Enabling Scalable Geographic Service Sharing with Weighted Imprecise Voronoi Cells

Xike Xie, *Member, IEEE*, Peiquan Jin, *Member, IEEE*, Man Lung Yiu, Jiang Du, Mingxuan Yuan, and Christian S. Jensen, *Fellow, IEEE*

Abstract—We provide techniques that enable a scalable so-called Volunteered Geographic Services system. This system targets the increasing populations of online mobile users, e.g., smartphone users, enabling such users to provide location-based services to each other, thus enabling citizen reporter or citizen as a sensor scenarios. More specifically, the system allows users to register as service volunteers, or micro-service providers, by accepting service descriptions and periodically updated locations from such volunteers; and the system allows users to subscribe to notifications of available, nearby relevant services by accepting subscriptions, formalized as continuous queries, that take service preferences and user locations as arguments and return relevant services. Services are ranked according to their relevance and distance to a query, and the highest ranked services are returned. The key challenge addressed is that of scalably providing up-to-date results to queries when the query locations change continuously. This is achieved by the proposal of a new so-called safe-zone model. With safe zones, query results are accompanied by safe zones with the property that a query result remains the same for all locations in its safe zone. Then, query users need only notify the system when they exit their current safe zone. Existing safe-zone models fall short in the papers setting. The new model is enabled by (i) weighted and (ii) set weighted imprecise Voronoi cells. The paper covers underlying concepts, properties, and algorithms, and it covers applications in VGS tracking and presents findings of empirical performance studies.

Index Terms—Volunteered geographic service, weighted imprecise Voronoi Cell, spatial keyword search

1 Introduction

With the diffusion of on-line mobile users, it is relevant to provide scalable means of enabling such users to connect with other nearby users so that they may help each other with specific tasks. In North America,[1] the total numbers of car sharing members and vehicles are expected to grow from 126,911 and 22,315 in 2013, respectively, to 399,917 and 63,968 in 2017, respectively. It is of interests to consider offering more general voluntary services at scales. For example, drivers may be interested in tracking available nearby users who are willing to help with motor repair or are willing to provide travel directions or first aid.[2] In another example, a journalist arriving at a scene may like to connect with nearby users who witnessed an event that just occurred at the scene. In this type of scenario, mobile users are able to volunteer as providers of specific location-based services (LBS); and users are able to subscribe to notifications of available, nearby services, in turn enabling them to subscribe specific services.

We envision a so-called Volunteered Geographic Service (VGS) system that enables this scenario. Service volunteers can provide service descriptions and periodically updated locations to the system. Potential service users can provide subscriptions for relevant, nearby services, thus receiving in return notifications when there are changes to the services available.

Fig. 1 describes the setting. Here, users serve in two roles: as service users and as service providers.

More precisely, we call the service users query users, or simply users, when this does not cause ambiguity because they issue subscriptions to the system that take the form of continuous queries; and we call the service providers volunteers because they volunteer their services to other users.

The volunteers register descriptions of their services and provide their maximum speeds to the system. In addition, they periodically send their location to the system. In the figure, two users have volunteered services that are described by keywords. Query users issue subscriptions with the objective of being notified of available services that match their subscriptions. Specifically, a subscription is a continuous query that takes (i) keywords that describe the user's interests and (ii) the user's continuously changing location as arguments. In response to a query, the system continuously ranks all available services according to their relevance to the query keywords and the query location, and it returns the highest ranked services to the query user. In the figure, a user has subscribed to "first aid," and one service is relevant for the query.

1. http://www.sec.gov/Archives/edgar/data/1517492/000104746914008072/a2221526zf-1.htm
2. http://www.travel-buddies.com/

- *X. Xie and C.S. Jensen are with the Department of Computer Science, Aalborg University, Denmark. E-mail: {xkxie, csj}@cs.aau.dk.*
- *P. Jin and J. Du are with the Department of Computer Science, University of Science and Technology of China, Hefei, China. E-mail: jpq@ustc.edu.cn, duj@mail.ustc.edu.cn.*
- *M.L. Yiu is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. E-mail: csmlyiu@comp.polyu.edu.hk.*
- *M. Yuan is with Huawei Noah's Ark Lab. E-mail: yuan.mingxuan@huawei.com.*

Manuscript received 2 June 2015; revised 28 July 2015; accepted 30 July 2015. Date of publication 9 Sept. 2015; date of current version 6 Jan. 2016. Recommended for acceptance by K. Yi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2464804

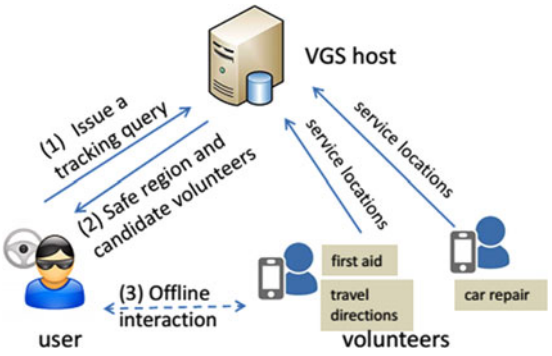
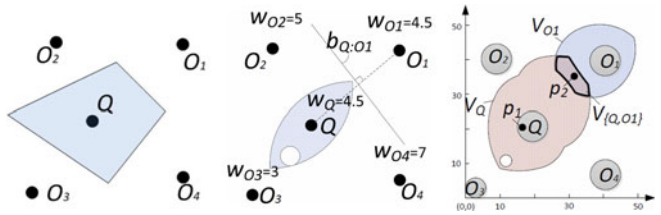


Fig. 1. VGS tracking.



(a) Plain Voronoi Cell (b) Weighted Voronoi Cell (c) Weighted Imprecise Voronoi Cell

Fig. 2. Voronoi cells.

The general approach to achieve efficient and scalable support for queries with continuously changing locations is to use so-called safe zones: each query result is assigned a safe zone that is a region that contains the query location and has the property that the query result is the same for all locations in the safe zone. Then the users need to notify the system only when they exit their safe zones, and the system needs only re-compute a query when notified.

Next, we assume that all volunteers update their locations at regular intervals. The choice of interval duration represents a tradeoff between the accuracy with which a volunteer's location is known and thus service quality on the one hand and system performance and volunteer location privacy on the other hand. For instance, statistics from location tracking services [1] show that the location update frequency is low (e.g., less than once per 2 minutes) for a large portion of objects, which introduces considerable location uncertainty. We use so-called *imprecise regions* to model the locations of volunteers. An imprecise region is a region that contains all possible locations over a period of time [2], [3], [4], [5], [6], [7], [8]. The size of a volunteer's imprecise region is thus determined by the product of the volunteer's maximum speed and update interval.

A query must be recomputed when: (1) the user exits the safe zone; (2) the safe zone is invalidated by an update to the database of volunteers. Condition 1 is checked by the user's device, and Condition 2 is checked on the server side.

The location of a query does not cause the current result of the query to change as long as the location remains in the safe zone. In our settings, three aspects must be taken into account when defining such safe zones: (a) the distances between the query locations and the locations of the volunteers; (2) the imprecision of volunteer locations; and (3) the relevance of volunteered service to the query.

The concept of safe zone has been studied in the past decade [9], [10], [11], [12], [13], [14], [15].

In the settings that consider only spatial distance and precise loctions [9], [10], [11], [12], [16], it is possible to use plain Voronoi Cells as safe zones. Given a set of point locations, the Voronoi cell for a point is the part of space that contains all other points in the underlying space that have the point as their nearest neighbor. Fig. 2a shows the Voronoi cell of Q. The cell is delineated by perpendicular bisectors between Q and the four other objects.

In other settings, queries also involve non-spatial attributes [13], [14], [15]. For example, restaurants have textual descriptions, such as "seafood" and "pizza." A query may then provide a query location and the keyword "pizza" to retrieve the restaurant that ranks the highest with respect to relevance to the keyword and spatial proximity. Here, weighted Voronoi cells [17] are used for building safe zones. These combine textual relevance and spatial proximity when ranking objects. An example cell is shown in Fig. 2b.

However, none of these existing notions of safe zones apply in our settings.

Instead, we propose so-called *Weighted Imprecise Voronoi cells* that can be used as safe zones in our setting. The weighted imprecise Voronoi cell of an object contains all locations that have non-zero probability of having the object as their weighted nearest neighbor (PNN). A PNN query returns all objects that possibly have the smallest weighted distance to a query point p.

We use Table 1 to show the weighted imprecise distances between the objects and the query points p_1 and p_2 in Fig. 2c. The weight representing the text relevance can be calculated at runtime by techniques in [18]. Since object locations are imprecise, the weighted distance is a range. Calculation details are shown in Section 2. Here, we observe that p_2 has two PNNs: Q and O_1. Object O_2 is not a PNN because its minimum weighted distance to p_2 is 3.6, which exeeds O_1's maximum weighted distance of 3.3. But we can not tell which of Q and O_1 is best, because the value ranges of their weighted distances overlap. So, both are possible.

We summarize two important features about the PNN query. *Nondeterminacy*: an object is in the result if it is possibly, but perhaps not definitely, p's nearest neighbor. *Non-uniqueness*: a query result may contain multiple objects.

TABLE 1
Weighted Imprecise Distances in Fig. 2c

Objects	Center	Radius	Weight
Q	(20,21)	4.5	4.5
O_1	(40,40)	4.5	4.5
O_2	(9,40)	4.5	5
O_3	(3,3)	3	3
O_4	(40,7)	4.5	7

Points	Locations
p_1	(23,23)
p_2	(31,35)

	Imprecise Distance		Weighted Imprecise Distance	
Obj	$\|p_1, O\|_E$	$\|p_2, O\|_E$	$\|p_1, O\|$	$\|p_2, O\|$
Q	[0, 8.1]	[13.3, 22.3]	[0, 1.8]	[3.0, 5.0]
O_1	[19.6, 28.6]	[5.8, 14.8]	[4.3, 6.3]	[1.3,3.3]
O_2	[17.5, 26.5]	[18.0, 27.0]	[3.5, 5.3]	[3.6, 5.4]
O_3	[25.3, 31.3]	[39.5, 45.5]	[8.4, 10.4]	[13.2, 15.2]
O_4	[18.9, 27.9]	[24.9, 33.9]	[2.7, 4.0]	[3.6, 4.8]

Two reflect these two features, we define two types of Voronoi cells. The weighted imprecise Voronoi cell (π-cell) of an object Q consists of all locations that have Q as a PNN. The set weighted imprecise Voronoi cell (ψ-cell) of a set $S = \{Q_i\}$ of object consists of all locations that have each Q_i as a PNN. Fig. 2c exemplifies π-cells and ψ-cells. Object Q's π-cell V_Q is colored red, and object O_1's π-cell V_{O1} is colored blue. Their overlap is the ψ-cell of $S = \{Q, O_1\}$. Thus, any point in V_Q has Q as its PNN, and any point in the ψ-cell has both Q and O_1 as its PNNs. Table 2 lists all notations used throughout this paper.

