Research Online Publishing
www.ronpub.com

Open Access

Open Journal of Databases (OJDB)
Volume 2, Issue 1, 2015
http://www.ronpub.com/journals/ojdb
ISSN 2199-3459

Deriving Bounds on the Size of Spatial Areas

Erik Buchmann, Patrick Erik Bradley, Klemens Böhm

Institute for Program Structures and Data Organization, Karlsruhe Institute of Technology (KIT),
Am Fasanengarten 5, 76131 Karlsruhe, Germany,
{buchmann, bradley, klemens.boehm}@kit.edu

Abstract

Many application domains such as surveillance, environmental monitoring or sensor-data processing need upper and lower bounds on areas that are covered by a certain feature. For example, a smart-city infrastructure might need bounds on the size of an area polluted with fine-dust, to re-route combustion-engine traffic. Obtaining such bounds is challenging, because in almost any real-world application, information about the region of interest is incomplete, e.g., the database of sensor data contains only a limited number of samples. Existing approaches cannot provide upper and lower bounds or depend on restrictive assumptions, e.g., the area must be convex. Our approach in turn is based on the natural assumption that it is possible to specify a minimal diameter for the feature in question. Given this assumption, we formally derive bounds on the area size, and we provide algorithms that compute these bounds from a database of sensor data, based on geometrical considerations. We evaluate our algorithms both with a real-world case study and with synthetic data.

Type of Paper and Keywords

Regular research paper: *bounds, sensor data, spatial areas*

1 Introduction

Determining the size of a surface area that is covered by a certain attribute or feature from a database of samples is a standard task in many application domains. For example, meteorologists are interested in the area size where the thickness of the ozone layer is below 200 Dobson units, traffic planners need to know the size of the area where the density of the particles PM_{10} is above $50\mu g/m^3$, and firefighter units that are approved for radiation zones must know the size of the area where radiation is higher than $15mSv$. However, in many real-world applications it is an expensive and time-consuming task to take samples of the feature in question, be it by deploying sensor networks, by constructing weather stations or by taking samples manually. In consequence, the surface area frequently must be estimated from a rather small database. Furthermore, in many situations, some samples in the database may well be correlated. This makes it difficult to impossible to reliably determine the error of the surface area estimated. In such scenarios, upper and lower bounds of the area sizes allow a precise assessment of the information provided by the samples.

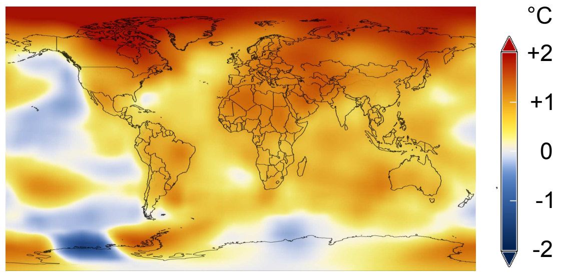

Figure 1: Temperature Anomalies [6]

To provide a concrete example, Figure 1 shows a color-encoded map of the database of five-year global temperature anomalies [6] from 1880 to 2012. Temperatures that are higher than the global mean temperature over a time interval of five years are shown in red, and lower than normal temperatures are shown in blue. Figure 1 has been estimated according to the contextual knowledge that temperature anomalies correlate strongly [7] for measuring stations separated by up to $1000km$. However, due to incomplete spatial coverage in the era before satellite-aided meteorology, the degree of uncertainty of the estimated temperature anomalies in this figure can be high.

Existing estimation methods, e.g., Random Sampling, Spline Lattices, or Voronoi Diagrams (cf. Section 2 for a more extensive list), are best-effort approaches. That is, such methods try to estimate the area in question as well as possible, but cannot provide guarantees on the minimal or maximal size of the area. The well-researched skyline database operator [13] does not solve this problem either. This is because it is based on the assumption that the values of interest describe a convex area that does not consist of multiple isolated regions. We strive for a different approach towards a database operator that provides bounds on area sizes. Our starting point is the assumption that one can specify a minimal diameter for the feature in question. This is typical in many scenarios:

- Temperature anomalies have a minimal radius of $r_{min} = 500km$ ([7], Figure 1).
- In meteorology a low pressure area is never smaller than some hundred kilometers.
- Thunderstorms have a minimal size and a borderline without sharp angles.
- The diffusion of fluids follows a circular pattern, and the minimal radius can be calculated from the Brownian motion depending on the temperature of the fluid.
- The minimal diameter of a car is $70cm$ (Bubble Car, e.g., BMW Isetta).

We formally derive upper and lower bounds on the size of an area that is covered by a certain feature. In particular, we ensure that, given a database of samples and a minimal radius of the feature, the area covered cannot be smaller or larger than what is expressed by our bounds. Except for straightforward bounds such as zero as the minimal area size, this is challenging. In particular, it is unclear how to specify these bounds, and how to transform this specification into a computable problem. In this paper, we make the following contributions:

1. We motivate the concept of computing bounds on the size of an area that is partly covered by a feature which in turn has a minimal radius.
2. We formally specify a lower bound $LB_{present}$ on the size of an area where the feature measured is present, and a lower bound LB_{absent} for an area where the feature measured is absent. The corresponding upper bounds are the total area considered minus $LB_{present}$ and LB_{absent}, respectively.
3. We develop measures to verify if the data set is consistent with the minimal radius, and if taking more samples is likely to bring the bounds much closer to the real area sizes.
4. We provide proof-of-concept algorithms that compute the bounds $LB_{present}$ and LB_{absent} from a database of samples by geometrical considerations, and we evaluate our approach by means of a real-world case study and by experiments with synthetic data.

Our experiments confirm the applicability of our approach with real-world problems and indicate directions for future research, e.g., to optimize the processing of our bounds for application scenarios that depend on "big data".

Paper Structure: The next section contains a running example and reviews related work. Section 3 describes our upper and lower bounds, followed by an experimental evaluation in Section 4 and a discussion of our approach in Section 5. Section 6 concludes.

2 Background

In this section, we introduce an application scenario which we use as our running example, and we outline related work.

2.1 Application Scenario

Think of a smart-city-infrastructure that has been deployed in an urban area to execute Directive 2008/50/EC on ambient air quality and cleaner air for Europe [5]. 2008/50/EC specifies that the smallest polluted area to be considered must have a size of at least $250m \cdot 250m$, i.e., the minimal radius of the area covered by the feature "$PM_{10} > 50\mu g/m^3$" is explicitly given. To enforce the directive, the infrastructure contains a number of measuring points that observe the air pollution with particles PM_{10}, and it controls a number of electronic road signs to re-route local motor-vehicle traffic out of each area where the density of PM_{10} exceeds the limit of $50\mu g/m^3$. Thus, the infrastructure possesses information of positions within a city where the pollution with PM_{10} is above and below the limit, but it does not know about the spatial extent of the pollution at these positions. The infrastructure must use this information to assess the size of the polluted areas in each city district, and it must

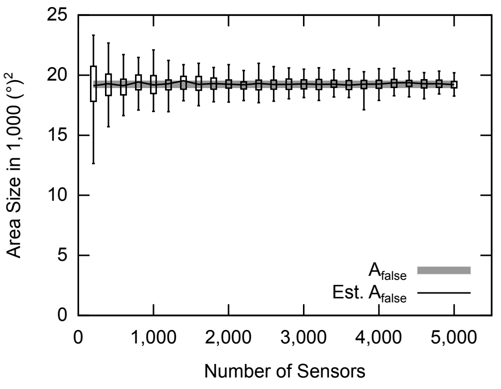

Figure 2: Real data set, Monte Carlo approach

take action in a way that local traffic is affected as little as possible.

A straightforward solution to obtain the size of the polluted area would be random sampling [4]: If the measuring points have been deployed independently from the sources of pollution, the polluted area can be estimated as the total area surface, multiplied with the number of sensors that have detected $PM_{10} > 50\mu g/m^3$, divided by the number of all sensors. However, the uncertainty of this approach is high for a number of sensor nodes that is realistic.

We illustrate this with a simple Monte Carlo simulation (Figure 2) where 100 to 5000 sensors have been distributed over the surface of the earth, i.e., over a grid with (latitude · longitude) $180° \cdot 360° = 64,800(°)^2$. The ratio of the unpolluted area from the total area is $A_{absent} = 29.68\%$. We have simulated each number of sensors 100 times. The vertical axis of Figure 2 shows the area size in 1.000 square degrees. The real area size is drawn in gray. The estimated area sizes are drawn as a standard box plot, where the whiskers denote the mean, the minimal and the maximal values of the estimation, and the boxes show the 25%- and 75% quantiles. The figure confirms that the Monte Carlo approach has a high degree of uncertainty, with few sensor tuples in particular. For example, with 200 sensor tuples the maximal and minimal estimations of the area size are $10,690(°)^2$ (≈ 131 Mio. km^2) away from each other. Since the surface of the earth has 510 Mio. km^2, this is a lot. Even with 5,000 samples, the maximal and minimal values differ by about $1,940(°)^2$ ($\approx$ 24 Mio. km^2), and it is impossible to tell if this is an over- or underestimation of the real area.

