O(\log n)$ larger than the optimal solution. In an instance of the minimum set cover problem one is given a tuple $(S, \mathcal{R})$, with $\mathcal{R} \subseteq 2^S$ and the goal is to find a set $\mathcal{D} \subseteq \mathcal{R}$ which covers $S$, i.e. for which $\cup_{C \in \mathcal{D}} C = S$ holds. It is well known that this problem is polynomial time approximable within a factor of $(1 + \log |S|)$ and this is best possible [21]. If we now let $S$ be the edges of the visibility graph and for each candidate hyperplane $h$ we build a set $R_h \in \mathcal{R}$, $R_h$ which contains all the edges of the visibility graph which are stabbed by $h$, we obtain a $O(\log |S|)$-approximation algorithm.

**Problem to Investigate 1.** Find a constant factor approximation algorithm for Problem 4.3, or show that it is inapproximable within a factor of $\omega(\log n)$.

The authors of [24] remark that it might be possible that the bounded VC dimension of the problem allows an approach of Brönnimann’s geometric set cover algorithm [10].

![Figure 14. The visibility graph with its edges drawn as dashed lines and the shattering lines drawn as continuous lines.](image)
Figure 15. An example of a convex subdivision shattering where each cell is unbounded and no cell is empty.

Another interesting problem occurs when one does not only allow shatterings induced by half-planes but by an arbitrary convex subdivision of the plane as described in the following problem.

**Problem to Investigate 2.** Determine if there exists a convex subdivision of the set $S$ of objects such that each cell contains exactly one object. Figure 13 gives an example for which such a shattering does not exist.

If some convex cells are allowed to be empty, as argued in [42], one can compute the convex subdivision of the plane in time $O(n \log n)$ if $S$ is a set of line segments (or polygonal objects). This can be done by two line sweeps, one from left to right, extending the right endpoint of the line segments, and one from right to left, extending the left endpoint of the line segments. In general, the obtained subdivision depends on the order in which the line segments are extended. It is unique if and only if no extension meets any other extension.

Another variant of the problem is stated in the following problem.

**Problem to Investigate 3.** Determine if there exists a convex subdivision of the set $S$, of objects such that each cell contains exactly one object and every cell is unbounded (see Figure 15).
4.3. Separating Line Segments using Lines.

Problem 4.4. Given a set $S$ of $n$ disjoint line segments, find an arrangement of lines which shatter $S$.

In [44] an $O(n^2)$ time algorithm is proposed which finds a set of shattering lines or determines that no such set exists. This improves the $O(n^2 \log n)$ proposed in [24] by Freimer and Mitchel. The proposed algorithm can easily be extended if the objects are disjoint polygons keeping the time and space complexities invariant.

The algorithm works in the geometric dual plane which is defined as follows (see for example [19]). A point $p = (a, b) \in \mathbb{R}^2$ gets mapped to the line $p^* : y = ax - b$ in the dual plane and a non-vertical line $l : y = mx - c$ gets mapped to the point $l^* = (m, c)$. For example, the dual of a line segment is a double wedge where the intersection point of the two wedge lines corresponds to the supporting line of the line segment. An important characteristic of the dual transformation is that it preserves incidence relations, i.e. a point $p$ lies above (below) a line $l$ if and only if $l^*$ lies above (below) $p^*$. Thus, a line $l$ stabs a line segment $s$, if $l^*$ lies inside the double wedge $s^*$. Given an arrangement $S$ of line segments,
a line $l$ misses all segments in $S$ if $l^* \text{ lies outside } S^* = \{s^* : s \in S\}$. The line $l$ misses $S$ trivially, if $l^*$ lies above the upper envelope or below the lower envelope of $S^*$. In that case, all members of $S$ lie on the same side of $l$.

The double wedges in $S^*$ form a line arrangement $A(S^*)$, i.e. a subdivision of the plane induced by a set of lines, which according to the Zone Theorem (see for example [19]), can be constructed incrementally in time $O(n^2)$. Using a sweep line algorithm, the set of separating lines can then be obtained by finding all faces of the arrangement which are outside of any double wedge. Any point $l^*$ in such a region corresponds to a line $l$ which splits $S$ into $S_1$ lying above $l$ and $S_2$ lying below $l$. The sweep line algorithm gets extended using the following two data structures:

1. A list containing the lines in $S^*$ and the cells in $A(S^*)$ intersected by the sweep line at its current position, ordered from top to bottom.
2. A list of subsets of $S^*$ induced by the partition of the lines found so far.

The algorithm now sweeps the arrangement $A(S^*)$. At the intersection of two lines $l_1$ and $l_2$, we have to distinguish whether $l_1$ and $l_2$ correspond to the dual of the same line segment or not. Denoting by the degree of a cell, the number of double wedges overlapping the cell, $l_1$ and $l_2$ correspond to the dual of the same line segment if the degree of the new cell does not change (otherwise the degree of the new cell needs to be recalculated). If the degree of the new cell is zero, any point inside this cell corresponds to a separator for $S$. Using the second list, one can check if a line $l$ splits at least one of the remaining clusters, by checking whether there is a cluster which has its topmost line above the point $l^*$ and its bottom-most line below it.

The running time of $O(n^3)$ follows since the time of all operations in the algorithm is upper bounded by the splitting of the $O(n^2)$ many zero degree cells which takes time $O(n)$ per cell. Using an additional list and slightly changing the existing data structures, splitting a cluster into two can be done in constant time, thus yielding an $O(n^2)$ time algorithm.
References


E. W. H. Edelsbrunner. Halfplanar rangesearch in linespace and o(n0.695) query time. 1986.

E. F. Harding. The number of partitions of a set of n points in k dimensions induced by hyperplanes. 1966.


(I. Vigan) Dept. of Computer Science, City University of New York, The Graduate Center

*E-mail address: ivigan@gc.cuny.edu*