It is non-trivial to support such Voronoi cells. First, we shall see that such cells are delineated by complex curves. Second, cells can have holes and may even consist of disconnected regions. To the best of our knowledge, this paper presents the first study of weighted imprecise Voronoi cells.

Contributions. We study safe zones for service subscriptions in VGS settings, offering a comprehensive coverage of the concepts, properties, and algorithms needed for the use of weighted imprecise Voronoi cells as safe zones. First, we define half spaces, given by higher-order polynomials, and define the resulting Voronoi cells; and we propose succinct approximations with simple geometries, thus avoiding expensive materialization. Second, we provide two ways of rendering Voronoi cells, namely using an object representation [13], [19], [20] and a shape representation [21], [22], thus improving versatility. Third, we show how to support different weighted distance functions and arbitrarily shaped imprecise regions, further improving the versatility. Fourth, we report on empirical performance studies of our proposals with different index structures (e.g., aggregate R-tree, IR-tree [18]) using synthetic and real data.

Outline. The remainder of the paper is organized as follows. Section 2 presents concepts and properties of weighted imprecise Voronoi cells and describes the filter-and-refine framework adopted. Section 3 investigates half-space approximation and derives an early stop condition for the filtering phase. Section 4 covers the refinement phase. Section 5 extends the filter-and-refine framework for evaluating ψ-cells. Section 6 extends the proposed techniques to cover additional weighted distance functions. Section 7 reports on the empirical study. Section 8 covers related work, and Section 9 concludes.

2 Problem Setting and Definition

We consider a set $\mathbb{O}$ of imprecise objects in two-dimensional euclidean space $\mathbb{R}^2$. An imprecise object $O \in \mathbb{O}$ has the property that its exact location is unknown but is known to be inside a given region, called the object's imprecise region [3], [23], [24]. We assume that imprecise regions are circular and denote a circular region with center c_O and radius r_O by $\odot(c_O, r_O)$. Arbitrarily shaped regions are covered in Section 6. Each object O is also associated with a weight w_O. In summary, $O = (\odot(c_O, r_O), w_O)$.

We cover weighted imprecise distances in Section 2.1. We define bisectors, half spaces, and Voronoi cells for weighted imprecise objects in Section 2.2. We analyze corresponding geometric properties in Sections 2.3 and 2.4 and describe the framework for Voronoi cell construction in Section 2.5.

TABLE 2
Notation

Notation	Meaning
D	domain space
$\mathbb{O}$	a set of uncertain objects $\{O_1, \ldots, O_n\}$
S	a set of query objects $S = \{Q_1, \ldots, Q_m\}$
$\odot(c, r)$	a circle centered at c with radius r
c_O, r_O, w_O	center, radius, and weight of circular object O
$\|a, b\|_E$	Euclidean distance between a and b
$\|a, b\|_E^l, \|a, b\|_E^u$	lower and upper bound for $\|a, b\|_E$
$\|a, b\|$	weighted distance between a and b
$\|a, b\|^l, \|a, b\|^u$	lower and upper bound for $\|a, b\|$
$\|Q, O\|_{bd}$	min center-border distance between Q and O
$b_{Q:O}$	bisector of Q with respect to O
$H_{Q:O}$	half space from Q to O
$\overline{H_{Q:O}}$	complement half space from Q to O
$H^+_{Q:O}$	outer region of half space $H_{Q:O}$
Φ_Q, Φ^+_Q	partial Voronoi cell, approximated Partial cell
V_Q, V_S	π-cell, ψ-cell
C, C_Q, C^*_Q	candidate set, result set, optimal result set
$f_{\theta,\Delta}(p)$	transform point p from original space to θ-space
$f^{-1}_{\theta,\Delta}(p)$	transform point p from θ-space to original space
cvx(h),P_f(h)	convex hull, f-sided outer tangent polygon of h
β, β^+	$\beta = w_Q/w_O$, $\beta^+ = r_Q + r_O + w_Q - w_O$

2.1 Weighted Imprecise Distance

Let $|p, O|_E$ be the euclidean distance between point p and object O. If object O is precise, i.e., a point, distance $|p, O|_E$ is a deterministic value. Existing work on weighted distances assumes that O is precise.

Weighted distance. The weighted distance, between point p and object O is denoted by $|p, O|$ and is defined in terms of the euclidean distance $|p, O|_E$ and the weight w_O.

$$|p, O| = \frac{|p, O|_E}{w_O} \quad (w_O > 0). \tag{1}$$

This is a multiplicatively weighted distance [13], [14]. Section 6 extends the coverage to also include so-called additively weighted distances [15], [18]. Both kinds of distances satisfy the V-assignment rule [17] and are thus sufficient for defining Voronoi cells.

Imprecise euclidean distance. For an imprecise object O, the euclidean distance $|p, O|_E$ is a random variable, the values of which are bounded by the interval $[|p, O|_E^l, |p, O|_E^u]$.

$$|p, O|_E^l = \begin{cases} |p, c_O|_E - r_O & \text{if } p \notin \odot(c_O, r_O) \\ 0 & \text{otherwise} \end{cases}$$
$$|p, O|_E^u = |p, c_O|_E + r_O. \tag{2}$$

Weighted imprecise distance. We obtain the *weighted imprecise distance* $|p, O|$ by substituting the euclidean distance in Eq. (1) by the imprecise euclidean distance in Eq. (2). This yields the following bounds.

$$|p, O|^l = \begin{cases} \frac{|p, c_O|_E - r_O}{w_O} & \text{if } p \notin \odot(c_O, r_O), \\ 0 \end{cases} \quad |p, O|^u = \frac{|p, c_O|_E + r_O}{w_O}. \tag{3}$$

For example, in Fig. 2c, the minimum weighted imprecise distance between p_1 and Q is 0 because p_1 is inside Q. The maximum weighted imprecise distance between p_1 and Q is $(3.6 + 4.5)/4.5 = 1.8$. The weighted imprecise distance can be used for ranking objects with respect to a point p. The one that possibly has the smallest weighted distance is p's possible weighted nearest neighbor.

Definition 1. ***Possible weighted nearest neighbors*** *(*PNN*). Given a set of objects $\mathbb{O}$, the possible weighted nearest neighbors of a query point p is those objects in $\mathbb{O}$ for which no other object exists in $\mathbb{O}$ that is definitely closer to p.*

$$\mathrm{PNN}(p) = \{O_i \in \mathbb{O} \mid \nexists O_j \in \mathbb{O}\ (|p, O_j|^u < |p, O_i|^l)\}. \quad (4)$$

In other words, an object O_i cannot be a PNN of p if another object O_j exists that is definitely closer to p. For example, in Fig. 2b, Q is a PNN of p_2 because their minimum weighted distance ($|p_2, Q|^l$) is 2.96 (Table 1), and no other object's maximum weighted distance to p_2 ($|p_2, O_i|^u$) is smaller. Similarly, O_1 is also a PNN of p_2. This is captured by p_2 being in ψ-cell $V_{\{Q,O_1\}}$. We proceed to introduce the bisector and half space for a pair of weighted imprecise objects, based on which we define weighted imprecise Voronoi cells.

2.2 Weighted Imprecise Voronoi Cell

Given a pair of objects Q and O, we define the weighted imprecise bisector of Q with respect to O as: $b_{Q:O} = \{p \in D || p, Q|^l = |p, O|^u\}$. The bisector partitions the domain space into two. We call the part closer to Q the half space from Q to O, denoted by $H_{Q:O}$, and the other part closer to O the complement half space, denoted by $\overline{H_{Q:O}}$. In other words, $H_{Q:O} \cup \overline{H_{Q:O}} = D$ and $H_{Q:O} \cap \overline{H_{Q:O}} = \emptyset$.

Definition 2. ***Half space*** **($H_{Q:O}$).** *Given two objects Q and O and a weighted distance function, the half space from Q to O is:*

$$H_{Q:O} = \{p \in D \mid |p, Q|^l < |p, O|^u\}.$$

The half space from Q to O has the property that if a point p does not belong to it, p cannot be a PNN of Q; and if p does belong to the half space, p may be a PNN. This follows from the definition of PNN. We define the notion of a weighted imprecise Voronoi cell using half spaces.

Definition 3. *Given a set $\mathbb{O}$ of objects in a domain space D, the* ***Weighted Imprecise Voronoi Cell*** *(π-cell) of an object $Q \in \mathbb{O}$ is a subset V_Q of D such that for any $p \in V_Q$, Q has non-zero probability of being a PNN of p.*

$$V_Q = \bigcap_{O \in \mathbb{O} - \{Q\}} H_{Q:O}.$$

Thus, Q's π-cell is the intersection of all half spaces from Q to another object. This definition contends with the non-determinacy characteristic of PNNs. The notion of ψ-cell addresses the non-uniqueness feature.

Definition 4. *Given a set $\mathbb{O}$ of objects in a domain space D, the* ***Set Weighted Imprecise Voronoi Cell*** *(ψ-cell) of a set of objects $S = \{Q_1, \ldots, Q_m\}$ is a subset V_S of D such that for any $p \in V_S$, each object in S has a non-zero probability of being a PNN of p.*

$$V_S = \bigcap_{Q \in S} V_Q.$$

Here, S's ψ-cell is the intersection of all π-cells of objects in S. Thus, a π-cell is a special case of a ψ-cell where S contains only one object.

By letting S be the PNN of a point p, ψ-cells can be used to find the region in which a previously retrieved PNN result remains unchanged.

The intuitive way of building a π-cell for an object Q is to follow the definition and construct half spaces from Q to all other objects and then form the intersection. Similarly, a ψ-cell can be formed by intersecting π-cells. However, these straightforward methods are not promising, as will be discussed next.

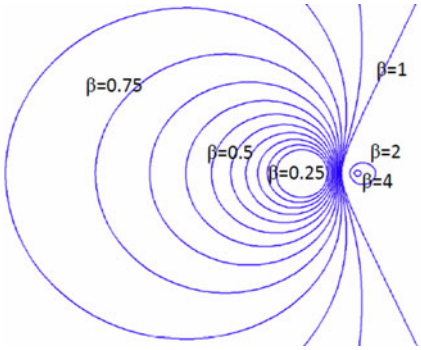


Fig. 3. Bisectors. ($\beta = w_Q/w_O$).