In our scenario, the smart-city-infrastructure wants to know the size of the areas within city districts that have been polluted in the best and in the worst case, to take actions with a minimal impact. Assume that, due to gas dispersion and wind drift, air-borne particles spread over an area of at least $50m$ radius around the source of pollution. Based on this information, it is possible to identify locations that definitively have not been polluted, e.g., if a disc with a radius $r_{min} = 50m$ does not fit into a group of sensors that are close together and have not detected PM_{10} above the allowed limit. We will exploit this idea to obtain bounds on the area size.

2.2 Related Work

To the best of our knowledge, we are first to compute bounds on the area size based on the weak assumption that the feature observed has a minimal diameter. Besides the random sampling method sketched in the last subsection, a number of approaches have been researched to estimate the size of a feature without this assumption. Such approaches can provide stochastic guarantees, i.e., that the border is inside a certain interval with a given confidence. However, they cannot provide upper and lower bounds, as required by our application scenario.

Geographic Information Systems (GIS) frequently use splines or polygon lattices to interpolate areas from a limited set of sampling points (“Kriging”, [9]). Such approaches can provide stochastic guarantees. Nevertheless, in real settings, 3% of the values interpolated with well-researched methods fall into the percentile “30% error and more” [8], i.e., it is impossible to obtain bounds this way.

In the context of wireless ad-hoc networks, approaches for *Boundary Recognition* [14] have been researched. Such approaches strive to detect holes and failures in the routing graph of the ad-hoc network. Boundary recognition approaches can be used to approximate position and perimeter of an area of interest if it is covered by sensor nodes. However, we want to provide guarantees even for areas that are not observed by sensors.

Approaches like *Voronoi Diagrams* or *Delaunay Triangulation* [1] partition a region into cells, according to a given set of discrete points. Each cell of a Voronoi Diagram contains one point at the center. The border between two cells is orthogonal to the straight line connecting their centers and is exactly in the middle between them. The Delaunay Triangulation is the dual graph of the Voronoi diagram. Both approaches are frequently used to visualize measured values, but it remains unclear how well the partitioning of the area matches reality. *Minimum Bounding Rectangles* [3] and *Spatial Skyline Queries* [13] approximate a region from a set of points. However, minimum bounding rectangles assume that a rectangle is a good approximation for a region, and skyline queries assume that the area is convex. Furthermore, both approaches require that the area in question does not consist of multiple isolated regions.

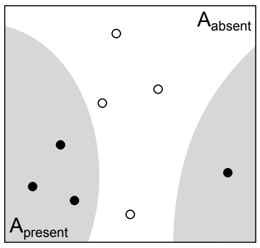

Figure 3: Scenario

The *Meeting Scheduling Problem* answers the question where and when a meeting should take place, given that all participants start at different places and need some time to reach the meeting location. Related algorithms [2] compute spatial and temporal bounds, based on a set of points. However, such approaches cannot be used to estimate the size of an area that is covered by a certain feature: In contrast to the meeting scheduling problem, our data set is incomplete, i.e., the feature in question might be present at unknown positions.

3 Bounds

In this section, we specify bounds in line with the scenario described, and we provide algorithms that compute these bounds based on geometric models. We also propose measures to verify if the data set is consistent with the minimal radius, and to evaluate if taking more samples brings the bounds closer to the real area sizes.

3.1 Formal Framework

Figure 3 serves as our running example. It shows sensor positions where the feature has been detected, represented as filled dots. Sensor positions where the feature has not been detected are shown as empty dots. The area $A_{present}$ where the feature is present is in light gray. We compute upper and lower bounds on the area sizes where the feature is present/not present for sure, based on the sensor positions and a minimal radius of the feature in question.

We formalize this problem as follows: We consider a geographic region of interest A in the Euclidean plane $\mathbb{R}^2$. In Figure 3, this is the entire area. Regions are infinite sets of points so that we can use set operations to express operations on geometric objects, e.g., $A_{absent} \cap A_{present} = \emptyset$. A sensor tuple $t = (pos, ft)$ stores a position $pos = (lat, long) \in A$, together with a feature value $ft \in \{present, absent\}$.

We consider two sensor databases $\mathrm{P}_{\mathrm{present}}$ and $\mathrm{P}_{\mathrm{absent}}$. Database $\mathrm{P}_{\mathrm{present}}$ is a finite set containing the positions $\mathrm{P}_{\mathrm{present}} := \{pos_1, pos_2, \cdots pos_n\}$ of all sensor tuples where the feature has been detected, i.e., $ft = present$. Analogously, database $\mathrm{P}_{\mathrm{absent}}$ stores all positions where $ft = absent$. $\mathrm{P}_{\mathrm{present}} \cap \mathrm{P}_{\mathrm{absent}} = \emptyset$, and each position $p \in \mathrm{P}_{\mathrm{present}} \cup \mathrm{P}_{\mathrm{absent}}$ corresponds to a unique point in A. This is because no sensor can observe the presence and absence of the feature at the same time, and no two sensors can be located at identical positions.

We assume that the feature we want to observe has a minimal radius r_{min}, i.e., it has at least the size and shape of a disc with radius r_{min}, but can be any larger.

Definition 1 (Feature Region): A region $A_{present}$ in the region of interest A is a **Feature Region** if the following holds: There exists a set of overlapping discs D such that

1. $\forall d \in D\colon d_{rad} = r_{min}$
2. $\forall p \in A_{present}\colon p \in \bigcup_{d \in D} d$
3. $\forall p \in \bigcup_{d \in D} d\colon p \notin A_{absent}$

$\square$

The properties require that it is possible for each point in the area $A_{present}$ to construct a disc d with radius r_{min} which does not overlap with A_{absent}. Thus, the perimeter of the feature must not contain sharp bulges. Figure 4 illustrates this. This definition of a feature region is is in line with our application scenario.

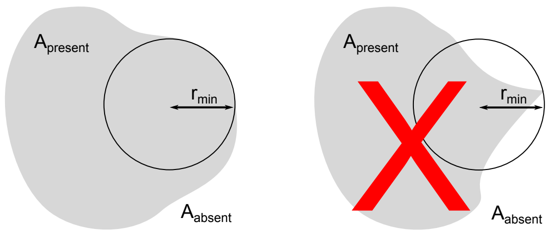

Figure 4: Feature Region with and without r_{min}

Definition 1 implies that if a sensor measures $ft = present$ at a certain position pos, we know for sure that at least an area of size $\pi \cdot (r_{min})^2$ is covered by the feature. This is true even if the real feature has an irregular shape (cf. Figure 4). Some of this area may be outside of A (cf. Figure 3). Our objective is to obtain upper and lower bounds on the size of the area where the feature is present ($A_{present} \subseteq A$, the light gray area in Figure 3) or absent ($A_{absent} \subseteq A$, the white area in Figure 3). Obviously, valid upper and lower bounds on $A_{present}$ are A and 0. However, such bounds are hardly useful in practice. We strive for bounds that are close to the real values. To do so, we define several auxiliary functions: $\mathrm{dist}()$ returns the distance between two points p, q. Without loss of generality, we use the Euclidean distance:

$$\mathrm{dist}(p, q) = \sqrt{(p.lat - q.lat)^2 + (p.long - q.long)^2} \quad (1)$$

A	A region of interest
r_{min}	Radius of a disc describing the minimal size of the feature in question
$\mathrm{P_{present}}, \mathrm{P_{absent}}$	Databases of points where the feature is present ($\mathrm{P_{present}}$) and absent ($\mathrm{P_{absent}}$)
$LB_{present}, LB_{absent}$	Lower bounds on the area size in A where $ft = present$ or $ft = absent$
$VI_{present}, VI_{absent}$	Volatility indicators for the bounds $LB_{present}, LB_{absent}$
$\mathrm{dist}(p_1, p_2)$	Distance of two points p_1, p_2
$\mathrm{area}(E)$	Total area of element E
$\mathrm{intpts}(D_1, D_2)$	The intersection points of the borders of two discs D_1, D_2

Table 1: Symbols and Notations

Furthermore, we define a function $\mathrm{area}(E)$ that returns the total area of a geometric figure E, which might be a disc, a triangle or an area of interest. Finally, $\mathrm{intpts}()$ returns the set of all intersection points of the borders of two discs D_1, D_2. Thus, $\mathrm{intpts}(D_1, D_2)$ returns $\emptyset$ if D_1, D_2 do not overlap, the two intersection points if D_1, D_2 overlap, and one point for touching discs.

3.2 Lower Bound LB_{absent}

In this subsection, we define a lower bound LB_{absent} on the size of an area A_{absent} where the feature is absent. Furthermore, we present an algorithm to calculate this value based on geometric considerations. Translated to our running example, we compute the size of the area A_{absent} that is within our region of interest A, and is not polluted with PM_{10} for sure. Note that $\mathrm{area}(A) - LB_{absent}$ is an upper bound on the size of the area where the feature measured is present. Thus, in our example it is an upper bound on the size of the area that might have been polluted in the worst case.

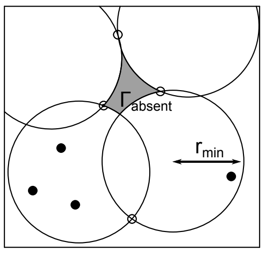

Figure 5: Bound LB_{absent} with a small r_{min}

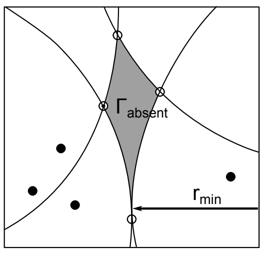

Figure 6: Bound LB_{absent} with a large r_{min}

Intuitively, we exploit that a disc with the minimal radius of the feature might not fit in between some points where $ft = absent$. This is true if the distances between these points are smaller than r_{min}. In Figure 5, our lower bound is the size of the dark gray area. Given the placement of the sensors that have measured $ft = absent$ (empty dots) and the minimal radius r_{min}, it can be excluded that any point in the gray area has $ft = present$, even if it has not been measured. However, all locations outside of this area might be polluted in the worst case.

Note that an increased r_{min} increases the area between the measurements with $ft = absent$ where we can exclude the presence of the feature. To illustrate, compare Figures 5 and 6: If r_{min} is increased, the bound LB_{absent} (the gray area in the figure) becomes larger. Having clarified this, we will be able to compute a lower bound LB_{absent} on the size of the area $A_{absent} \subseteq A$ where a feature with a minimal radius r_{min} is not present.

We compute LB_{absent} by constructing a geometric object Γ_{absent} (cf. Figures 5 and 6) from all points in A where we can exclude the presence of the feature. That is, the lower bound LB_{absent} proposed in the following is based on geometric considerations: We refer to a disc d^{pos}_{rad} in the Euclidean plane $\mathbb{R}^2$ by means of its center d^{pos} and radius d_{rad}. Our starting point for the computation of a lower bound LB_{absent} is the set of all discs possibly having the feature everywhere in A, as follows:

Definition 2 (Possible Discs $\mathrm{D_{ftpossible}}$): The set of **Possible Discs** $\mathrm{D_{ftpossible}}$ contains all discs d that fulfill the following properties:

1. $d_{rad} = r_{min}$
2. $\exists p \in d\colon p \in A$
3. $\nexists q \in d\colon q \in \mathrm{P_{absent}}$

□

That is, any disc in $\mathrm{D_{ftpossible}}$ has radius r_{min} (first property). Furthermore, it includes at least one point in the region of interest A (second property), but does not contain a point where a sensor has measured $ft = absent$ (third property). The area of A minus each point in any disc in $\mathrm{D_{ftpossible}}$ is our LB_{absent}, i.e., it is the dark gray area in Figures 5 and 6:

$$LB_{absent} = \mathrm{area}\left(A \setminus \bigcup_{d \in \mathrm{D_{ftpossible}}} d \right) \qquad (2)$$

Lemma 1: *LB_{absent} is a lower bound on the size of the area $A_{absent} \subseteq A$.* □

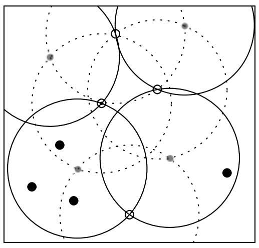
Figure 7: Outside

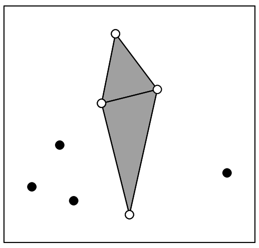
Figure 8: Inside

Proof sketch: This follows from the construction of possible discs. In particular, since no disc in $\mathrm{D}_{\mathrm{ftpossible}}$ must include a point in $\mathrm{P}_{\mathrm{absent}}$ and the set of discs is exhaustive, the area that is not overlapped by any disc must be a lower bound on A_{absent}. This holds if r_{min} is a minimal radius for the feature in question.

3.2.1 Computing LB_{absent}

To arrive at a value for LB_{absent}, we determine the area size of a geometric object Γ_{absent}. Γ_{absent} is described by two components where the first one is subtracted from the second one: a set of discs that are just *outside* of Γ_{absent} (the discs with solid lines in Figure 7), and a set of triangles that overlaps with its *inside* (the gray area in Figure 8).

Observation 1: *Γ_{absent} and $A_{present}$ are disjoint.* $\square$

In the following, we will show how to construct the components of Γ_{absent} so that we obtain the lower bound sought.

Outside: To distinguish the inside from the outside of Γ_{absent}, we define a set of discs $\mathrm{D}_{\mathrm{closeby}} \subset \mathrm{D}_{\mathrm{ftpossible}}$ (solid-line discs in Figure 7). The discs are outside of the area A_{absent}, but as close as possible to all positions where $ft = absent$ has been measured:

Definition 3 (Close-by Discs $\mathrm{D}_{\mathrm{closeby}}$): The set **Close-by Discs** $\mathrm{D}_{\mathrm{closeby}}$ consists of all discs d that fulfill two properties:

1. $d_{rad} = r_{min}$
2. $\exists p, q \in \mathrm{P}_{\mathrm{absent}} : d^{pos} \in \mathrm{intpts}(disc^{p}_{r_{min}}, disc^{q}_{r_{min}}) \wedge \forall x \in \mathrm{P}_{\mathrm{absent}} \setminus \{p, q\} : d^{pos} \notin disc^{x}_{r_{min}}$

$\square$

The first property specifies the radius d_{rad} of the discs. The second property requires that the center d^{pos} of each disc $d \in \mathrm{D}_{\mathrm{closeby}}$ is an intersection point of the discs around any two points in $\mathrm{P}_{\mathrm{absent}}$ (dashed discs in Figure 7). Furthermore, this intersection point must not lie within another disc around a position in $\mathrm{P}_{\mathrm{absent}} \setminus \{p, q\}$. Note that the set of discs is exhaustive, i.e., there is no disc $d' \notin \mathrm{D}_{\mathrm{closeby}}$ which fulfills the properties described.

Observation 2: *For a given sensor database $\mathrm{P}_{\mathrm{absent}}$, there is only one set of discs $\mathrm{D}_{\mathrm{closeby}}$ that fulfills these properties, i.e., the set of close-by discs is unambiguous.* $\square$

Only discs which overlap with the border of Γ_{absent} are required to compute the bound. Thus, $\mathrm{D}_{\mathrm{closeby}}$ might contain more discs than necessary to obtain Γ_{absent}.

Inside: We now define a set of triangles $\mathrm{T}_{\mathrm{inside}}$ that include A_{absent}.

Definition 4 (Interior Triangles $\mathrm{T}_{\mathrm{inside}}$): The set **Interior Triangles** $\mathrm{T}_{\mathrm{inside}}$ consists of all triangles t that fulfill three properties:

1. t is bounded by a set of three pairwise different vertices $\{v_1, v_2, v_3\}$ s.t. $v_1, v_2, v_3 \in \mathrm{P}_{\mathrm{absent}}$
2. $\forall t_1, t_2 \in \mathrm{T}_{\mathrm{inside}} : t_1 \cap t_2 = \emptyset$
3. $\forall \{v_1, v_2, v_3\} \in \mathrm{T}_{\mathrm{inside}} : \max\big(\mathrm{dist}(v_1, v_2), \mathrm{dist}(v_2, v_3), \mathrm{dist}(v_1, v_3)\big) \leq 2 \cdot r_{min}$

$\square$

The first property says that each triangle is a 3-tuple of disjoint vertices from the set of sensor positions where the feature is absent. The second property requires non-overlapping triangles which may have a common edge or vertex. Finally, we rule out that an area with $ft = absent$ and minimal diameter is inside a triangle by requiring that the edges of each triangle have a length of at most $2 \cdot r_{min}$. The set of triangles is exhaustive, i.e., there is no triangle $t' \notin \mathrm{T}_{\mathrm{inside}}$ which fulfills the properties described. Figure 8 shows the set of triangles that follows from this definition.

Observation 3: *Definition 4 allows several different decompositions of $\mathrm{P}_{\mathrm{absent}}$ into triangles. However, all decompositions allowed have the same area size.* $\square$

A concrete decomposition might depend on the algorithm used to create the triangles. However, since Definition 4 requires that all points in $\mathrm{P}_{\mathrm{absent}}$ are part of at least one triangle, the area size of Γ_{absent} does not depend on the particular layout of the triangles.

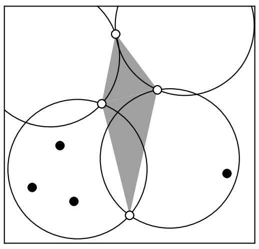
Figure 9: Subtracting the outside from the inside

Area Size of Γ_{absent}: Finally, we introduce $F_{absent} : (\mathrm{P_{present}}, \mathrm{P_{absent}}, A, r_{min}) \to \mathbb{R}$, a function that returns the area size of Γ_{absent}. Intuitively, F_{absent} returns the area of the triangles $\mathrm{T_{inside}}$ clipped by the discs $\mathrm{D_{closeby}}$, as illustrated in Figure 9. This results in the dark gray area Γ_{absent} shown in Figure 5.

$$F_{absent} = \text{area}\Big(\bigcup_{t \in \mathrm{T_{inside}}} t \setminus \bigcup_{d \in \mathrm{D_{closeby}}} d \Big) \qquad (3)$$

We conclude:

Lemma 2: *F_{absent} returns the lower bound LB_{absent}.* □

Proof sketch: The lemma holds, because F_{absent} returns the size of the area of Γ_{absent}, and Γ_{absent} is a geometric object consisting of all points in A not overlapped by any disc in $\mathrm{D_{ftpossible}}$ (see Lemma 1).