2.3 Analysis on Geometric Properties

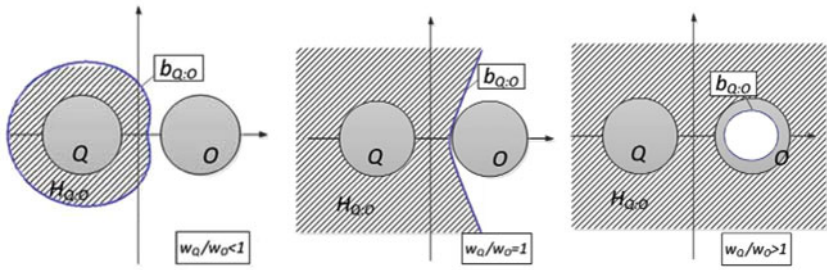
The weighted imprecise bisectors and half spaces are the building blocks from which Voronoi cells constructed.

Bisector. Consider objects Q and O with centers $c_Q = (x_Q, y_Q)$ and $c_O = (x_O, y_O)$, respectively. By substitution, the bisector can be written as:

$$\frac{\sqrt{(y - y_Q)^2 + (x - x_Q)^2} - r_Q}{w_Q} = \frac{\sqrt{(y - y_O)^2 + (x - x_O)^2} + r_O}{w_O}.$$

This equation can be expanded into a fourth-order polynomial. Rather than showing the derivation, we plot bisectors in Fig. 3 when varying the weights of the two objects. The ratio of the weights is given by $\beta = w_Q/w_O$. We observe that the bisector moves from Q towards O as β increases. In particular: 1) if $\beta < 1$, the bisector is a heart-shaped curve; 2) if $\beta = 1$, the bisector is a hyperbola; 3) if $\beta > 1$, the bisector is an ellipse-shaped curve, which is a forth-order polynomial curve. To avoid ambiguities, we call it a *quasi-ellipse*.

Half Space. We show half spaces according to different "β" cases in Fig. 4. They are given by the dashed regions. When $\beta < 1$, the half space is a heart-shaped, closed region. When



(a) $\beta < 1$ (b) $\beta = 1$ (c) $\beta > 1$

Fig. 4. Half space.

$\beta = 1$, the half space is the region enclosed by the hyperbola (Fig. 4b). When $\beta > 1$, the half space equals the domain space D save the region enclosed by a quasi-ellipse (Fig. 4c). We assume domain D is bounded so that $H_{Q:O}$ is also bounded. Otherwise, $H_{Q:O}$ can be an infinite region when $\beta \geq 1$.

2.4 Preliminaries—Geometric Representation

As background for describing the geometric representation of the Voronoi cells proposed in this paper. We first review the bisector for UV-cells [19] that is the special case of the bisector of a weighted imprecise Voronoi cell.

Bisector for non-weighted imprecise objects [19]. ($\beta = 1$) Let $f_0 = (f_x, f_y)$ be the middle point of the centers of Q and O, c_Q and c_O ($f_0 = \frac{c_Q + c_O}{2}$). Let $\cos\theta = \frac{c_O.x - c_Q.x}{|c_Q, c_O|_E}$ and $\sin\theta = \frac{c_O.y - c_Q.y}{|c_Q, c_O|_E}$. After substitution, the bisector equation is:

$$\frac{x_\theta^2}{a^2} - \frac{y_\theta^2}{b^2} = 1, \tag{5}$$

where

$$a = \frac{r_Q + r_O}{2},\ b = \sqrt{c^2 - a^2},\ \text{and}\ c = \frac{|c_Q, c_O|_E}{2};$$
$$\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \Theta \cdot \begin{bmatrix} x - f_x \\ y - f_y \end{bmatrix}\ \text{and}\ \Theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{6}$$

Eq. (5) is a hyperbola equation with c_Q and c_O as the foci, rotated anticlockwise by θ, and with the middle of their centers translated onto the origin. Bisector $b_{Q:O}$ is the half of the hyperbola closer to O. Then, we show the coordinate system transformation.

***θ*-space.** We call the new coordinate system obtained through translation Δ ($\Delta = (f_x, f_y)$ in the above example) and anticlockwise rotation by θ the θ-space [25]. Since it is an orthogonal coordinate transformation, the distances in the original space are preserved in the θ-space. For any point p in the original space, the corresponding point in θ-space is $p_\theta = (x_\theta, y_\theta)$. We define two operators that implement the transformation between the two spaces

$$p_\theta^T = f_{\theta,\Delta}(p) = \Theta \cdot (p^T - \Delta^T),\ \text{where}\ \Theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix},$$
$$p^T = f_{\theta,\Delta}^{-1}(p_\theta) = \Theta^{-1} p_\theta^T + \Delta^T,\ \text{where}\ \Theta^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Notice that if the imprecision is removed, the heart-shaped curve and the quasi-ellipse degenerate into circles, and the hyperbola becomes a straight line. If the weights are removed, the bisector degenerates into a hyperbola [19].

2.5 Cell Construction Framework

We have seen that $H_{Q:O}$ can be concave (Fig. 4), that the bisectors rendering a weighted imprecise Voronoi cell can be higher-order polynomials (Fig. 3), and that weighted imprecise Voronoi cells might have holes (Fig. 2b). These properties render the construction of such Voronoi cells, and it is not clear how to scalably compute a cell given a large database of objects.

To address the challenge, we design a filter-and-refine framework. The filtering step reduces the problem by retrieving a small subset of candidate objects that generate a coarse-grained Voronoi cell (Section 3). The refinement step yields a fine-grained Voronoi cell and candidate objects for defining it by removing false positives from the result of the filtering step (Section 4). The framework applies to both π-cells and ψ-cells. The input is an object if building a π-cell, and a set of objects if building a ψ-cell. We show how the framework accommodates ψ-cells in Section 5.

3 Filtering

3.1 Overview

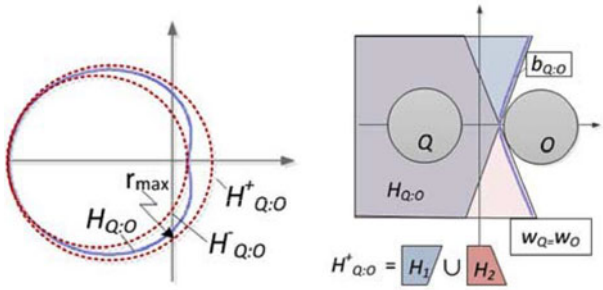
We assume that an aggregate R-tree is built on the set of weighted imprecise objects. Each index entry has a weight that is the largest one among all objects in its subtree.

The IR-tree [18] can be viewed as an aggregate R-tree if we consider the textual relevance information stored in index nodes as virtual weights. The textual relevance, or weight, of an index node is calculated at runtime and is is guaranteed to be higher than that of any descendant entry. The correctness follows from the monotonicity property [18]. For brevity, we use simply the term R-tree for such a tree.

The process of constructing a Voronoi cell follows the definition, by initializing the cell as the entire domain and then progressively refining it with half spaces w.r.t. other objects. When traversing an index node in the filtering step, it is determined whether any other object in a subtree of a node entry can possibly refine the cell.

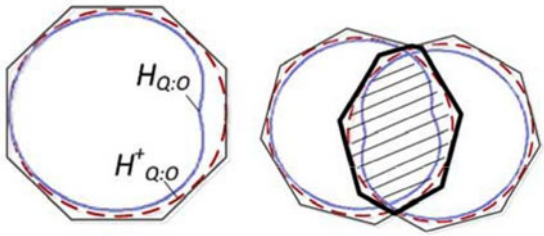
Consider a π-cell as an example. Let Φ_Q be the currently refined π-cell, and let C be the set of objects considered so far. We have:

$$\Phi_Q = \bigcap_{O \in C} H_{Q:O} \supseteq \bigcap_{O \in \mathbb{O} - \{Q\}} H_{Q:O} = V_Q$$



(a) $H^+_{Q:O}$ and $H^-_{Q:O}$ ($\beta < 1$) (b) $H^+_{Q:O}$ ($\beta = 1$)

Fig. 5. Outer region $H^+_{Q:O}$.



(a) $P_f(H_{Q:O})$ ($f = 8$) (b) Polygon Intersection

Fig. 6. Examples of externally tangent polygons.

Clearly, Φ_Q covers the Voronoi cell V_Q. As only some objects have been considered, we call Φ_Q a partial cell. An unvisited R-tree subtree can be filtered if each half space of each object in the subtree totally covers Φ_Q and thus not refine Φ_Q. The R-tree is explored until all objects have been considered. The key difficultly lies in the complex nature of the curves that define the half space.

We study how to best approximate complex half spaces with simple geometries in order to construct partial Voronoi cells (Section 3.2). We design an early stop condition that avoids exploring subtrees with only non-qualifying objects (Section 3.3). We devise a filtering algorithm for π-cells (Section 3.4).

3.2 Partial Cell Construction

We approximate half spaces in Section 3.2.1, and describe how to construct a partial cell in Section 3.2.2.

3.2.1 Half Space Approximation

Given two objects, Q and O, we find that when $w_Q \neq w_O$, the complex shape of $H_{Q:O}$ can be approximated by two circular regions, namely an *inner region* $H^-_{Q:O}$ and an *outer region* $H^+_{Q:O}$. An example of an outer region when $\beta < 1$ is shown in Fig. 5a. For the case $\beta = 1$, where the bisector is a hyperbola, we can offset its asymptotes to build the outer region. The outer region $H^+_{Q:O}$ is the union of two trapezoids H_1 and H_2, each of which is a convex region enclosed by an offset asymptote (see Fig. 5b).

Lemma 1. *The outer regions $H^+_{Q:O}$ are represented by equations as follows.*

$$\beta < 1 : \odot(f^{-1}_{\theta,\Delta}(p_1 + [r_{max}, 0]), r_{max})$$

$$\beta > 1 : D - \odot\left(f^{-1}_{\theta,\Delta}\left(\frac{p_5 + p_6}{2}\right), \frac{|p_5, p_6|_E}{2}\right)$$

$$\beta = 1 : H_1 \cup H_2, \quad \text{where}$$

$$H_1 = \left\{ p \,|\, \begin{bmatrix} b \\ -a \end{bmatrix}^T \cdot \left(f_{\theta,\Delta}(p) - \begin{bmatrix} a \\ 0 \end{bmatrix} \right) \leq 0 \right\}, \tag{7}$$

$$H_2 = \left\{ p \mid \begin{bmatrix} b \\ a \end{bmatrix}^T \cdot \left(f_{\theta,\Delta}(p) - \begin{bmatrix} a \\ 0 \end{bmatrix} \right) \leq 0 \right\}. \tag{8}$$

Here, points p_1, p_5, and p_6 are feature points for half spaces, as detailed in the appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2015.2464804, and r_{max} is the radius of the outer region $H^+_{Q:O}(\beta < 1)$, as also detailed in the appendix, available in the online supplemental material.