3.2.2 An Algorithm for F_{absent}

We now discuss how a lower bound on A_{absent} can be computed. Note that our algorithm is a proof-of-concept, i.e., performance considerations will be part of our future work.

Algorithm 1 computes F_{absent} in three steps: First, we compute all intersection points between the borders of all discs around points in $\mathrm{P_{present}}$ (Lines 1 and 2), but remove intersection points that are inside another disc (Line 3). This corresponds to computing all center points in $\mathrm{D_{closeby}}$. Second, we compute the set of triangles $\mathrm{T_{inside}}$ that overlap with the inside of A_{absent} (Line 4). We do so by adapting the Flip-algorithm for Delaunay-Triangulation so that it removes all triangles where one or more edges are longer than $2 \cdot r_{min}$. Third, we subtract from the surface of $\mathrm{T_{inside}}$ all overlapping segments of the discs in $\mathrm{D_{closeby}}$. Thus, this step computes Γ_{absent}. We use an R-tree to do this efficiently (Line 5). The bound is the size of the remaining area (Line 6).

3.3 Lower Bound $LB_{present}$

Now we specify a lower bound $LB_{present}$ on the size of the area $A_{present}$ where the feature is present, and we introduce an algorithm to calculate this value. Regarding our running example, we obtain the size of an area that is placed within the region of interest A, and is polluted with certainty. Furthermore, the value $\text{area}(A) - LB_{present}$ is an upper bound on the size of the area where the feature is absent, i.e., on the size of the area that is unpolluted in the best case.

Intuitively, since the feature in question is at least a disc with radius r_{min}, a lower bound on the size of the area where $ft = present$ is the area size of a geometric object $\Gamma_{present}$ that is a set of overlapping discs with a minimal area that can explain all sensor positions in $\mathrm{P_{present}}$ and $\mathrm{P_{absent}}$.

```
  Input: Set of points $\mathrm{P_{absent}}$
  Result: $F_{absent}$
  // Compute the outside
1 Discs $D := generateDiscs(r_{min}, \mathrm{P_{absent}})$;
2 Points $I := computeIntersections(D)$;
3 $I := pruneIntersections(I)$;
  // Compute the inside
4 Triangles
  $T := generateTriangles(r_{min}, \mathrm{P_{absent}})$;
  // Obtain the bound
5 GeoShape $G := subtract(T, generateDiscs(I))$;
6 return $computeSurface(G)$;
```

Algorithm 1: *compute F_{absent}*

Figure 10 depicts $\Gamma_{present}$ as a dark gray area. In the figure, it is the area of two disc segments with radius r_{min} that contain all points where the feature has been detected, and that have been placed s.t. its surface within A is as small as possible. This corresponds to a scenario where two places just outside of the observed urban area have been polluted with PM_{10}, and only a small part of the traffic within a city district needs to be re-routed.

Note that our lower bound depends on the measurements and r_{min}. For example, with a small r_{min} multiple overlapping discs are needed to construct a $\Gamma_{present}$ that covers all positions where the feature has been measured with a minimal total area size (Figure 11). In contrast, a large r_{min} means that large parts of the feature might be outside of A. A comparison of Figure 3 with Figure 10 provides an intuition for this effect.

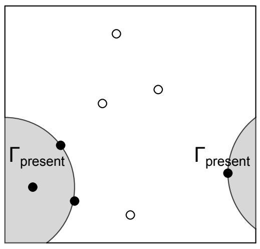

Figure 10: Bound $LB_{present}$ with a large r_{min}

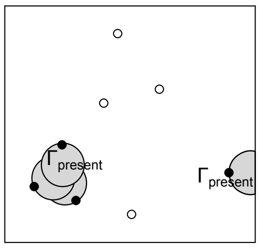

Figure 11: Bound $LB_{present}$ with a small r_{min}

In the following, we introduce a lower bound $LB_{present}$ on the size of the area $A_{present} \subseteq A$ where a feature with a minimal radius r_{min} is present. Our starting point is a set of discs:

Definition 5 (Covering Discs $\mathrm{D}_{\mathrm{cover}}$): Covering Discs $\mathrm{D}_{\mathrm{cover}}$ is a set of discs so that:

1. $\forall d \in \mathrm{D}_{\mathrm{cover}}: d_{rad} = r_{min}$
2. $\mathrm{P}_{\mathrm{present}} \setminus \bigcup_{d \in \mathrm{D}_{\mathrm{cover}}} d = \emptyset$
3. $\mathrm{P}_{\mathrm{absent}} \cap \bigcup_{d \in \mathrm{D}_{\mathrm{cover}}} d = \emptyset$
4. Let $\mathbb{D}$ be the family of discs fulfilling Axioms 1, 2, 3. Then $\nexists D' \in \mathbb{D}$ s.t. $\mathrm{area}(A \cap \bigcup_{d' \in D'} d') < \mathrm{area}(A \cap \bigcup_{d \in \mathrm{D}_{\mathrm{cover}}} d)$

□

Thus, $\mathrm{D}_{\mathrm{cover}}$ contains a set of discs with radius r_{min} (first property) that overlap with all points in $\mathrm{P}_{\mathrm{present}}$ (second property), but no points in $\mathrm{P}_{\mathrm{absent}}$ (third property). The fourth property requires that there does not exist any set of discs whose area size in A is smaller than the one of $\mathrm{D}_{\mathrm{cover}}$.

Observe that this definition allows an infinite number of valid instances of $\mathrm{D}_{\mathrm{cover}}$ if the points in $\mathrm{P}_{\mathrm{present}}$ and $\mathrm{P}_{\mathrm{absent}}$ do not enforce a certain placement of discs. For example, with our example in Figure 10 only one set $\mathrm{D}_{\mathrm{cover}}$ with a minimal area size exists (fourth property). On the other hand, Figure 12 shows a scenario where multiple placements of discs exist that have the same minimal area size. However, the rightmost instance in Figure 12 is invalid because $\mathrm{D}_{\mathrm{cover}}$ overlaps with $\mathrm{P}_{\mathrm{absent}}$.

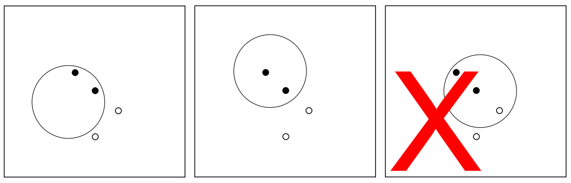

Figure 12: Valid and invalid covering discs

Observation 4: *$\mathrm{D}_{\mathrm{cover}}$ and A_{absent} are disjoint.* □

$LB_{present}$ then is the area size of the set of discs $\mathrm{D}_{\mathrm{cover}}$ in A:

$$LB_{present} = \mathrm{area}\left(A \cap \bigcup_{d \in \mathrm{D}_{\mathrm{cover}}} d \right) \quad (4)$$

Lemma 3: *$LB_{present}$ is a lower bound on the size of the area $A_{present} \subseteq A$.* □

Proof sketch: This follows from our construction of $\mathrm{D}_{\mathrm{cover}}$. In particular, since all points in $\mathrm{P}_{\mathrm{present}}$ are covered by at least one disc, and the total area size of all discs in $\mathrm{D}_{\mathrm{cover}}$ is required to be minimal, this area size must be a lower bound on $A_{present}$. This holds as long as r_{min} is a minimal radius for the feature in question.

3.3.1 Computing $LB_{present}$

To compute our lower bound $LB_{present}$, we determine the area size of a geometric object $\Gamma_{present}$ that consists of one concrete set of discs, as shown in Figure 10. In the following, we will show how to construct $\Gamma_{present}$ so that we obtain $LB_{present}$. For this purpose, we consider the properties of the set of discs, the number of the discs and their placement in A.

Properties of the Discs Our construction of $\Gamma_{present}$ is based on the following discs:

Definition 6 (All Possible Discs $\mathrm{D}_{\mathrm{allpres}}$): The set **All Possible Discs** $\mathrm{D}_{\mathrm{allpres}}$ contains all discs d that fulfill the following properties:

1. $d_{rad} = r_{min}$
2. $\forall p \in \mathrm{P}_{\mathrm{present}}: p \in \bigcup_{d \in \mathrm{D}_{\mathrm{allpres}}} d$
3. $\forall q \in \mathrm{P}_{\mathrm{absent}}: \{q\} \cap \bigcup_{d \in \mathrm{D}_{\mathrm{allpres}}} d = \emptyset$

□

Thus, $\mathrm{D}_{\mathrm{allpres}}$ is a set of discs with radius r_{min} (first property). The second property requires that all positions where a sensor has measured $ft = present$ are covered by at least one disc. No disc contains a point where $ft = absent$ (third property). Note that these properties overlap with Definition 5. However, in order to arrive at a computable bound, it is not sufficient to define that the set of discs has a minimal area size. Instead, we need to pay attention to the number and the locations of the discs, as we will explain.