Proof. (*sketch*) To verify the correctness of the equations, consider the case $\beta < 1$ as an example. We transform the equation in Lemma 1 into the θ-space and then combine it with the equation of the bisector as plotted in Fig. 5a. Considering the second quadrant, if for every x-value, the y-value of the equation in Lemma 1 is never less than that of the bisector equation, we conclude that $H^+_{Q:O}$ contains $H_{Q:O}$ in that quadrant. Similar arguments can be made for other quadrants, from which the correctness follows. □

With the half space approximation, we obtain simple geometric shapes like circles and polygons. But intersecting them as needed to obtain partial cells is still problematic because the intersection of circles are irregular shapes and the polygon of $H^+_{Q:O}(\beta = 1)$ is concave.

3.2.2 Partial Cell Construction

We consider the circular outer region case ($\beta < 1$) and the polygonal outer region case ($\beta = 1$). We skip the $\beta > 1$ case because a quasi-ellipse "hole" has little effect on a partial cell's contour.[3]

For the $\beta < 1$ case, we adopt existing techniques [13] to build an f-sided polygon that is externally tangent with it (Fig. 6a). The polygon is guaranteed to be convex, so that the intersection of two such polygons is also convex. There exists a trade-off between the number of sides f for the polygon and the approximation accuracy. According to our experiments, we set f to 16, which achieves a good tradeoff between accuracy and efficiency. Very accurate approximation is not necessary because a less accurate approximation only slightly increases the false positives that are efficiently eliminated in the refinement phase.

In the $\beta = 1$ case, the polygonal outer region are decomposed into two convex polygons H_1 and H_2 (Fig. 5b), each of which is intersected with partial cell Φ_Q separately, upon which their union is formed. This is correct because

$$\Phi_Q \cap H^+_{Q:O} = \Phi_Q \cap (H_1 \cup H_2) = (\Phi_Q \cap H_1) \cup (\Phi_Q \cap H_2).$$

3. Although it incurs false positives, these can be reduced by the early stop condition (Section 3.3) and are eliminated during refinement (Section 4).

Here, the resulting Φ_Q can be concave. So, each time we update partial cell Φ_Q with an $H_{Q:O}$, we build the f-sided convex hull for the updated Φ_Q if it is not convex.

For both the circular and polygonal outer region cases, we adopt polygon approximation to facilitate the computation. We use Φ^+ to denote the approximated partial cell to distinguish it from Φ_Q.

3.3 Access Order and Early Stopping

3.3.1 Early Stop Condition

We choose an appropriate order for accessing R-tree entries in order to minimize the number of nodes accessed, thus reducing the computational cost. Specifically, we access entries according to the *minimum center-border distance*.

Definition 5. ***Minimum center-border distance.*** *Given two objects Q and O, their* minimum center-border distance *is the minimum euclidean distance from Q's center c_Q to the bisector $b_{Q:O}$:*

$$|Q, O|_{bd} = min_{p \in b_{Q:O}} |p, c_Q|_E.$$

We give a simple equation for computing $|Q, O|_{bd}$ in Section 3.3.2. Intuitively, the object with smaller minimum center-border distance tends to contribute more to refining Φ_Q^+. Also, it enables an *early stop condition* that allows us to disregard entries with large minimum center-border distances.

Lemma 2. ***Early stop condition.*** *Let Φ_Q^+ be a partial cell for object Q. An entry O can be disregarded if:*

$$|c_Q, \Phi_Q^+|_E^u < |Q, O|_{bd.}$$

Proof. Let us call $|c_Q, \Phi^+|_E^u$ the max_border distance, as it is the maximum euclidean distance from Q's center to partial cell Φ^+. Then, if O's mininum center-border distance exceeds the max_border distance, O cannot refine the partial cell. □

With Lemma 2, we can design an algorithm in the style of best-first search [26], that accesses entries according to the minimum center-border distances. All that remains is to calculate minimum center-border distances.

3.3.2 Minimum Center-Border Distance Calculation

We cover how to derive the minimum center-border distance for both objects and R-tree entries.

$|Q, O|_{bd}$ for objects. We can calculate the minimum center-border distance based on the feature points in the θ-space. The equation is shown in Lemma 3.

Lemma 3. $|Q, O|_{bd} = \frac{w_Q |c_Q, c_O|_E + w_Q r_O + w_O r_Q}{w_O + w_Q}$.

Proof. We transform the bisector equation into θ-space (Fig. 13), which is distance preserving. From the figure, the min center-border distance is $|c_Q, p_2|_E$ when $\beta < 1$, $|c_Q, p_4|_E$ when $\beta = 1$, and $|c_Q, p_5|_E$ when $\beta > 1$. After expanding the equations, they follow the same form, $\frac{w_Q |c_Q, c_O|_E + w_Q r_O + w_O r_Q}{w_O + w_Q}$. □

$|Q, N_R|_{bd}$ for R-Tree Entries. For an R-tree node N_R with weight w_h, we can derive its minimum center-border distance by creating a virtual object $\widehat{N_R}$, with its nearest point to Q's center as the imprecise region and with weight w_h. Formally, $\widehat{N_R} = (p, w_h)$, where $p = argmin_{q \in N_R}(|c_Q, q|_E)$. The virtual object $\widehat{N_R}$ is constructed in such a way that if $\widehat{N_R}$ is rejected by the early stop condition, all objects in its subtree must also be rejected. Then, all that remains to be shown is that the minimum center-border distance of $\widehat{N_R}$ is no larger than those of its children. We call this the *monotonic property*. It is formulated in Lemma 4 and proven in the appendix, available in the online supplemental material.

Lemma 4. ***Monotonic property.*** *Given object Q and two entries N_1 and N_2, if N_2 is a subtree of N_1 then:*

$$|Q, N_1|_{bd} \leq |Q, N_2|_{bd}.$$

Next, we summarize the techniques devised so far and present the filtering algorithm.

3.4 Filtering Algorithm

The filtering phase is shown as Algorithm 1. Initially, the partial cell is set to be the domain, and max_border is infinite. The R-tree entries are accessed in a best-first manner. At each iteration, the partial cell is updated with convexification (Section 3.2.1), and the algorithm terminates if the early stop condition is met (Section 3.3).

Algorithm 1. Filtering

```
Data: Query object Q, R-tree T
Result: Partial Cell Φ_Q^+, candidate object set C
Φ_Q^+ ← D; C ← ∅; max_border (|Q.c, Φ^+|_E^u) ← ∞
H is a min heap sorted by the min center-border distance;
H.enheap(T.root);
while H is not empty do
  e ← H.deheap();
  if max_border < |Q, e|_bd then
    break; ▷early stop, Lemmas 2, 3, 4
  if e is a leaf-node then
    for each object O ∈ e do
      if w_Q ≤ w_O then
        Φ_Q^+ ← cvx(Φ_Q^+ ∩ H_{Q:O}^+); ▷Lemma 1
        Update max_border with |Q.c, Φ_Q^+|_E^u;
        Add O to C;
      else if w_Q > w_O then
        Add O to C;
  else
    for each child node e_i ∈ e do
    H.enheap(e_i);
  return Φ^+ and C;
```

Complexity analysis. Determining whether two f-sided polygons intersect and computing the intersection takes $O(f)$ time. Then, given C candidates, we need $O(Cf)$ time for calculating Φ_Q^+. Next, the number of candidates inserted into the heap is $O(C)$, and each insertion costs $O(log\ C)$. The traversal of the R-tree takes $O(log\ n)$ if there are n objects. In total, the filtering step costs $O(log\ n + Clog\ C + Cf)$.

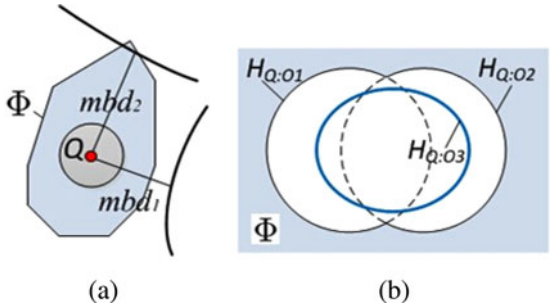


Fig. 7. False positives in filtering.

Discussion. There are three sources of false positives in the result returned by the algorithm. First, there are three levels of outer approximations that enlarge a partial cell, including the outer region, the circumscribed polygon, and the convexification. Second, the early stop condition rejects objects with larger minimum center-border distances. But objects that have passed the early stop condition do not necessarily contribute to a partial cell. In Fig. 7a, mbd_1 is smaller than mbd_2, but mbd_1's corresponding half space does not contribute to the partial cell while mbd_2's half space does. Third, objects in the $\beta > 1$ case need to be further refined. In Fig. 7b, O_1, O_2, and O_3 are in C because their weights are smaller than that of Q, and because their half spaces overlap with the partial cell. Object O_3 can be disregarded because its half space $H_{Q:O_3}$ covers the union of $H_{Q:O_1}$ and $H_{Q:O_2}$ and thus does not contribute to the partial cell.

4 Refinement

4.1 Overview

The filtering phase returns a coarse-gained partial Voronoi cell and a set of candidate objects containing false positives. False positives do not affect correctness, but are unattractive from a performance perspective. For example, in a location-based service where safe zones [13], [27] are used for communication cost optimization, false positives result in extra transmission costs. We proceed to define the notion of candidate objects, which offers an alternative representation for Voronoi cells.

Definition 6. *The **candidate objects** C_Q is a set of objects, such that:*

$$V_Q = \bigcap_{O \in \mathbb{O} - \{Q\}} H_{Q:O} = \bigcap_{O \in C_Q} H_{Q:O}.$$

A trivial set of candidate objects is thus all objects in $\mathbb{O}$. The optimal set of C_Q is the smallest set that can render V_Q, denoted as C^*. In other words, we cannot get V_Q if any object $O^* \in C^*$ is removed. Formally,

$$V_Q = \bigcap_{O \in C^*} H_{Q:O}, \quad \text{and} \quad V_Q \subset \bigcap_{O \in C^* - \{O^*\}} H_{Q:O}. \tag{9}$$

It is challenging to determine C_Q^* for imprecise objects. This has been observed in past studies [19], [20], [22], where no solutions are provided, where only coarse approximations of Voronoi cells are provided, and where candidate objects are derived without guarantees. We design two refinement algorithms by progressively expanding a quad-tree to refine Φ_Q^+ and C simultaneously. The first method, δ-refinement, controls the quad-tree expansion by a threshold δ on the sizes of leaf nodes. With a sufficiently small δ, the result (Φ_Q^+, C_Q) can approach (V_Q, C_Q^*) arbitrarily closely. The second algorithm, refinement*, is a parameter-free method that guarantees obtaining an exact C^*. But the algorithm does not produce the Voronoi cell's region.[4]

We study the cell-region relationship in Section 4.2 and cover the refinement algorithms in Section 4.3.

4.2 Cell-Region Relationship

We introduce a set of operators for deciding whether a region is located inside or outside a Voronoi cell. Based on that, we can recursively decompose a region into subregions and test if a subregion is part of the cell, based on which we design the refinement algorithm. We start by providing a means of deciding whether a region is inside a half space.

Lemma 5. *Given objects Q and O and a region R, we have:* $\max_{p \in R} |p,Q|^l < \min_{p \in R} |p,O|^u \Rightarrow R \subseteq H_{Q:O}$

Proof.

$$\left.\begin{array}{l} |p,Q|^l < \max\limits_{p' \in R} |p',Q|^l \\ \min\limits_{p' \in R} |p',O|^u < |p,Q|^u \end{array}\right\} \Rightarrow \forall p \in R(|p,Q|^l < |p,O|^u) \Rightarrow R \subseteq H_{Q:O}.$$

□

Lemma 6. *Given objects Q and O and a region R, we have:*

$$|R,Q|^l \geq |R,O|^u \Rightarrow R \subseteq \overline{H_{Q:O}}.$$

Lemmas 5 and 6 can be used to decide whether or not a region R is part of a Voronoi cell, as shown in Lemmas 7 and 8. Proofs are omitted for brevity.

Lemma 7. ***Region acceptance.*** *Given an object Q and region R, we have: $\forall O \in \mathbb{O}\ (R \subseteq H_{Q:O}) \Leftrightarrow R \subseteq V_Q$.*

Lemma 8. ***Region rejection.*** *Given an object Q and region R, we have: $\exists O \in \mathbb{O}\ (R \in \overline{H_{Q:O}}) \Leftrightarrow R \cap V_Q = \emptyset$.*

Since Lemmas 5 and 6 provide sufficient, but not necessary, conditions for determining the cell-region relationship, there might be an "undetermined" case. Our algorithm solves that by decomposing region R into subregions, repeatedly applying Lemmas 7 and 8. Imaging a limit case, where a subregion is infinitely small, Lemma 5 is equivalent to testing whether a point is in the half space with Definition 2. Our refinement algorithm has two variants for determining whether the subregion is small enough.

4.3 Refinement Algorithms

Both refinement algorithms are based on a quad-tree. The algorithms take the output of the filtering phase, a partial cell Φ_Q^+ and a candidate set C, as input. Each node of the quad-tree is a three-tuple: (region, child pointers, C). The

4. By combining the two techniques, we can derive C^* and a δ-approximated cell simultaneously. We omit this for brevity.

region of the quad-tree root is set to be the minimum bounding square of Φ_Q^+. Its set C is that obtained from the filtering step. The pointers refer to its four children, each of which occupies a quadrant. Quad-tree nodes are then split until they meet the stopping conditions. The two refinement algorithms use different stopping conditions.

δ-refinement. First, δ-refinement is shown in Algorithm 2. We use Lemma 6 to trim objects passed from a parent node N_Q to a children node. If the parent node is inside half space $H_{Q:O}$, so are its children. This means that O can be pruned from N_Q's candidates. A region can be marked as part of V_Q if it is found to be inside V_Q; and it can be disregarded if it is outside V_Q. If a region is marked as undetermined, we can split it. If a region overlaps with V_Q's contour, it is neither inside nor outside V_Q, which then incurs infinite splitting. To avoid that, we set a threshold δ for the quad-tree leaf node size. The splitting stops either if a leaf node's region is marked as part of V_Q or its size is smaller than δ.

The quad-tree leaf nodes output by Algorithm 2 approximate V_Q, and the gap between the approximated V_Q and the exact V_Q is within δ. The union of the undetermined regions' candidates is returned as C_Q because the regions are on the contour, and their candidates help in rendering V_Q. The derived C_Q must be a superset of C^*. But the quality of C_Q is also controlled by δ. For a false positive object $O \in C_Q - C^*$, the weighted distance error depends on the form of the weighted distance function, the euclidean distance error being at most $\sqrt{2}\delta$.

Algorithm 2. δ-Refinement(Q, N_Q)

Data: Query object Q, Quad-tree node N_Q
Result: Voronoi cell V_Q, Result set C_Q
Global variables: δ, Result set $C_Q = \emptyset$, Partial cell Φ^+;
if $area(N_Q) < \delta$ **then**
  $C_Q \leftarrow C_Q \cup \{O \in N_Q.C\}$; return;
else
  if $N_Q.region \cap \Phi^+ = \emptyset$ **then**
    return;
  Split N_Q into 4 child nodes $\{N_q\}$;
  For each N_q, let $N_q.C \leftarrow N_Q.C$, then clear $N_Q.C$;
  for each $N_q \in N_Q$ **do**
    $flag \leftarrow true$;
    if $|Q, N_q|^l > min\{|O, N_q|^u\}$ **then**
      continue; ▷(Lemma 6, 8)
    else
      for each object $O \in N_q.C$ **do**
        if $\max_{p \in N_q} |p, Q|^l < \min_{p \in N_q} |p, O|^u$ **then**
          Remove O from $N_q.C$; ▷(Lemma 5)
        else
          $flag \leftarrow undetermined$;
      if flag is true **then**
        Mark N_q as a part of V_Q; ▷(Lemma 7)
      else if *flag is undetermined* **then**
        δ-Refinement(Q, N_q);

Refinement.* The refinement* algorithm guarantees to find the optimal candidate set C^*. The algorithm is obtained by replacing Algorithm 2's stopping condition with $N_Q.C = 1$ or $N_Q.C \subseteq C^*$ and rewriting Algorithm 2 to proceed in a breath-first manner instead of depth-first (e.g., with a queue).

Intuitively, if we consider V_Q as a curved polygon with edges and vertices, the first condition covers the case where a region overlaps with an edge, and the second condition covers others cases, including the one where a region overlaps with a vertex. The correctness of the condition $N_Q.C = 1$ follows from Lemma 15 in the appendix, available in the online supplemental material. The condition $N_Q.C \subseteq C^*$ eliminates unnecessary comparisons when a node's candidates are already contained in the current C^*. If a node contains more than one candidate, we split it into subnodes and repeatedly apply Lemma 5 to eliminate candidates. However, the splitting does not work, if a node N_v overlaps with a vertex, because none of the two candidate objects constituting the vertex can be further removed through splitting. Notably, since the vertex is an intersection of two edges, there must exist two nodes, N_{e_1} and N_{e_2}, overlapping with the two edges, respectively. So, the splitting of N_v stops after the splitting on N_{e_1} and N_{e_2} stop. That is why we use breadth-first search for refinement*. In summary, the second condition ensures that the splitting always stops.

5 Construction of Ψ-Cells

5.1 Filtering for ψ-Cells

For ψ-cells, the input is a set S of query objects $\{Q_i\}$. An intuitive but inefficient algorithm is to invoke Algorithm 1 multiple times for the filtering phase. According to the definition of ψ-cell, we can apply Algorithm 1 to each $Q_i \in S$ to get $\{(\Phi_{Q_i}^+, C_i)\}$. Then, we can combine them into (Φ_S^+, C), where $\Phi_S^+ = \cap_{Q_i \in S} \Phi_{Q_i}^+$ and $C = \cup_{Q_i \in S} C_i$. Here, Φ_S^+ is no larger than any $\Phi_{Q_i}^+$ because C contains more candidates than C_i does.

This algorithm suffers from two disadvantages. It requires multiple index traversals, which can be costly and is redundant, since the index entries accessed for closely located Q_is are similar. Further, some candidates in the C_i are added unnecessarily to the candidate set, which increases the refinement workload. We thus adopt a different approach to extending Algorithm 1 for ψ-cells.

Partial ψ-cell construction. We rewrite the definition of ψ-cell (Definition 4) as:

$$V_S = \bigcap_{Q_i \in S} V_{Q_i} = \bigcap_{Q_i \in S} \cap_{O \in \mathbb{O}} H_{Q_i:O} = \bigcap_{Q_i \in S, O \in \mathbb{O}} H_{Q_i:O}$$

From the equation, a ψ-cell is essentially the intersection of half-spaces constituted by each (Q_i, O) pair. Then, in Algorithm 1, for each visited leaf entry e, we change the loop (line 9) into a dual loop that checks each pair $(Q_i, O)_{Q_i \in S, O \in e}$, applies the polygon approximation techniques (Section 3.2.1) to their half spaces, and computes the partial ψ-cell.

Access order and early stopping. As the input is now an object set S, we use the *minimum set center-border distance* between S and an entry e, defined as $|S, e|_{bd} = min_{Q_i \in S}(|Q_i, e|_{bd})$. Thus, $|S, e|_{bd}$ returns the minimum value of all possible minimum center-border distances for $Q_i \in S$ and e.

Lemma 9. ***Early stop condition (ψ-cell).*** *Given a partial cell Φ_S^+ with max_border distance $max_{Q_i \in S}\{|c_{Q_i}, \Phi_S^+|_E^u\}$, an entry e can be rejected if $|S, e|_{bd}$ exceeds the max_border distance.*

The lemma and the monotonicity property can be proved by techniques similar to those used in Section 3.3.

5.2 Refinement of ψ-Cells

A ψ-cell contains all points of the domain that take each object in S as a possible nearest neighbor. We can improve Lemmas 5, 6, 7, and 8 for the ψ-cell problem as follows.

Lemma 10. *Given a set S of query objects $\{Q\}$, an object O, and a region R, we have:*

$$\max_{p\in R, Q\in S} |p,Q|^l \leq \min_{p\in R} |p,O|^u \Rightarrow \forall Q \in S\ (R \subseteq H_{Q:O}).$$

Lemma 11. ***Set-region acceptance.*** *Given a set S of query objects $\{Q\}$, and a region R, we have:*

$$\forall Q \in S\ \forall O \in \mathbb{O} - \{Q\}\ (R \subseteq H_{Q:O}) \Leftrightarrow R \subseteq V_S.$$

Lemma 12. ***Set-region rejection.*** *Given a set S of query objects $\{Q\}$, and a region R, we have:*

$$\exists O \in \mathbb{O} - S\ \exists Q \in S\ (R \subseteq \overline{H_{Q:O}}) \Leftrightarrow R \cap V_S = \emptyset.$$

Here, Lemma 12 is supported by Lemma 6 to decide if a region is in the half space complement. Also, it is easy to see that the space dominance monotonicity holds for ψ-cells, that is: $R \subseteq V_S \Rightarrow r \subseteq V_S (r \subseteq R)$. Then, Algorithm 2 can be used for handling ψ-cells if we replace the equations in Lemmas 5, 7, and 8 with those in Lemmas 10, 11, and 12, respectively, and add an inner loop for handling every $Q \in S$ for the loop in line 14.