Number and Placement of Discs: We construct $\Gamma_{present}$ from a subset $\mathrm{D}_{\mathrm{optpres}}$ of the infinite set of discs $\mathrm{D}_{\mathrm{allpres}}$. We derive this subset from the following observation:

Observation 5: *Definition 6 does not imply a certain minimal number of discs in $\mathrm{D}_{\mathrm{optpres}}$ greater than 1.* □

An upper bound on the number of discs in $\mathrm{D}_{\mathrm{optpres}}$ is the number of sensors $|\mathrm{P}_{\mathrm{present}}|$ that have observed the feature. However, in many cases it is optimal to cover multiple points in $\mathrm{P}_{\mathrm{present}}$ with the same disc (cf. Figure 10). That is, two or more discs might be congruent, i.e., share the same center point. Thus, in a first step we define a set $\mathrm{D}_{\mathrm{optpres}}$ that contains one disc per point in $\mathrm{P}_{\mathrm{present}}$. In a second step we obtain $\Gamma_{present}$ by placing the discs in $\mathrm{D}_{\mathrm{optpres}}$ so that they overlap as much as possible, i.e., $\Gamma_{present}$ fulfills Definition 5.

Definition 7 (Optimal Present Discs $\mathrm{D}_{\mathrm{optpres}}$): The set **Optimal Present Discs** $\mathrm{D}_{\mathrm{optpres}} \subset \mathrm{D}_{\mathrm{allpres}}$ contains one disc per point in $\mathrm{P}_{\mathrm{present}}$, i.e., $\forall p \in \mathrm{P}_{\mathrm{present}}: \exists d \in \mathrm{D}_{\mathrm{optpres}}$ with $p = d^{pos}$. □

We formulate the placement of discs as an optimization problem. The objective function $F_{present}: (\mathrm{P}_{\mathrm{present}}, \mathrm{P}_{\mathrm{absent}}, A, r_{min}) \to \mathbb{R}$ calculates the part of A covered by $\mathrm{D}_{\mathrm{optpres}}$. The optimization variables are the centers d^{pos} of the discs $d \in \mathrm{D}_{\mathrm{optpres}}$. The goal of the optimization is to minimize $F_{present}$ without violating the properties of $\mathrm{D}_{\mathrm{allpres}}$ from Definition 6:

$$\min_{\forall d^{pos} \in \mathrm{D}_{\mathrm{optpres}}} \left(\mathrm{area}\left(A \cap \bigcup_{d \in \mathrm{D}_{\mathrm{optpres}}} d\right)\right) \quad (5)$$

After having obtained the optimal placement of discs in $\mathrm{D}_{\mathrm{optpres}}$, we calculate the area size of $\Gamma_{present}$:

$$F_{present} = \mathrm{area}(A \cap \bigcup_{d \in \mathrm{D}_{\mathrm{optpres}}} d) \quad (6)$$

We conclude:

Lemma 4: *$F_{present}$ returns the lower bound $LB_{present}$.* □

Proof sketch: The lemma holds, because $F_{present}$ returns the size of $\Gamma_{present}$, and $\Gamma_{present}$ is the result of an optimization problem that computes one realization of $\mathrm{D}_{\mathrm{cover}}$ (see Lemma 3).

3.3.2 An Algorithm for $F_{present}$

In this subsection, we propose a proof-of-concept algorithm to compute $F_{present}$. This includes a nonlinear optimization problem that can be solved by different methods. Quasi-Newton approaches like DFP and BFGS [12] rely on the Hessian matrix, which is very difficult to calculate for complex nonlinear problems. We propose to use the Downhill Simplex [11], which transforms the optimization problem so that hill-climbing can solve it. The disadvantage of downhill simplex is a very small probability of ending up in a local optimum. However, both the runtime performance and the probability of avoiding a local optimum can be improved by using a good initial solution as a starting point.

Our algorithm consists of two steps. The first step (Line 1 in Algorithm 2) computes an initial solution. The second step (Lines 2-10) iteratively solves the optimization problem, i.e., it places the discs so that the area of $\Gamma_{present}$ is minimal and Lemma 4 holds.

Optimization Algorithm: Our optimization algorithm starts by generating a set of discs from all points in $\mathrm{P}_{\mathrm{present}}$ (Line 1 in Algorithm 2). Thus, this step obtains $\mathrm{D}_{\mathrm{allpres}}$. The next step creates the simplex (Line 2). For this purpose, the method $genSimplex(D)$ adds random values to the initial solution D and verifies that each node

```
Input: Set of points $\mathrm{P}_{\mathrm{present}}$, $\mathrm{P}_{\mathrm{absent}}$, $threshold$,
       $counter$
Result: $F_{present}$
// Obtain initial solution
1  Discs $D := setOfDiscs(\mathrm{P}_{\mathrm{present}})$;
   // Optimization procedure
2  Simplex $S := genSimplex(D)$;
3  double $d := \infty$;
4  while $counter > 0$ && 
   $d - alternatingSum(D) > threshold$ do
5      $counter := counter - 1$;
6      $d := alternatingSum(D)$;
7      $S := downhillSimplex(S)$;
8      $D := selectBest(S)$;
9  end
10 return $alternatingSum(D)$;
```

Algorithm 2: *compute $F_{present}$*

in the simplex still meets the optimization constraints, as specified in Definition 6.

With each iteration, the placement of discs will be modified in a way that the total area size within A is reduced, i.e., the algorithmm converges towards the set $\mathrm{D}_{\mathrm{optpres}}$. In particular, the Downhill Simplex algorithm (Line 7) folds the simplex [11] from the solution with the highest area size (worst solution) towards the solution with the smallest area size (best solution). We compute the area size of a set of discs as the alternating sum of overlapping disc segments, i.e.: (1) Compute the total surfaces of all discs, (2) subtract all surfaces of two overlapping discs, (3) add all surfaces of three overlapping discs, (4) subtract all surfaces of four overlapping discs, and so forth (Method $alternatingSum()$). Our optimization procedure uses two stop conditions (Line 4 in Algorithm 2): A maximal number of iterations ($counter$) and a convergence criterion ($threshold$). If a stop condition is met, the algorithm returns the area size of the best solution (Line 10).

Since Algorithm 2 assigns each point $p \in \mathrm{P}_{\mathrm{present}}$ its own disc (Line 1), the number of discs is equal to the number of points where the feature has been observed, i.e., $|D| = |\mathrm{P}_{\mathrm{present}}|$. The properties in Definition 6 ensure that the optimization procedure does not produce a placement of discs where a point $p \in \mathrm{P}_{\mathrm{present}}$ is not covered by any disc.

Algorithm 2 returns the exact value of $LB_{present}$ only if it converges to the optimal solution ($threshold = 0$). Otherwise, it returns an approximation. Note that, in some scenarios, multiple optimal solutions might exist.

Initial Solution: To let the Downhill Simplex (Line 7 in Algorithm 2) converge quickly towards the optimum, we strive for an initial solution that is already close to an optimal one. Furthermore, this initial solution should be efficiently computable. We have devised a recursive algorithm, shown in Algorithm 3, that is inspired by the Smallest Enclosing Discs approach [15]. However, our approach differs from [15], since we do not search for one smallest enclosing disc, but for a set of discs with fixed radius.

The input of Algorithm 3 is the set of points $\mathrm{P_{present}}$. The algorithm starts by calculating a set of discs in a way that each disc overlaps with a different set of points (Lines 5-8). The Method $d.shiftPos(\mathrm{P_{absent}}, \mathrm{P_{present}})$ in Line 6 moves the center of a disc so that it overlaps with the old center and as many points in $\mathrm{P_{present}}$ as possible, but does not overlap with a point in $\mathrm{P_{absent}}$. In a second step (Line 13), we insert all indispensable discs into the result set D. We say that a disc is indispensable if it contains one point from the input set which is not contained in any other disc. Finally, the algorithm recursively invokes itself with all points that are not contained in any indispensable disc (Line 15).

```
   Input: Set of points P
   Result: Set of discs D
   // Stop condition
1  if $P = \emptyset$ then return $\emptyset$;
2  ;
   // Create a set of unique discs
3  Discs $U := \emptyset$;
4  foreach $p \in P$ do
5      Disc $d$ := disc($p, r_{min}$);
6      d.shiftPos($\mathrm{P_{absent}}, \mathrm{P_{present}}$);
7      if $\nexists d' \in U : P \cap d = P \cap d'$ then
       $U := U \cup \{d\}$;
8      ;
9  end
   // Identify indispensable discs
10 Discs $D := \emptyset$;
11 foreach $d \in U$ do
12     if $\exists p \in P \cap d : p \notin P \cap \bigcup_{i \in D - \{d\}} i$ then
       $D := D \cup \{d\}$;
13     ;
14 end
   // Recursively assign remaining
      points
15 return $D \cup setOfDiscs(P - \bigcup_{j \in D} j)$;
```

Algorithm 3: *setOfDiscs*

3.4 Time Complexity of our Algorithms

Our algorithm to obtain $F_{present}$ requires to compute the alternating sum of overlapping disc segments (cf. Subsection 10). In theory, every disc might overlap with each other disc. Thus, $F_{present}$ has a worst-case time complexity of $O(|P|^2)$.

The time complexity of our algorithm for F_{absent} is $O(|P_{absent}|^2)$. This is because the algorithm generates $\mathrm{D_{closeby}}$ from intersection points of the borders of the discs around any position in P_{absent}. In the worst case, each of these discs might have two intersection points with any other one (cf. Subsection 3.2.1). However, the worst case requires that any pair of points is very close together. In real settings, such cases are unlikely, and spatial indexes such as R-trees can speed up the processing.

3.5 Correctness of Radius r_{min}

In some application areas, to compute the bounds derived so far a user might have to estimate the minimal diameter of the feature in question (cf. Section 2). Thus, it is important to find out if r_{min} is not correct, according to the values measured.

With a r_{min} value that is smaller than the real minimal diameter of the feature in question, $LB_{present}$ and LB_{absent} would produce bounds that are too small (cf. Figures 5 and 6). Thus, in use cases where the minimal radius of the feature in question cannot be obtained, we recommend to conservatively overestimate r_{min}.

Without knowing the real layout of the feature in the area, one cannot verify if the estimate of r_{min} is too small. On the other hand, we can prove that r_{min} is too large if the feature has been detected in between a set of measurements with $ft = absent$ in a way that would be forbidden, given the value of r_{min}. For example, consider Figure 13: The feature has been observed in between the three points where the absence of the feature has been measured. Thus, r_{min} is too large.

Lemma 5: r_{min} *is incorrect if the following holds:* $\exists p \in \mathrm{P_{present}}: \nexists pos$ *s.t.* $p \in disc^{pos}_{r_{min}} \wedge disc^{pos}_{r_{min}} \cap \mathrm{P_{absent}} = \emptyset$ □

Proof sketch: r_{min} is incorrect if there is a point $p \in \mathrm{P_{present}}$ so that all discs with radius r_{min} containing p also contain a point in $\mathrm{P_{absent}}$. Since this property is mutually exclusive with Definition 6, r_{min} must be incorrect.

Regarding Algorithm 2, we have identified an incorrect r_{min} if the optimization process cannot converge towards an optimal solution.

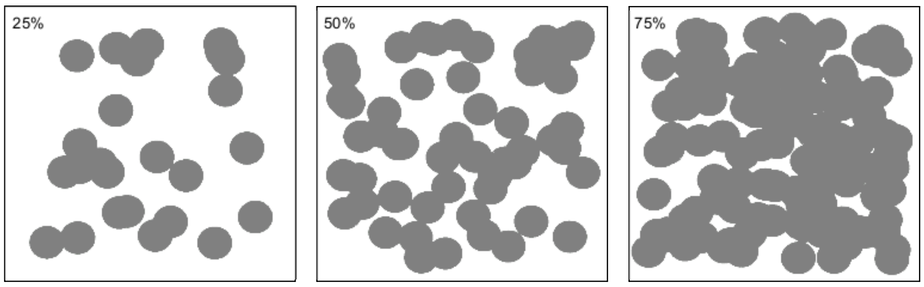

Figure 15: Synthetic test cases

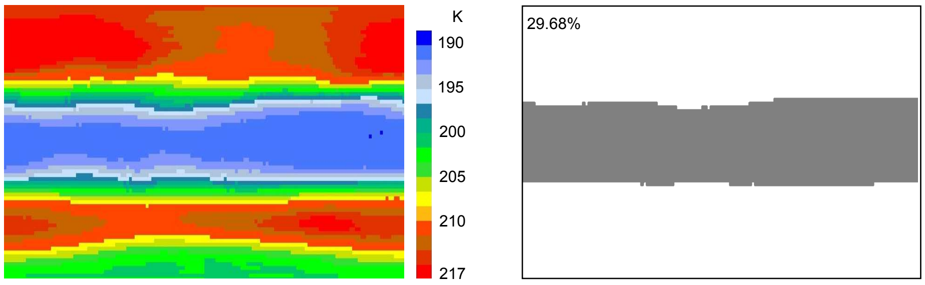

Figure 16: Real test case [10]

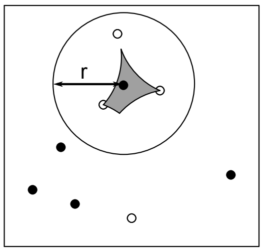

Figure 13: Verification

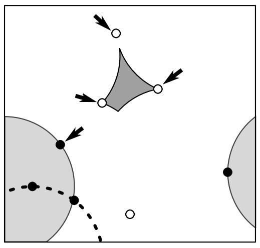

Figure 14: Bound Volatility

3.6 Volatility of our Bounds

From a practical point of view it is important to know how volatile our computed bounds are. For example, the operator of the smart-city infrastructure of our running example would like to compute the gain of taking more samples, i.e., to deploy more measuring points in order to have bounds that are closer to the real values.

Most well-known approaches to estimate the confidence of random samples cannot be used to quantify the volatility of our bounds. This is because such approaches usually require stochastically independent values. In our scenario, the feature in question has a minimal radius, and the sensor tuples measured depend on each other. In particular, sensors that have positions close to each other are likely to measure the same feature value.

Our volatility indicators have been inspired from the stop criteria of heuristics like Genetic Programming or Evolutionary Algorithms. Such algorithms use an iterative process to search for a quasi-optimal solution. The algorithms stop if the current solution improves the preceding one only marginally, i.e., the volatility measure is the difference between the current and the last solution. We define volatility indicators $VI_{present}, VI_{absent} \in [0,1]$ for the bound functions $F_{present}, F_{absent}$ by borrowing from this concept:

Definition 8 (Volatility Indicators VI_*): The **Volatility Indicators $\mathbf{VI_{present}}, \mathbf{VI_{absent}}$** are as follows:

Let $P = \mathrm{P_{present}} \cup \mathrm{P_{absent}}$ and $* = \{present, absent\}$

$$VI_* = \begin{cases} 1 & \text{if } F_* = 0 \\ 1 - \min\limits_{p \in P}\left(\frac{F_*(P \setminus \{p\}, A, r_{min})}{F_*(P, A, r_{min})}\right) & \text{otherwise} \end{cases}$$

□

The volatility indicators calculate the relative difference between the bound computed on database $P = \mathrm{P_{present}} \cup \mathrm{P_{absent}}$ and the bound computed on P minus the sensor tuple that has the highest impact on the bound. Thus, $VI = 1$ means that removing one value from the database changes the computed bound by 100%, i.e., taking more samples is likely to change the bound very much. $VI = 0$ refers to the opposite situation.

Figure 14 illustrates this volatility measure. In the figure, the sensor tuples with the highest impact on the bounds are highlighted with arrows. Consider function $F_{present}$. Without the feature detected (filled dot) in the middle of the area, the left part of the area $A_{present}$ would be reduced to the dashed area, resulting in $VI_{present} = 0.302$. Thus, the volatility indicator tells us that the computed bound is volatile. This is because it would decrease by up to 30.2% if one tuple was removed from the database of points. Now consider F_{absent}. Without any of the three values where $ft = absent$ in the upper part of the area A, the computed bound decreases to zero, i.e., $VI_{absent} = 1$. Thus, our volatility measure indicates that, in this case, the bounds might be far away from the real size of the area and could be improved by taking more samples.

4 Evaluation

In this section, we evaluate our algorithms with synthetic data and with real data from climatology research. We describe our experimental setup first, then our experiments.

4.1 Experimental Setup

In a nutshell, we want to find out how well our bounds converge towards the real area sizes with an increasing number of sensor tuples and with different coverage $A_{present}$ of the area with the feature in question. Note that we do not have to investigate the effect of the radius r_{min} or the total size of A. This is because r_{min}, $A_{present}$ and A are related: r_{min} cannot be larger than the feature in question, and an increased A has the same effect as decreasing $A_{present}$ and r_{min} at the same time. Thus, it is sufficient to vary the area $A_{present}$ that is covered by the feature.

We have computed the bounds with our algorithms for $F_{present}$ and F_{absent} on a Sun X2200 M2 server with two dual-core Opteron 2218 CPUs and 24 GB RAM. We have calculated the minimal, maximal, average values and the variance of the bounds calculated for synthetic and real data sets, which we will describe in the following. Since performance optimizations are not part of this paper, we will leave aside systematic runtime experiments.

Synthetic Setting: Experiments with synthetic data allow us to explore extreme cases and deviations between settings systematically. Our data set generator starts with an empty, squared area with a given size. We then iteratively add discs with a predefined radius at random positions. The stop criterion is the desired coverage of the area. Figure 15 shows exemplarily three of our synthetic test cases for an area where the coverage with the feature is 25%, 50% and 75%. By varying the coverage of the feature between 10% and 60% in steps of 2.5%, we have obtained 210 different data sets. Thus,

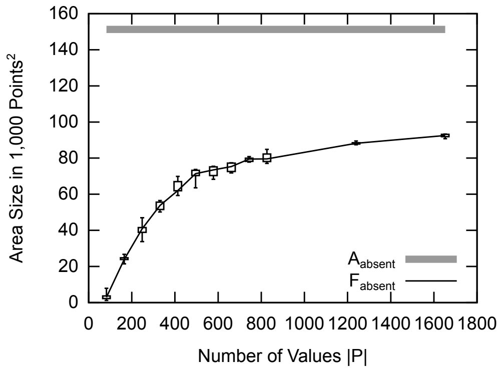

Figure 17: Synthetic data set, F_{absent} vs. $|P|$

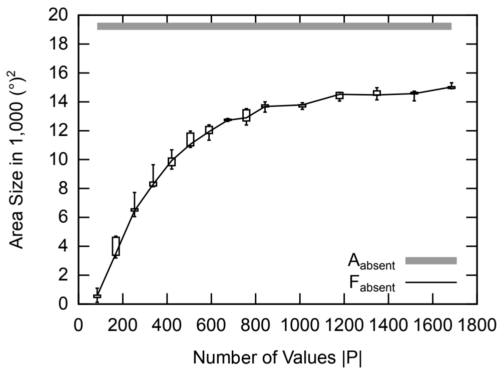

Figure 18: Real data set, F_{absent} vs. $|P|$

for each coverage percentage, we have 10 data sets. Furthermore, we have varied the database size from 10 to 1600, we have used the area sizes $500 \cdot 500$ points and $800 \cdot 800$ points, and the radii $r_{min} = 35$ and $r_{min} = 50$.