6 Extension to Additively Weighted Distances

We have so far considered Voronoi cells with multiplicatively weighted distance functions. Another common way of combining two heterogeneous domains, i.e., spatial and non-spatial domains, is by using additively weighted distance functions [15], [18], denoted by the "+" case. The distance function is formulated as:

$$|p,O| = |p,O|_E - w_O \quad (\text{"+" case}).$$

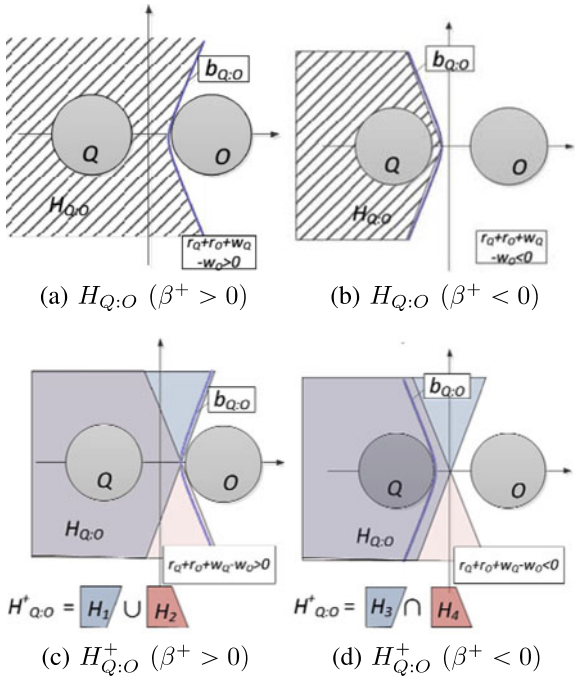
We can obtain the additively weighted imprecise distance $|p,O|$ by substituting the euclidean distance in the above equation by the imprecise euclidean distance in Eq. (2). Based on that, we define the notion of bisector.

Weighted imprecise bisector ("+" case). The bisector $b_{Q:O}$ for the "+" case consists of points $p = (x, y)$ satisfying:

$$\sqrt{(y-y_Q)^2 + (x-x_Q)^2} - r_Q - w_Q = \sqrt{(y-y_O)^2 + (x-x_O)^2} + r_O - w_O.$$

We transform the equation into θ-space and derive the bisector equation similar to Eq. (5), where:

$$a = \frac{|r_O + r_Q + w_Q - w_O|}{2}, b = \sqrt{c^2 - a^2}, c = \frac{|c_Q, c_O|_E}{2}.$$



(a) $H_{Q:O}$ $(\beta^+ > 0)$ (b) $H_{Q:O}$ $(\beta^+ < 0)$

(c) $H^+_{Q:O}$ $(\beta^+ > 0)$ (d) $H^+_{Q:O}$ $(\beta^+ < 0)$

Fig. 8. $H_{Q:O}$ and $H^+_{Q:O}$ for "+" case.

For computing Voronoi cells, everything we have said so far also works for the "+" case, except where we provide modifications in the following. In particularly, the refinement phase (Algorithm 2) is not specific to a particular weighted distance functions. Then, all that remains is to handle minimum center-border distance calculation and half space approximation for the filtering phase (Algorithm 1).

Half space approximation. The half space enclosed by the bisector is illustrated in Figs. 8a and 8b. By viewing the bisector equations in the hyberbola form, we can categorize them into two cases, depending on the value of $\beta^+ = r_Q + r_O + w_Q - w_O$. Specially, if $\beta^+ = 0$, the hyperbola degenerates into a straight line. Under the special condition $|c_Q, c_O|_E < r_Q + r_O$, the bisector equation does not have any roots, meaning that the half space is the entire domain. For $\beta^+ \neq 0$, we can use asymptotes to approximate their half spaces. If $\beta^+ > 0$, the bisector is the hyperbola branch closer to O. The half space (Fig. 8a), can be approximated by $H_1 \cup H_2$ (Fig. 8c). H_1 and H_2 are formulated by Eqs. (7) and (8), respectively. If $\beta^+ < 0$, the bisector is the hyberbola branch closer to Q. The half space (Fig. 8b), can be approximated by $H_3 \cap H_4$ (Fig. 8d), as given in Eqs. (10) and (11):

$$H_3 = \{p \mid [b-a] \cdot f_{\theta,\Delta}(p) \leq 0\}, \tag{10}$$

$$H_4 = \{p \mid [b \quad a] \cdot f_{\theta,\Delta}(p) \leq 0\}. \tag{11}$$

Again, we can derive the minimum center-border distances for the "+" case as shown in Table 3.

To summarize, the "+" case yields different half spaces and thus different minimum center-border distances. To apply the filtering algorithm (Algorithm 1), it is necessary and sufficient to replace the minimum center-border distances (line 6) with Table 3's equations and replace the

TABLE 3
$|Q, O|_{bd}$ in "+"-Case

case	equation	expanded form
$\beta^+ \geq 0$	$\lvert c_Q, p_4\rvert_E$	$c + a$
$\beta^+ < 0$	$\lvert c_Q, p_7\rvert_E$	$c - a$

convexification procedure (line 11) with the one shown in Figs. 8c and 8d.

Discussion. We have demonstrated how it is possible to support a different kind of distance function by replacing some of the building blocks in the provided framework with new building blocks while reusing the overall framework and many of the existing building blocks. Other distance functions such as exponential distance, logarithmic distance, and power distance can be potentially supported. Theoretically, they are considered as "trivial generalization," as the half spaces constructed are the same as those for euclidean distances [17]. Thus, further discussion is omitted.

7 Empirical Study

Section 7.1 covers the experimental setup, where default parameters are given in bold. Section 7.2 presents findings.

7.1 Settings

Synthetic dataset. We use Theodoridis et al.'s data generator[5] to obtain sets of 100K, **500K**, and 1M objects that are uniformly distributed in a 10K $\times$ 10K space. By default, imprecise regions are circular regions with a radius of 15 units[6] and an associated weight ranging from 0 to **20**. Synthetic data is used for studying scalability.

Real datasets. We use two series of datasets. For experiments with textual attributes, we use two real geolocated Twitter datasets, namely USA[7] and WW[8] with 140,749 and 68,119 objects, respectively. For each tweet, we model the GPS inaccuracies associated with its location [28] by introducing a circle with radius from 0 to **20** units around its location. We use the IR-tree for indexing.

We set δ to **50** by default and also vary δ in experiments. To gain insight into different aspects of our proposals, we consider four competitors, FRδ, FR$*$, Rδ, and R$*$, where prefix F refers to the filtering phase and suffixes Rδ and R$*$ refer to δ-refinement and refinement$*$, respectively. Thus, FRδ and FR$*$ represent filtering with different refinement algorithms; Rδ and R$*$ refer to different refinement algorithms without filtering. Each reported value is the average of 50 runs. All our programs were implemented in Java and run on a PC with an Intel dual core 3.30 GHz processor and 4 GB RAM.

5. http://www.chorochronos.org/sites/default/files/algorithms/SpatialDataGenerator.zip
6. Default imprecision setting follows statistics of DGPS's accuracy: http://en.wikipedia.org/wiki/Differential_GPS.
7. http://www.ark.cs.cmu.edu/GeoText/
8. http://www.ntu.edu.sg/home/gaocong/data/toisdata.zip

7.2 Results

7.2.1 Results on π-Cells

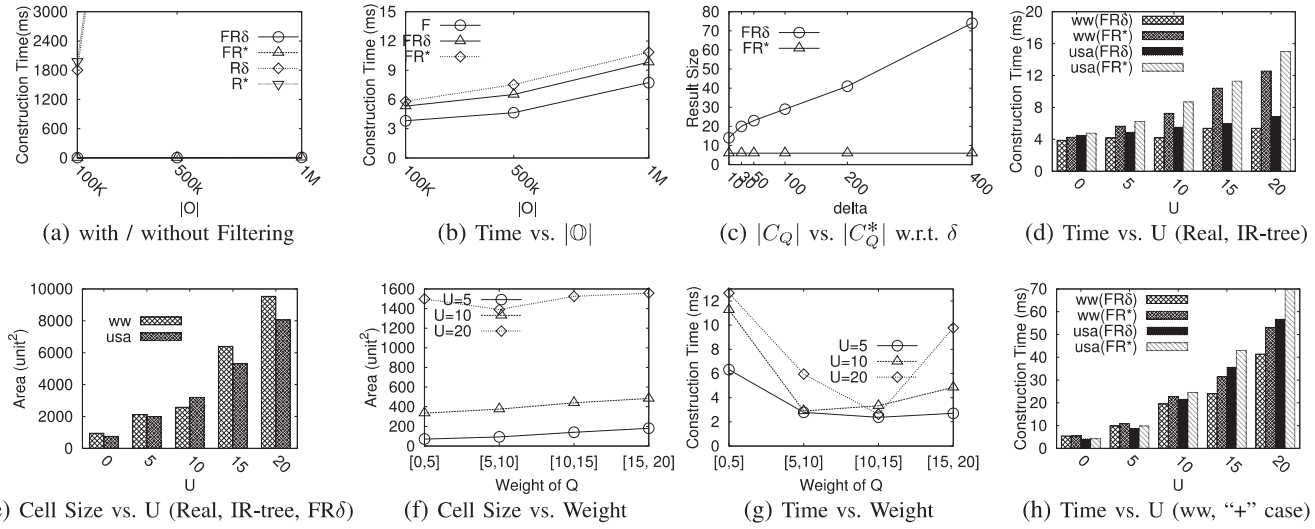
Results for π-cells are shown in Fig. 9. We first consider the effectiveness of the filtering algorithm in Fig. 9a. If there is no filtering, i.e., when using Rδ and R$*$, we use the domain space and the entire dataset $\mathbb{O}$ as input to the refinement algorithms. The results show that the filtering is essential, since the performance degenerates drastically (about 300 times slower) with no filtering. We magnify the performance of the integrated framework, i.e., FRδ, FR$*$, in Fig. 9b. The curve F represents only the filtering time. The results show that: 1) our algorithm is efficient at computing π-cells, using less than 15 ms on all synthetic data; 2) the filtering and refinement time increase moderately; 3) refinement$*$ is a bit slower than δ-refinement. We proceed to examine the effectiveness of FRδ and FR$*$ in terms of the quality of candidate objects $|C_Q|$ in Fig. 9c. Refinement$*$ is parameter-free and is guaranteed to find the smallest candidate object set C_Q^*; thus, FR$*$ is a horizontal line. In contrast, δ-refinement is sensitive with δ. To approximate C_Q^* well, a small δ is needed, and the recursive depth of Algorithm 2 is increased, which incurs higher refinement time.