Our settings are difficult, because our feature has exactly radius r_{min}, and we generate data sets where the feature is distributed equally over A. In typical use cases, e.g., meteorology or pollution control, the feature is almost always much larger than r_{min} and usually forms several large clusters (cf. Figures 1 and 16). This makes it easier to identify an allowed decomposition into triangles (cf. Subsection 3.2.1) or a good placement of discs (cf. Subsection 3.3.1).

Real Setting: To assess the applicability of our approach on real data, we use data from the International Satellite Cloud Climatology Project [10]. It contains the mean Tropopause temperature of the earth since December 2009 (left part of Figure 16). The resolution of the data set is a $144 \cdot 72$ (longitude · latitude) grid with an edge length of 2.5°. Thus, we have a total of 10,368 data points, and a point on the equator is equivalent to a rectangle of $277.5 \cdot 278.4 km^2$.

To distinguish $A_{present}$ from A_{absent} we have used

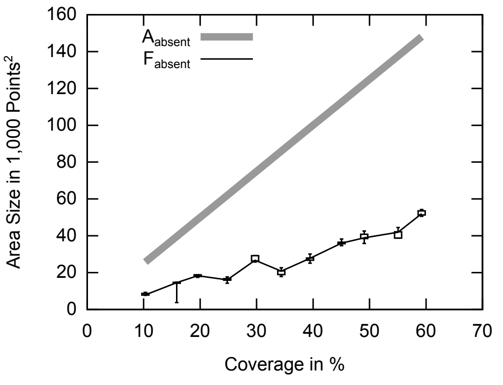

Figure 19: Synthetic data set, F_{absent} vs. coverage

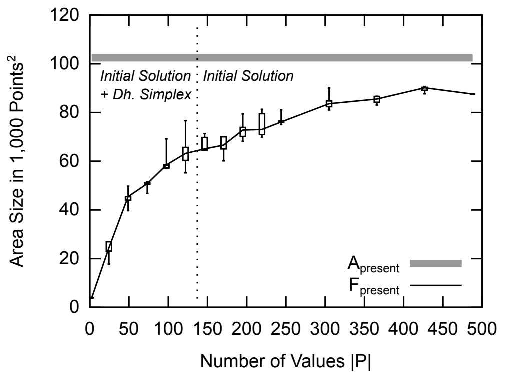

Figure 20: Synthetic data set, $F_{present}$ vs. $|P|$

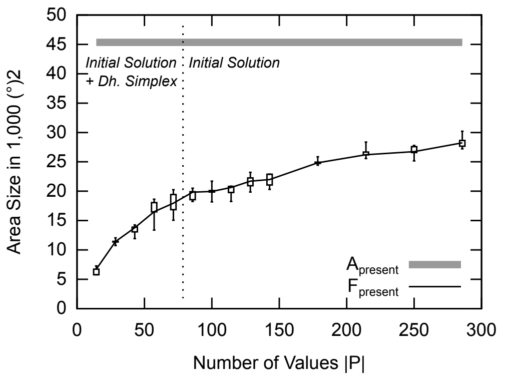

Figure 21: Real data set, $F_{present}$ vs. $|P|$

a threshold of 200 Kelvin, which results in a coverage of 29.68% of the area (right part of Figure 16). Preliminary experiments have shown that different thresholds produce similar test results. We assume that our feature has a minimal radius of 7.5°. This is a realistic assumption, since it has been observed [7] that temperature anomalies correlate strongly for measuring stations up to 1000km apart: 1000km corresponds to $\approx 3.6°$ on the Equator and $\approx 10°$ in Central Europe. Note that this scenario is the one which we have used for our Monte-Carlo simulation in Section 2.

4.2 Lower Bound F_{absent}

The Figures 17 and 18 show the output of our algorithm for F_{absent}, i.e., a bound on the size of A_{absent}. Figure 17 has been obtained with our synthetic data set with $800 \cdot 800$ points, 50% coverage and a radius $r_{min} = 50$. In comparison, Figure 18 results from experiments with our real data set (29.68% coverage). We have varied the number of points $|P|$, and we have repeated each experiment 10 times. The vertical axis of the figures show the area size in 1.000 square points (synthetic setting) or square degrees (real setting). The real size of A_{absent} is drawn as a thick gray line. The estimated area sizes are drawn as a standard box plot, where the whiskers denote the mean, the minimal and the maximal values of the estimation, and the boxes show the 25%- and 75% quantiles. The figures confirm that the computed bound is well below the real area size, and that the variability of the operator output is low, even for a very small number of 50 sensor tuples.

Furthermore, the figures confirms the expectation that, with an increasing number of sensor tuples, our algorithm strives towards the real size of A_{absent}. Recall that our synthetic setting (Figure 17) describes a challenging scenario. With our real data set (Figure 18), the result of F_{absent} comes closer to the real size of area A_{absent}, given a sufficient number of sensor tuples.

In direct comparison, a Monte Carlo approach (Figure 2) converges faster and is independent from the spatial distribution of the feature, but it cannot provide hard lower bounds and does come with a high variance of the values.

Figure 19 shows a series of experiments where we have varied the size of A_{absent} from 10% to 60% of our area of $800 \cdot 800$ points. For all experiments we have used $|\,\mathrm{P_{absent}}\,| = 200$ sensor tuples, i.e., since only positions where $ft = absent$ are used to calculate F_{absent}, we have kept $|\,\mathrm{P_{absent}}\,|$ constant. Thus, in a scenario where the feature covers 50% of the area, the total number of values in the database would be $|P| = 2 \cdot |\,\mathrm{P_{absent}}\,|$, given an equi-distributed placement of sensors. Again, the gray line shows the size of A_{absent}, and the box plot shows the computed F_{absent}. The figure indicates that, with our challenging setting, 200 sensor positions in $\mathrm{P_{absent}}$ result in a bound of about 25% of the real size of A_{absent}. This is in line with Figure 17.

With 200 values in $\mathrm{P_{absent}}$, our algorithm requires approx. 10 seconds to compute F_{absent}. Preliminary runtime experiments have shown that the runtime increases less than logarithmically with the number of values in $\mathrm{P_{absent}}$.

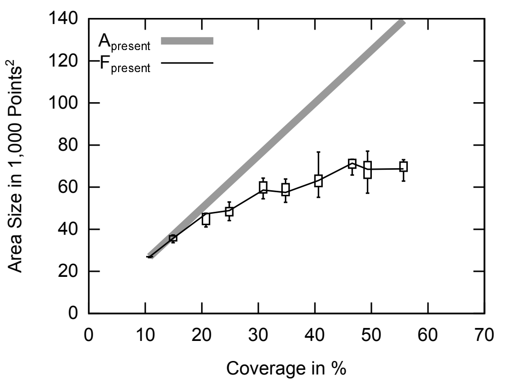

Figure 22: Synthetic data set, $F_{present}$ vs. coverage

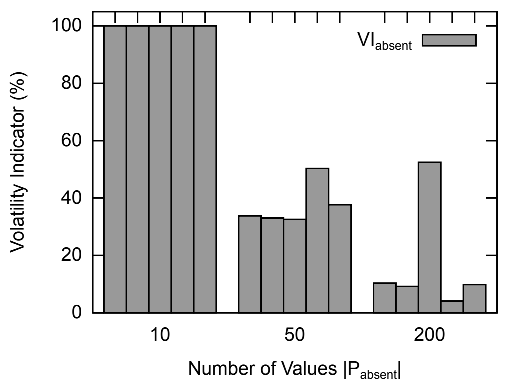

Figure 23: Volatility Indicator VI_{absent}

4.3 Lower Bound $F_{present}$

To test our algorithm for $F_{present}$, we have used the initial solution described in Subsection 3.3 as a starting point, and we have used Downhill Simplex [11] to solve the optimization problem.

Figures 20 and 21 graph the output of our algorithm for $F_{present}$ for the size of $A_{present}$. We have tested our synthetic setting with a grid of $500 \cdot 500$ points, $r_{min} = 35$ and 50% coverage. Figure 21 shows $F_{present}$ for our real setting, where $ft = present$ has a coverage of 70.32%. We have varied the number of sensor tuples $|P|$, and we have repeated each experiment five times. The axes and units of the figures are the same as in Figures 17 and 18. The figures confirm the expectation that, similarly to F_{absent}, our algorithm for $F_{present}$ converges towards the real area size with an increasing number of sensor tuples $|P|$.

Note that we have omitted the optimization stage for data sets of $|P| > 50$ in Figure 21, because, with our hardware configuration, the time per experiment exceeds 30 minutes for such data sets. The right side of the figure only shows our initial solution.

Figure 22 shows the computed bound $F_{present}$ for a series of experiments with the synthetic data set where we have varied the coverage of $A_{present}$ from 10% to 60%. For all experiments we have used $|\,\mathrm{P_{present}}\,| = 50$ sensors that have measured $ft = present$. The figure indicates that for a feature with 10% to 30% coverage, 50 sensors inside $A_{present}$ are sufficient to obtain a bound that is close to the real area size. This results from the fact that a few evenly-distributed sensors are sufficient for our algorithm, to identify the positions where the discs describing $\Gamma_{present}$ should be placed (cf. Figure 10). Furthermore, the figure indicates that our initial solution provides a reasonable bound in many cases. With our Sun X2200 M2 server, $F_{present}$ was computed in less 10 seconds from 30 values in $\mathrm{P_{present}}$. Preliminary runtime experiments have shown that the runtime of our algorithm increases logarithmically with an increasing number of values in $\mathrm{P_{present}}$.

4.4 Volatility Indicators

Finally, we evaluate the applicability of our volatility indicator VI_{absent} with an synthetic setting of $500 \cdot 500$ points and $r_{min} = 35$, 50% coverage. We have tested the database sizes $|P| = \{10, 50, 200\}$ five times each. We expect that VI_{absent} tends to have relatively small values for large sets of sensors. This is because a single value might not have a high impact on F_{absent} if the data set is large.

Figure 23 shows the results of our experiments. The figure shows that for each of the five experiments with database size $|P| = 10$ the volatility indicator is 100%. This indicates that the output of our algorithm for F_{absent} can be expected to change much with additional sensor tuples. Our experiments with $|P| = 200$ show a different outcome: Only one of the experiments uses a data set where taking additional samples might result in a F_{absent} that is much closer to the real value of $\mathrm{area}(A_{absent})$. Experiments with $VI_{present}$ have produced similar results.

5 Discussion

In this section, we will discuss options to improve the runtime of our algorithms, and we will point to further use cases for our bounds.

Surface Calculation: Computing the size of the total surface area of a large number of overlapping discs can be time-consuming. This issue appears with the optimization function of $F_{present}$ and with the computation of the exterior of F_{absent}. An option to improve this computation is to apply a stochastic approach that allows to compute the area of overlapping discs with low accuracy but quickly at the first iterations of the optimization problem, and with higher precision at later iterations. In particular, a Monte-Carlo simulation, combined with an R^+ Tree, could randomly select points in A and verify if it is inside one of the discs. The number of points that fall in one of the discs multiplied with the area size and divided by the number of all points is an estimate of the size of the surface area of all overlapping discs. The accuracy of this estimation of the area size depends on the number of points, i.e., it can be quick when a coarse estimation is sufficient to find out if two discs overlap to a large extent or not.

Optimization: A promising option to speed up the computation of $F_{present}$ is to reduce the complexity of the optimization problem. One approach is to compute $F_{present}$ from the maximal square that is enclosed by the disc describing the minimal size of the feature. Since this square is inside of a disc with radius r_{min}, the bound is still correct, although the difference to the real area size would be larger. This approach reduces the constraints to a set of linear equations. However, a function that computes the size of the area of overlapping squares is still a nonlinear one.

Sensor Placement In Subsection 3.6 we have described an approach to assess the volatility of the bounds, according to the deviation of the bounds when removing the most impacting sensor tuple from D. This volatility measure cannot only be used to find out if deploying more sensors is likely to result in bounds that are closer to the real size. It can be also used to identify ideal positions for the deployment of further sensors. An approach to determine such ideal positions according to our volatility measure would be to successively compute the position in A where a an observed presence or absence of the feature maximizes $F_{present}$ and/or F_{absent}.

Spatial Boundaries: F_{absent} is not only a bound of the total surface size of A_{absent}, but also its spatial boundaries. In particular, the sequence of disc segments that is defined by the interior minus the exterior (see Subsection 3.2) describes the total circumference and the spatial placement of the area where the feature is absent for sure. Note that $F_{present}$ computes only the area size. This is because the optimization approach searches for the minimal set of discs that can 'explain' a set of measurements without taking into account the actual placement of the disc borders.

6 Conclusion

Estimating upper and lower bounds on the surface of an area that is covered by a certain feature is an important task in many application domains. This is challenging, due to databases containing incomplete information, small samples and correlated values. Existing work does not tackle this problem. For example, best-effort approaches like random sampling do not provide upper and lower bounds, and database operators like skyline queries depend on the assumption of having a convex area.

In this paper, we have introduced formal specifications of upper and lower bounds on the size of the area where a certain feature with a minimal radius is or is not present for sure. Thus, we have taken the first steps towards a database operator that computes bounds on area sizes. To this end, we have proposed algorithms that compute the bounds, based on geometrical models. We have evaluated our algorithms with a real-world scenario and with synthetic data sets. Our experiments have shown that our approach can be readily applied to real world-problems up to a certain size.

Acknowledgements

We thank Johannes Walter for his excellent work on the source code and the experiments.

We acknowledge support by Deutsche Forschungsgemeinschaft and Open Access Publishing Fund of Karlsruhe Institute of Technology.

References

[1] F. Aurenhammer, "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, 1991.

[2] F. Berger, R. Klein, D. Nussbaum, J.-R. Sack, and J. Yi, "A Meeting Scheduling Problem Respecting Time and Space," *GeoInformatica*, vol. 13, no. 4, 2009.

[3] D. Caldwell, "Unlocking the Mysteries of the Bounding Box," *Coordinates*, vol. A, no. 2, 2005.

[4] R. Eckhardt, "Stan Ulam, John von Neumann, and the Monte Carlo Method," *Los Alamos Science*, vol. 15, 1987.

[5] European Parliament and the Council of the European Union, "Directive 2008/50/EC of the European Parliament and of the Council of 21 May 2008 on ambient air quality and cleaner air for Europe," Official Journal L 152 , 11/06/2008, p.01., 2008.

[6] Goddard Institute for Space Studies, "GISS Surface Temperature Analysis," Available at http://data.giss.nasa.gov/gistemp/, 2013.

[7] J. Hansen and S. Lebedeff, "Global trends of measured surface air temperature," *Journal of Geophysical Research*, vol. 92, no. 11, 1987.

[8] J. Hofierka, J. Parajka, H. Mitasova, and L. Mitas, "Multivariate Interpolation of Precipitation Using Regularized Spline with Tension," *Transactions in Geographical Information Systems*, vol. 6, no. 2, 2002.

[9] P. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind, *Geographical Information Systems: Principles and Technical Issues*. John Wiley and Sons, 1999.

[10] NASA, "International Satellite Cloud Climatology Project," Available at http://isccp.giss.nasa.gov, 2013.

[11] J. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, 1965.

[12] J. Nocedal and S. J. Wrigh, *Numerical Optimization*. Springer, 1999.

[13] M. Sharifzadeh and C. Shahabi, "The Spatial Skyline Queries," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06)*, 2006.

[14] Y. Wang, J. Gao, and J. S. Mitchell, "Boundary Recognition in Sensor Networks by Topological Methods," in *Proceedings of the 12th International Conference on Mobile Computing and Networking (MobiCom'06)*, 2006.

[15] E. Welzl, "Smallest Enclosing Disks (balls and Ellipsoids)," *Lecture Notes in Computer Science (LNCS)*, vol. 555, no. 8, 1991.

AUTHOR BIOGRAPHIES

Dr. Erik Buchmann earned his Ph.D. in Computer Science from the Otto-von-Guericke-University of Magdeburg, Germany in 2006. Since 2007, Erik Buchmann is head of the young research group "Privacy Awareness in Information Systems" at the Karlsruhe Institute of Technology, Germany. His research interests include management and query processing of any kinds of imprecise, individual-related, large-volume data sets, e.g., from wireless sensor networks, location-based systems or smart-grid installations.

Dr. Patrick E. Bradley received his Ph.D. in mathematics from Karlsruhe University, Germany, in 2002 with an emphasis in non-archimedean geometry. Afterwards, he joined the Architectural Faculty of Karlsruhe University where he became a researcher on building stock dynamics and building information modelling. He is currently working at the Institute of Photogrammetry and Remote Sensing at Karlsruhe Institute of Technology, Germany, as a researcher in spatial information modelling.

Dr. Klemens Böhm is full professor (chair of databases and information systems) at Karlsruhe Institute of Technology (KIT), Germany, since 2004. Prior to that, he has been professor of applied informatics/data and knowledge engineering at University of Magdeburg, Germany, senior research assistant at ETH Zürich, Switzerland, and research assistant at GMD – Forschungszentrum Informationstechnik GmbH, Darmstadt, Germany. Current research topics at his chair are knowledge discovery and data mining, data privacy and workflow management. Klemens Böhm gives much attention to collaborations with other scientific disciplines and with industry.