Next, we examine the results on spatial tweets indexed by the IR-tree when varying the imprecision. In all tests, as shown in Fig. 9d, the time increases moderately w.r.t. the size of the imprecise region. We also report the size of the π-cell constructed with different imprecision in Fig. 9e. We only report the results of FRδ, since FR$*$ cannot derive a π-cell's shape. The cell size increases w.r.t. the amount of imprecision, which is expected. Notice that $U = 0$ refers to the precise weighted Voronoi cell, whose imprecise region degenerates into a point. The cell size of the precise case is about 20 percent of $U = 15$.

We next study the effect of target object O's weight on performance. From Fig. 9f, we observe that when Q's weight is small, the corresponding (partial) Voronoi cell is small. This is consistent with the plot of Fig. 3 where a smaller w_Q has smaller half spaces. Therefore, their intersection, the Voronoi cell, is also small. Also, larger imprecise regions yield a larger Voronoi cell. Fig. 9g shows the construction time w.r.t. object Q's weight. In all cases, the construction can be done efficiently, i.e., taking less than 13 ms. The construction time is higher when W_Q is either very small or very large. This is because a small w_Q yields a small Voronoi cell, which requires more iterations in the refinement step. On the other hand, a Q with a large w_Q tends to have more candidate objects, which also increases the construction cost. A similar trends can be observed in Fig. 9h, where we use additively weighted distances for spatial tweets.

7.2.2 Results on ψ-Cells

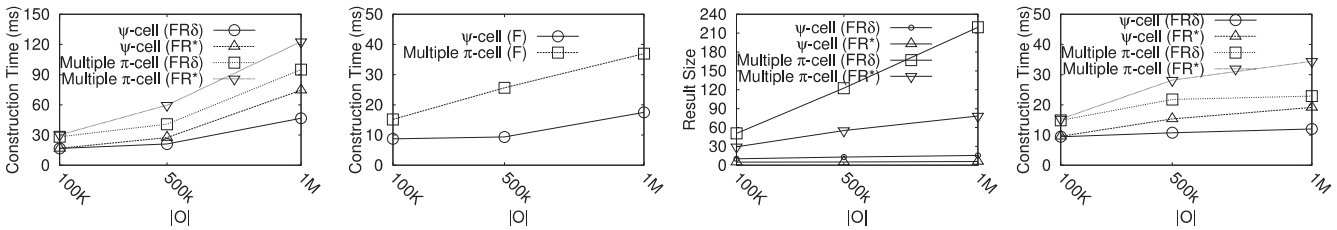
We examine the performance of computing ψ-cells in Fig. 10. A baseline solution for computing a ψ-cell is to: 1) evaluate multiple π-cells' candidate objects separately; 2) use the merged candidate objects for the refinement process. We call this baseline *multiple π-cell* and call our proposed solution *ψ-cell*. Fig. 10a shows the advantages of combing the filtering phases in our ψ-cell solution. Specially, when $|\mathbb{O}| = 500K$, ψ-cell(FR$*$) is three times faster than multiple π-cell(FR$*$). The multiple π-cell solution



(a) with / without Filtering (b) Time vs. $|\mathbb{O}|$ (c) $|C_Q|$ vs. $|C_Q^*|$ w.r.t. δ (d) Time vs. U (Real, IR-tree)

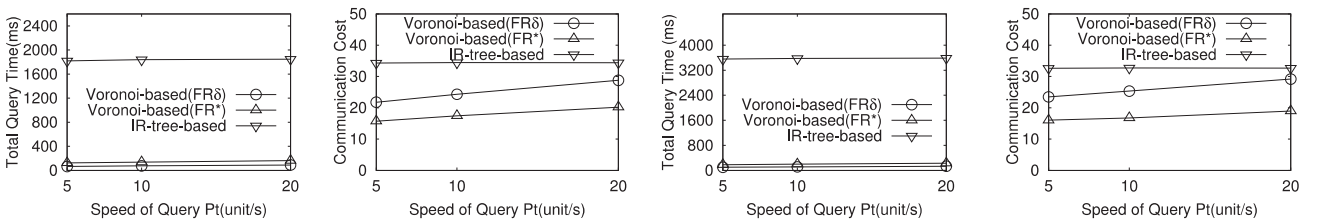
(e) Cell Size vs. U (Real, IR-tree, FRδ) (f) Cell Size vs. Weight (g) Time vs. Weight (h) Time vs. U (ww, "+" case)

Fig. 9. π-cell.



(a) Time vs. $|\mathbb{O}|$ (b) Time(filter) vs. $|\mathbb{O}|$ (c) $|C|$ vs. $|\mathbb{O}|$ (d) Time vs. $|\mathbb{O}|$ ("+" case)

Fig. 10. ψ-cell.



(a) Query time(ms) (ww, Real) (b) Communication cost (ww, Real) (c) Query time(ms) (usa, Real) (d) Communication Cost (usa, Real)

Fig. 11. Proximate VGS tracking (Q).

suffers from: 1) higher processing time due to multiple index traversals (Fig. 10b); 2) consideration of excessive numbers of candidate objects during filtering (Fig. 10c). Also, we report the results on the "+" case in Fig. 10d, where our methods also outperform the baselines.

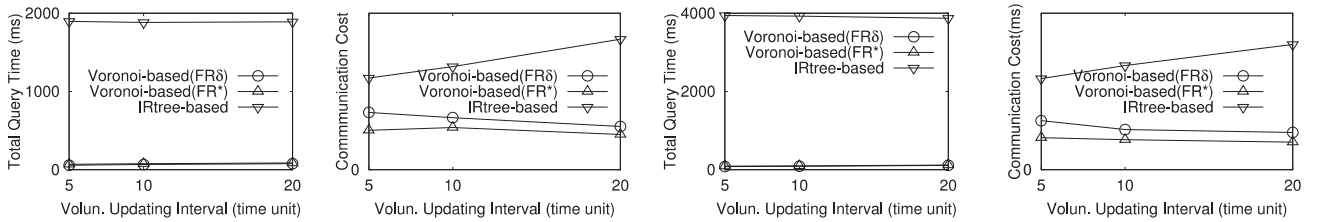
7.2.3 Results on VGS Tracking

We study how the computed Voronoi cell can be used in real applications, e.g., safe zones for tracking volunteers in VGS scenarios. With our method, a moving query first invokes the IR-tree on the server side to get its current PNN result. Then we build the ψ-cell as the safe zone by setting S to the PNN result. As long as the query stays inside the ψ-cell, the PNN answer remains unchanged. Thus the communication cost, represented by objects transferred, is reduced. So is the computation cost. We call this the Voronoi-based method. There is a variant for each refinement algorithm, denoted as Voronoi-based(FRδ) and Voronoi-based(FR$*$). If not using safe zones, we have to access the IR-tree periodically, say every second. Such a method is called the IR-tree-based method.

We use CanuMobiSim[9] to generate a set of trajectories following a Brownian motion model. The query location changes at a maximum speed of 20 units per second and is reported every second. The default trajectory length of a query is 30, i.e., each query has 30 location reports.

Fig. 11 compares the methods in terms of the server side query time and communication cost, on the WW and USA datasets. The Voronoi-based methods dominate the IR-tree-based method for all settings. The results show that the cost of the Voronoi-based methods increases moderately with more frequent updates (higher speed), as a higher speed increases the chance of exiting safe zones. The Voronoi-based(FR$*$) method has a slightly higher time but achieves a better communication cost than the Voronoi-based(FRδ) method. In Fig. 11, when

9. http://canu.informatik.uni-stuttgart.de/mobisim/downloads/



(a) Query time(ms) (ww, Real) (b) Communication cost (ww, Real) (c) Query time(ms) (usa, Real) (d) Communication Cost (usa, Real)

Fig. 12. VGS tracking.

speed equals 20, the communication cost of the Voronoi-based$*$ method is 70 percent better than that of the IR-tree-based method and 50 percent better than that of the Voronoi-based(FRδ) method.

Fig. 12 considers another aspect that affects the expiration of safe zones, namely the volunteers' update interval. We use CanuMobiSim to simulate objects' movement by considering the locations in USA and WW as initial positions. We assume that 20 percent of them are vehicles and the remaining 80 percent are pedestrians with maximum velocity as 20 and 1.5 units per second, respectively. The movements follow a Brownian motion model. The query's velocity is fixed as 10 units/second.

Similar to the trend in Fig. 11, the Voronoi-based solution saves significant computation and communication efforts when compared to the IR-tree based solution. For example, in Fig. 12c, the computational costs of the Voronoi-based methods are 50 times less than that of the IR-tree based method. In Fig. 12d, the Voronoi-based methods consume less than 50-80 percent of the communication cost. Voronoi-based(FRδ) is more computationally efficient but less communication efficient than the Voronoi-based(FR*) method. Thus, each method may be preferable under different optimization goals. Also, the communication cost of the Voronoi-based solution decreases as the update interval grows. This is because a larger interval corresponds to a less frequent updates of imprecise regions as well the safe zone.

8 Related Work

Voronoi cells have been studied extensively in the last decade, often motivated by the popularity of location-based services. Such cells are useful for deriving neighborhood related information, such as safe zones for efficient continuous queries [9], [10], [11], [12], [27], [29], [30], analytical proximity optimization [21], [31], [32], and spatial crowdsourcing [33]. Recently, weighted Voronoi cells have been studied in the context of spatial keyword querying [13], [15]. Imprecise Voronoi cells have also been studied [19], [20], [22].

The paper differs from previous works in several respects. First, previous work on plain Voronoi cells [9], [10], [11], [12], [21], [27], [29], [30], [31], [32] assumes that the spatial data is precise and equally weighted, whereas this paper eliminates both assumptions by proposing weighted imprecise Voronoi cells and designing novel techniques for accommodating such cells. The few recent works that address the effect of data imprecision and introduce imprecise Voronoi cells [19], [20], [22] only consider the spatial aspects of objects. Also, two of these studies [19], [20] represent a Voronoi cell by candidate objects, and one study [22] derives an MBR that coarsely contains a Voroni cell. Our work utilizes progressive approximation to achieve a finer-gained Voronoi cell and allows both representations simultaneously. Third, weighted Voronoi cells for spatial keyword querying are studied [13], [14], [15], by considering either only multiplicative weighted distance [13], [14], or only additively weighted distance [15]. Our work is capable of supporting both, and we do not assume that the spatial attributes are precise. The filtering phase in our work follows insights related to incremental updates [13], [21] in the sense of considering object access order and early stopping techniques derived from properties of bisectors. In past work, the bisector is a straight line [21] or a circle [13], [14], whereas in our problem, it is fourth-order polynomial for circular imprecise regions, and even uses unknown formats for other shapes, e.g., rectangular imprecise regions. Unlike related work [13], we do not maintain a separate candidate object set as well as pruning rules; thus, we retain the ability for general extensions, such as ψ-cells, differently shaped imprecise regions, and different weighted distances.

A Voronoi cell has been studied on the boundaries of weighted circular objects [17]. However, the precise circular shaped objects considered are different from imprecise circular regions; thus, the Voronoi cells considered are also fundamentally different.

An interesting and orthogonal direction is to consider safe regions for spaces with constraint, e.g., road networks. Zheng et al. [34] study continuous queries for objects with uncertainty in road networks. Kolahdouzan and Shahabi [16] study Voronoi-based safe regions for objects in spatial networks.

9 Conclusion

In this paper, we study the problem of volunteered geographic service tracking, which enables mobile user to efficiently monitor nearby services. We reduce the problem to the construction of Voronoi-based safe zones. Then, we propose two novel types of weighted imprecise Voronoi cells, π-cells and ψ-cells. We investigate their geometric features and propose a filter-and-refine framework that enables efficient construction of cells with quality guarantees. Our proposal is versatile in terms of: 1) its capability of deriving both object and shape representations for Voronoi cells; 2) its adaptivity to different distance functions, e.g., additively and multiplicatively weighted distances; 3) its flexibility with respect to differently shaped imprecise regions. Experiments offer insight into the efficiency and scalability of our proposal.

In future research, it is of interest to study how to utilize the proposal for evaluating advanced queries, such as reverse nearest and k-nearest neighbor queries, and how to extend the proposal to constrained spaces, e.g., road networks.

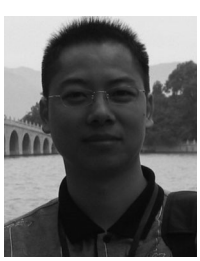
Acknowledgments

The work was supported in part by a grant from the Obel Family Foundation. Peiquan Jin was supported by the National Science Foundation of China (No. 61379037). Man Lung Yiu was supported by grant PolyU 5302/12E from Hong Kong RGC. The authors thank Dr. Dingmng Wu for sharing the source code on f-sided polygon. They also thank anonymous reviewers for their valuable comments. Peiquan Jin is the corresponding author of this paper.

References

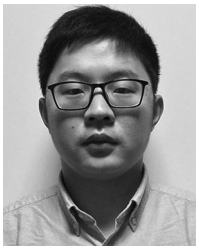
[1] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Proc. IEEE 28th Int. Conf. Data Eng.*, 2012, pp. 1144–1155.
[2] D. Pfoser and C. Jensen, "Capturing the uncertainty of moving-object representations," in *Proc. 6th Int. Symp. Adv. Spatial Databases*, 1999, pp. 111–132.
[3] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2003, pp. 551–562.
[4] X. Lian and L. Chen, "Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data," *VLDB J.*, vol. 18, pp. 787–808, 2009.
[5] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei, "Probabilistic reverse nearest neighbor queries on uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 4, pp. 550–564, Apr. 2010.
[6] K. Zheng, G. P. C. Fung, and X. Zhou, "K-nearest neighbor search for fuzzy objects," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 699–710.
[7] T. Emrich, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle, "Boosting spatial pruning: on optimal pruning of MBRs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2010, pp. 39–50.
[8] G. Trajcevski, R. Tamassia, H. Ding, P. Scheuermann, and I. F. Cruz, "Continuous probabilistic nearest-neighbor queries for uncertain trajectories," in *Proc. 12th Int. Conf. Extending Database Technol.: Adv. Database Technol.*, 2009, pp. 874–885.
[9] J. Xu and B. Zheng, "Energy efficient index for querying location-dependent data in mobile broadcast environments," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 239–250.
[10] S. Nutanong, R. Zhang, E. Tanin, and L. Kulik, "The V*-Diagram: A query-dependent approach to moving KNN queries," *Proc. VLDB Endowment*, vol. 1, pp. 1095–1106, 2008.
[11] G. Albers et al., "Voronoi diagrams of moving points," *Int. J. Comput. Geometry Appl.*, vol. 8, pp. 365–380, 1998.
[12] C. Li, Y. Gu, J. Qi, G. Yu, R. Zhang, and Y. Wang, "Processing moving kNN queries using influential neighbor sets," *Proc. VLDB Endowment*, vol. 8, pp. 113–124, 2014.
[13] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong, "Efficient continuously moving top-k spatial keyword query processing," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, pp. 541–552.
[14] D. Wu, M. L. Yiu, and C. S. Jensen, "Moving spatial keyword queries: Formulation, methods, and analysis," *ACM Trans. Database Syst.*, vol. 38, pp. 1–47, 2013.
[15] W. Huang, G. Li, K.-L. Tan, and J. Feng, "Efficient safe-region construction for moving top-K spatial keyword queries," in *Proc. 21st ACM Conf. Inf. Knowl. Manage.*, 2012, pp. 932–941.
[16] M. Kolahdouzan and C. Shahabi, "Voronoi-based k nearest neighbor search for spatial network databases," in *Proc. 30th Int. Conf. Very Large Databases*, 2004, pp. 840–851.
[17] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. New York, NY, USA: Wiley, 2000.
[18] G. Cong, C. S. Jensen, and D. Wu, "Efficient retrieval of the top-k most relevant spatial web objects," *Proc. VLDB Endowment*, vol. 2, pp. 337–348, 2009.
[19] R. Cheng, X. Xie, M. L. Yiu, J. Chen, and L. Sun, "UV-diagram: A Voronoi diagram for uncertain data," in *Proc. IEEE 26th Int. Conf. Data Eng.*, 2010, pp. 796–807.
[20] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen, "UV-diagram: A Voronoi diagram for uncertain spatial databases," *VLDB J.*, vol. 22, pp. 319–344, 2013.
[21] M. Yiu, N. Mamoulis, and P. Karras, "Common influence join: A natural join operation for spatial pointsets," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 100–109.
[22] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Züfle, Y. Tang, and T. Emrich, "Voronoi-based nearest neighbor search for multi-dimensional uncertain databases," in *Proc. IEEE Int. Conf. Data Eng.*, 2013, pp. 158–169.
[23] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1112–1127, Sep. 2004.
[24] X. Xie, M. L. Yiu, R. Cheng, and H. Lu, "Scalable evaluation of trajectory queries over imprecise location data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 2029–2044, Aug. 2014.
[25] D. Poole, *Linear Algebra: A Modern Introduction*, 3rd ed. Boston, MA, USA: Cengage Learning, 2010.
[26] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, pp. 265–318, 1999.
[27] B. Zheng, J. Xu, W.-C. Lee, and L. Lee, "Grid-partition index: A hybrid method for nearest-neighbor queries in wireless location-based services," *VLDB J.*, vol. 15, pp. 21–39, 2006.
[28] D. Pfoser and C. S. Jensen, "Capturing the uncertainty of moving-objects representations," in *Proc. Int. Conf. Sci. Database Manage.*, 1999, pp. 111–132.
[29] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2003, pp. 443–454.
[30] M. Sharifzadeh and C. Shahabi, "VoR-tree: R-trees with Voronoi diagrams for efficient processing of spatial nearest neighbor queries," *Proc. VLDB Endowment*, vol. 3, pp. 1231–1242, 2010.
[31] L. Hu, W.-S. Ku, S. Bakiras, and C. Shahabi, "Spatial query integrity with Voronoi neighbors," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 863–876, Apr. 2013.
[32] P. Wang et al., "Understanding the spreading patterns of mobile phone viruses," *Sci. Exp.*, vol. 324, pp. 1071–1076, 2009.
[33] L. Kazemi and C. Shahabi, "A privacy-aware framework for participatory sensing," *SIGKDD Explor. Newslett.*, vol. 13, pp. 43–51, 2011.
[34] K. Zheng, G. Trajcevski, X. Zhou, and P. Scheuermann, "Probabilistic range queries for uncertain trajectories on road networks," in *Proc. 14th Int. Conf. Extending Database Technol.*, 2011, pp. 283–294.

Xike Xie received the PhD degree from the Department of Computer Science, University of Hong Kong, in 2012. He is currently an assistant professor in the Department of Computer Science, Aalborg University, Denmark. His research interests include data uncertainty, spatiotemporal databases, and mobile computing. He is a member of the IEEE and ACM.



Peiquan Jin received the PhD degree in computer science from the University of Science and Technology of China (USTC) in 2003. He is currently an associate professor in the School of Computer Science and Technology, USTC. He was a visiting scientist at the University of Kaiserslautern in 2009 and Aalborg University in 2014. His research interests focus on spatiotemporal databases, flash-based databases, and web information retrieval. He is a senior member of the CCF and a member of IEEE and ACM.

Man Lung Yiu received the bachelor's degree in computer engineering and the PhD degree in computer science from the University of Hong Kong in 2002 and 2006, respectively. Prior to his current post, he was at Aalborg University for three years starting in the Fall of 2006. He is currently an associate professor in the Department of Computing, Hong Kong Polytechnic University. His research focuses on the management of complex data, in particular, query processing topics on spatiotemporal data and multidimensional data.



Jiang Du is currently working toward the master's degree in the School of Computer Science and Technology, University of Science and Technology of China (USTC). His research interests include spatiotemporal databases and location-based services.

Mingxuan Yuan received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, China, in 2011. He is a researcher in the Huawei Noah Ark Lab, Hong Kong. His research interests include privacy problems of social networks

Christian S. Jensen is an Obel professor of computer science at Aalborg University, Denmark. He was recently at Aarhus University for three years and at Google Inc. for one year. His research concerns data management and data-intensive systems, and its focus is on temporal and spatiotemporal data management. He has received several national and international awards for his research. He is an editor-in-chief of the *ACM Transactions on Database Systems (TODS)* and was an editor-in-chief of *The VLDB Journal* from 2008 to 2014. He is a fellow of the ACM and IEEE, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences.