Multi-Criteria Optimal Location Query with Overlapping Voronoi Diagrams

Ji Zhang
Computer Science and Software Engineering
Auburn University, AL, USA
jizhang@auburn.edu

Wei-Shinn Ku
Computer Science and Software Engineering
Auburn University, AL, USA
weishinn@auburn.edu

Min-Te Sun
Computer Science and Information Engineering
National Central Univ., Taiwan
msun@csie.ncu.edu.tw

Xiao Qin
Computer Science and Software Engineering
Auburn University, AL, USA
xqin@auburn.edu

Hua Lu
Department of Computer Science, Aalborg University
Aalborg, Denmark
luhua@cs.aau.dk

ABSTRACT

This paper presents a novel optimal location selection problem, which can be applied to a wide range of applications. After providing a formal definition of the novel query type, we explore an intuitive approach that sequentially scans all possible object combinations in the search space. Then, we propose an Overlapping Voronoi Diagram (OVD) model that defines OVDs and Minimum OVDs, and construct an algebraic structure under an OVD overlap operation. Based on the OVD model, we design an advanced approach to answer the query. Due to the high complexity of Voronoi diagram overlap computation, we improve the overlap operation by replacing the real boundaries of Voronoi diagrams with their Minimum Bounding Rectangles (MBR). We also propose a cost-bound iterative approach that efficiently processes a large number of Fermat-Weber problems. Our experimental results show that the proposed algorithms can evaluate the novel query type effectively and efficiently.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management

General Terms

Algorithms, Experimentation

Keywords

Voronoi diagram, Optimal location query

1. INTRODUCTION

Numerous spatial queries, including nearest neighbor and reverse nearest neighbor queries, have been extensively studied; however, there are still spatial queries applied in real applications, such as location decision making, that cannot be efficiently addressed by existing spatial query types. During the location selection process, multiple factors (*e.g.*, distance and reputation) are taken into account. A typical example is making residential location decisions, such as finding home locations that would maximize residential satisfaction, which is a critical part of community planning and development [15]. In order to attract more customers, an optimal location would be selected based on a comprehensive consideration of a number of factors, such as transportation accessibility (the ease of reaching a bus or subway station), the distance to an elementary school, or the distance to a supermarket.

Fig. 1 displays a simple example of residential location selection in a city. We assume that there are two schools, two bus stops, and two supermarkets in the city. Their locations are indicated by symbols. The figure also shows three potential community locations. Lines connect communities to their closest bus stop, school, and supermarket, respectively. The numbers on the lines indicate the distance between two locations. If we assume that the optimal location for a new community is the place that minimizes the total distance to its closest school, bus stop, and supermarket, the best place is *Community* 1, the total distance (16) of which is shorter than that of *Community* 2 (19) or *Community* 3 (18).

Tradeoffs of multiple factors are actually considered in

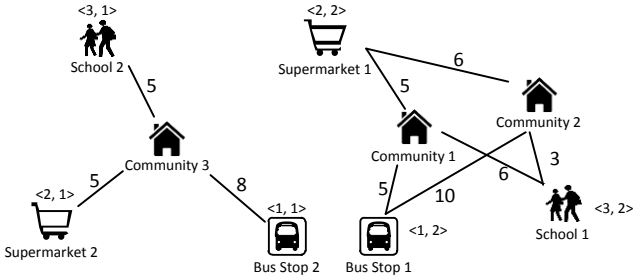


Figure 1: An example of residential location selection. The object weights are indicated as $<w^t, w^o>$.

10.5441/002/edbt.2014.36

real residential location selection [23]. The importance of schools, bus stops, and supermarkets varies greatly among different people. For example, some people may prefer living near a school because it is convenient to drive their children to school. In addition, objects of a particular type are considered differently. When selecting a school, the ones that provide higher quality programs are more attractive. In order to take these differences into consideration, a *type weight* w^t and *object weight* w^o are associated with each object. Providing objects with weights in the location selection allows users to prioritize objects based on their preference. If the weights $<w^t, w^o>$ customized by users are as indicated in Fig. 1, the best choice is *Community* 3 (33), which has the smallest sum of weighted distance to the nearest school (15), bus stop (8), and supermarket (10). We assume that the weighted distance of a community and an object is calculated as the product of the distance and the two weights. The more important object types and the more preferred objects have smaller weights. Instead of associating a single weight with an object, *type weight* and *object weight* are set individually in the example because various weight functions are allowed to be applied to the query. This will be described in Section 2. In the example, a multiplicatively-based weight function is applied to both *type weight* and *object weight*. Appropriately selecting the factors and their weight values is another interesting problem. More discussions can be found in [15, 23]. However, we focus mainly on the novel query type in this paper.

In order to efficiently answer the query, we propose an advanced solution that utilizes the Overlapped Voronoi Diagram (OVD) model and Fermat-Weber techniques. The OVD model integrates location information and object weights w^o of spatial objects by overlapping the diagrams generated from the objects. With the OVD model, the closest objects of different types to a particular location can be efficiently retrieved. Fermat-Weber techniques are used for finding the optimal location of given objects.

In addition, due to a surprisingly large number of Fermat-Weber problems generated in our solution, we propose a cost-bound iterative solution that is able to significantly reduce the computation complexity of the original iterative solution [24]. The contributions of this study are summarized below:

1. We identify a novel query type that is able to find optimal locations comprehensively by considering multiple criteria.

2. We build an OVD model, analyze its properties systematically, and create an algebraic structure of its overlap operations.

3. Based on the OVD model, we propose a Real Region as Boundary (RRB) solution that is able to efficiently evaluate the novel query type.

4. In the proposed Minimum Bounding Rectangle as Boundary (MBRB) solution, we optimize the overlap operation by avoiding overlapping region calculations.

5. Facing a large amount of Fermat-Weber problems, we propose a cost-bound iterative solution that is able to significantly reduce the computation complexity of the original iterative solution.

Table 1: Symbolic notations.

Symbol	Meaning
P_i	An object set
G	An object group
p_i^u	A spatial object in P_i
w^t	Type weight
w^o	Object weight
ς^t	A type weight function
ς^o	An object weight function
$\|S\|$	The number of elements in the set S
ϵ	An error bound
η	A distance bound
γ	A stopping rule used in iterative approaches [22, 24]
$d(.,.)$	Euclidean distance between two objects
$\mathbb{E}$	A set of object sets or groups
$\mathbb{V}$	A set of Voronoi diagrams
$\mathbb{R}$	The search space
$VD(P_i)$	Voronoi diagram of P_i
$Dom(p_j)$	Dominance region of p_j in a Voronoi diagram
OVD	An overlapped Voronoi diagram
OVR	An overlapped Voronoi region
$MOVD$	A minimum overlapped Voronoi diagram

6. We evaluate the performance of the proposed solutions through extensive experiments with real-world data sets.

The rest of this paper is organized as follows. The proposed query and relevant techniques utilized in our solutions are formally defined in Section 2. In Section 3, a basic approach is described. The OVD model is illustrated in Section 4. In Section 5, our solutions of the query are presented. The experimental validation of our design is presented in Section 6. Section 7 surveys related works. We conclude the paper with a discussion of future work in Section 8.

2. PRELIMINARIES

2.1 Definitions

A spatial object is defined by the triple $<l, w^t, w^o>$, where l is its location in the search space, and w^t and w^o are the type weight and object weight associated with the object. $\mathbb{E} = \{P_1, ..., P_n\}$ denotes a universal set of object sets, where $P_i = \{p_i^1, ..., p_i^m\}$ denotes a set of objects of a particular type. $G = \{p_1^u, ..., p_n^v\}$, where $p_1^u \in P_1, ..., p_n^v \in P_n$, denotes an object group, the objects of which are different types. ς^t and ς^o are monotonic weight functions applied to *type weight* and *object weight*. Notations used in this paper are summarized in Table 1.

2.1.1 Weighted Distance of Two Points

Given a point q, a spatial object p, a type weight function ς^t, and an object weight function ς^o, weighted distance considers both the distance between two points $d(.,.)$ and the weights of p. The formal definition is as follows:

$$WD(q, p, \varsigma^t, \varsigma^o) = \varsigma^t(\ \varsigma^o(\ d(q,\ p.l),\ p.w^o\ ),\ p.w^t) \quad (1)$$

2.1.2 Weighted Distance from a Query Point to an Object Group

Given a point q, an object group $G = \{p_1^u, ..., p_n^v\}$, a type weight function ς^t, and object weight functions $\sigma = \{\varsigma_1^o, ..., \varsigma_n^o\}$, we define the weighted distance from q to G as

the sum of WD(q, p_i^s, ς^t, ς_i^o), where $p_i^s \in G$, $\varsigma_i^o \in \sigma$. The formal definition is

$$WGD(q, G, \varsigma^t, \sigma) = \sum_{p_i^s \in G, \varsigma_i^o \in \sigma} WD(q, p_i^s, \varsigma^t, \varsigma_i^o) \quad (2)$$

2.1.3 Minimum Weighted Distance from a Query Point to Object Groups

Given a point q, a set of object sets $\mathbb{E} = \{P_1, ..., P_n\}$, a type weight function ς^t, and object weight functions $\sigma = \{\varsigma_1^o, ..., \varsigma_n^o\}$, we define the minimum weighted distance from q to object combinations of $\mathbb{E}$ as:

$$MWGD(q, \mathbb{E}, \varsigma^t, \sigma) = \min(\{WGD(q, G, \varsigma^t, \sigma) | G \in P_1 \times ... \times P_n\}) \quad (3)$$

2.1.4 Multi-criteria Optimal Location Query (MOLQ)

Given a set of object sets $\mathbb{E} = \{P_1, ..., P_n\}$, a type weight function ς^t, and object weight functions $\sigma = \{\varsigma_1^o, ..., \varsigma_n^o\}$ where ς_i^o is applied to an object $p_i^u \in P_i$, the purpose of the query is to find an optimal location l in the search space $\mathbb{R}$ that minimizes $MWGD(l, \mathbb{E}, \varsigma^t, \sigma)$.

$$MOLQ(\mathbb{E}, \varsigma^t, \sigma) = l, \quad \text{where}$$
$$MWGD(l, \mathbb{E}, \varsigma^t, \sigma) = \min(\{MWGD(l', \mathbb{E}, \varsigma^t, \sigma) \mid l' \in \mathbb{R}\}) \quad (4)$$

2.2 Voronoi Diagram

2.2.1 Ordinary Voronoi Diagram

Given a set of objects $P_i = \{p_i^1, ..., p_i^m\}$, the ordinary Voronoi diagram $VD^O(P_i)$ is defined as a number of dominance regions $\{Dom^O(p_i^u) | p_i^u \in P_i\}$, each of which is dominated by an object p_i^u. All locations in $Dom^O(p_i^u)$ are closer to p_i^u than other objects.

$$Dom^O(p_i^u) = \{\ l \mid d(l, p_i^u.l) \ \leq d(l, p_i^v.l), \ u \neq v, p_i^u, p_i^v \in P_i\} \quad (5)$$

2.2.2 Weighted Voronoi Diagram

In a weighted Voronoi diagram, generators have different weights reflecting their variable properties. Given a set of objects $P_i = \{p_i^1, ..., p_i^m\}$ and a weight function ς, the dominance regions are measured by weighted distance.

$$VD^W(P_i) = \{Dom^W(p_i^u) \mid p_i^u \in P_i\} \ where$$
$$Dom^W(p_i^u) = \{\ l \mid \varsigma(\ d(\ l,\ p_i^u.l),\ p_i^u.w^o\ ) \ \leq \varsigma(\ d(\ l,\ p_i^v.l),\ p_i^v.w^o\ ), u \neq v,\ p_i^u, p_i^v \in P_i\} \quad (6)$$

2.3 Fermat-Weber Point

Given a point group $G = \{p_1^u, ..., p_n^v\}$ in a d-dimensional space $\mathbb{R}^d$, the Fermat-Weber point is the point q which minimizes the cost function [2].

$$c(q, G) = \sum_{p_i^s \in G} p_i^s.w^t \times d(q, p_i^s.l) \quad (7)$$

The point exists for any point set and is unique except in the event that all the points lie on a single line [8]. In the non-collinear case, the cost function is strictly convex [22].

The solution to the three-point Fermat-Weber problem had been proposed in [9]. In the collinear case of any point set, an optimal point can be found in linear time [2]; however, to the best of our knowledge, if the number of points is greater than three, no exact solution has been reported for non-collinear cases. Instead, an iterative approach is used as an approximate solution proposed in [22, 24]. This approach converges monotonically to the unique optimal location during iterations.

Algorithm 1 SSC($\mathbb{E}$, ς^t, σ)

```
1. Ubound = ∞
2. l = < 0, 0 >
3. for < p_1^u, ..., p_n^v > ∈ P_1 × ... × P_n do
4.    Calculate the optimal location l_1 of < p_1^u, p_2^s >
5.    if WGD(l_1, {p_1^u, p_2^s}, ς^t, σ) < Ubound then
6.       /* use a Fermat-Weber method */
7.       Calculate the optimal location l_2 of < p_1^u, ..., p_n^v >
8.       Cost = WGD(l_2, {p_1^u, ..., p_n^v}, ς^t, σ)
9.       if Cost < Ubound then
10.         Ubound = Cost
11.         l = l_2
12.      end if
13.   end if
14. end for
15. return l
```

The iterative approach starts with an arbitrary location q_0 ($q_0 \notin G$) in $\mathbb{R}^d$. In each iteration, a new location $q_i = f(q_{i-1}, G)$ is produced based on a location q_{i-1} found before the iteration. According to the monotonic convergence property, q_i is closer to the Fermat-Weber point than q_{i-1}; hence, theoretically, the Fermat-Weber point is located at $\lim_{n\to\infty} f^n(q_0, G)$, which indicates a location obtained after infinite iterations. The function f is described below.

$$f(q, G) = \begin{cases} \sum_{p_i^s \in G} \{g_i^s(q) \times p_i^s.l\} & \text{if } q \notin G \\ q & \text{Otherwise} \end{cases} \quad (8)$$

where

$$g_i^s(q) = \frac{p_i^s.w^t}{d(q, p_i^s.l)} \times \{ \sum_{p_{i'}^{s'} \in G} \frac{p_{i'}^{s'}.w^t}{d(q, p_{i'}^{s'}.l)} \}^{-1} \quad (9)$$

Setting an acceptable deviation from the cost of the optimal location as the stopping rule is widely used in applications [17]. For example, given an error bound ϵ, the location after the n^{th} iterations l^n, and the optimal location l^∞, the iteration procedure will stop when $\frac{c(l^n,G)-c(l^\infty,G)}{c(l^\infty,G)} \leq \epsilon$, where $c(l^\infty, G)$ is approximated by a lower bound of the cost at l^n:

$$lb(l^n) = \sum_{k=1}^{d} (\min_x (\sum_{p_i^s \in G} p_i^s.w^t \frac{|l^n.x_k - p_i^s.l.x_k||x - p_i^s.l.x_k|}{d(l^n, p_i^s.l)})) \quad (10)$$

3. SEQUENTIAL SCAN COMBINATIONS ALGORITHM

One basic algorithm to solve MOLQ is to sequentially check optimal locations of all object combinations. In particular, given $\mathbb{E} = \{P_1, ..., P_n\}$, the optimal locations l's of all combinations $\{p_1^u, ..., p_n^v\}$, where $p_1^u \in P_1, ..., p_n^v \in P_n$, are calculated by the Fermat-Weber method, which considers both object locations and their type weights. The answer to the query is the best location among these l's. We call this algorithm the *Sequential Scan Combinations* (SSC) algorithm.

Since the computation of SSC is expensive, we can set an upper bound to reduce the complexity of the algorithm by filtering out a portion of combinations whose optimal locations cannot be the answer. For example, two combinations (object groups), G_1 and G_2, will be evaluated sequentially in SSC. We assume the optimal location of G_1 is at l_1. The weighted distance from l_1 to G_1 is denoted by d_1. Before processing $G_2 =< p_1^u, ..., p_n^v >$, we first set d_1 as an upper bound and calculate the optimal location l_2 of $< p_1^u, p_2^s >$, which costs much less than computing an optimal location of multiple points. If the weighted distance from l_2 to $< p_1^u, p_2^s >$ is greater than d_1, the weighted distance from any location to G_2 must be greater than d_1. Thus, calculating the optimal location of G_2 can be avoided. During SSC processing, the upper bound is initialized to infinity and will be reduced to the total weighted distance of the best solution found so far. The detailed steps of SSC are described in Algorithm 1.

4. OVD AND MOVD MODELS

Before describing our MOVD-based solutions, we will first introduce the OVD and MOVD models. In this section, we start with a simple OVD example which provides a basic understanding of the model. Then, we formally define OVD and Minimum OVD (MOVD) and systematically analyze their properties, which not only highlight the difference from and relationship with Voronoi diagrams, but also provide correctness analyses of our MOVD-based solutions. Finally, we build an algebraic structure of MOVD on an overlap operation $\oplus$. We also discuss the properties of MOVD, which are part of the proof of our proposed solution and will be used in the next section.

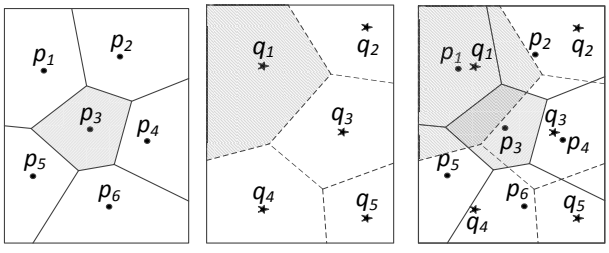
4.1 An OVD Example

Fig. 2(a) and Fig. 2(b) display two ordinary Voronoi diagrams generated by schools and supermarkets, respectively. The shaded areas in the figures are dominance regions of generators p_3 and q_1. Fig. 2(c) shows an OVD that overlaps the two ordinary Voronoi diagrams. Apparently, the OVD is comprised of a number of overlapping regions, each of which is generated by overlapping two ordinary Voronoi polygons. For example, the doubly shaded area in Fig. 2(c) is the overlapping region in both shaded regions of two ordinary Voronoi diagrams. According to the properties of Voronoi diagrams, p_3 and q_1 are the closest school and supermarket to any locations in the doubly shaded region.

4.2 Overlapped Voronoi Diagram Definition

4.2.1 Overlapped Voronoi Diagram (OVD)

Given a set of object sets $\mathbb{E} = \{P_1, ..., P_n\}$ and a set of Voronoi diagrams $\mathbb{V} = \{VD(P_i)|P_i \in \mathbb{E}\}$, where $VD(P_i)$ can be either an ordinary or a weighted Voronoi diagram generated by P_i in the search space $\mathbb{R}$, Overlapped Voronoi Diagram (OVD) is a set of Overlapped Voronoi Regions (OVR),



(a) Schools (b) Supermarkets (c) An OVD

Figure 2: Ordinary Voronoi diagrams and OVDs.

$$OVD(\mathbb{E}) = \{OVR_j \mid 1 \leq j \leq m\} \tag{11}$$

where OVR_j is

$$OVR(p_1^u, ..., p_n^v) = \{l | l \in Dom(p_1^u), ..., l \in Dom(p_n^v), \; p_1^u \in P_1, ..., p_n^v \in P_n\} \tag{12}$$

PROPERTY 1. *An OVD may have one or more empty set OVRs (e.g., $OVR_j = \emptyset$).*

PROOF. By definition, an OVR is the intersection of dominance regions from different Voronoi diagrams. These dominance regions may not overlap each other (see the dominance regions of p_1 in Fig. 2(a) and q_5 in Fig. 2(b)). If this is the case, no locations fall into both dominance regions, thus their overlapping region is an empty set. □

4.2.2 Minimum OVD (MOVD)

A Minimum Overlapped Voronoi Diagram (MOVD) is an OVD in which all empty OVRs have been removed. An OVD is an MOVD if it does not have any empty OVRs. The formal definition of MOVD is:

$$MOVD(\mathbb{E}) = OVD(\mathbb{E}) - \{\emptyset\} \tag{13}$$

In the extreme case that $\mathbb{E}$ is an empty set, no Voronoi diagrams are overlapped, and the search space is not decomposed into subregions. We define this case as:

$$MOVD(\emptyset) = OVD(\emptyset) = \{\mathbb{R}\} \tag{14}$$

4.2.3 OVD/MOVD Properties

A number of properties and proofs can be derived from OVD/MOVD definitions. These properties are the basis of the OVD/MOVD model utilized in our MOVD-based solution.

PROPERTY 2. $|MOVD(\mathbb{E})| \leq |OVD(\mathbb{E})| = \prod_{P_i \in \mathbb{E}} |P_i|$.

PROOF. By Equation 12, OVRs are generated by a combination of selected Voronoi regions. The number of OVRs in $OVD(\mathbb{E})$ is the product of the number of Voroni regions in these Voronoi diagrams. Because all the possible empty sets have been removed, the size of $MOVD(\mathbb{E})$ is less than or equal to $OVD(\mathbb{E})$. □

PROPERTY 3. *Any MOVD fully covers the entire search space $\mathbb{R}$.*

$$\bigcup_{OVR_j \in MOVD(\mathbb{E})} OVR_j = \mathbb{R} \tag{15}$$

PROOF. According to the Voronoi diagram property that a Voronoi diagram fully covers the entire search space, $\forall l \in \mathbb{R}$, there must exist Voronoi regions $\{Dom(p_i^s) \in VD(P_i) | l \in Dom(p_i^s), p_i^s \in P_i, P_i \in \mathbb{E}\}$. By Equation 12, the location l is at the $OVR(p_1^u, ..., p_n^v)$, and an OVD fully covers the entire search space. Moreover, because $MOVD(\mathbb{E})$ only removes empty sets from $OVD(\mathbb{E})$, $MOVD(\mathbb{E})$ covers the entire search space as well. □

PROPERTY 4. *The overlapping area of two different OVRs is a subset of their common boundaries.*

PROOF. By Equation 12, an OVR is the overlapping region of $\{Dom(p_1^u),..., Dom(p_n^v)\}$. If we have two OVRs from an OVD such that $OVR = \bigcap_{p_i^s \in \{p_1^u,...,p_n^v\}} Dom(p_i^s)$, and $OVR' = \bigcap_{p_i^{s'} \in \{p_1^{u'},...,p_n^{v'}\}} Dom(p_i^{s'})$, then the overlapping area of OVR and OVR' is

$$\begin{aligned} &OVR \bigcap OVR' \\ &= (\bigcap_{p_i^s \in \{p_1^u,...,p_n^v\}} Dom(p_i^s)) \bigcap (\bigcap_{p_i^{s'} \in \{p_1^{u'},...,p_n^{v'}\}} Dom(p_i^{s'})) \\ &= \bigcap_{i=1}^{n} (Dom(p_i^s) \bigcap Dom(p_i^{s'})) \end{aligned} \tag{16}$$

According to the properties of the Voronoi diagram, $Dom(p_i^s) \cap Dom(p_i^{s'})$, where $p_i^s \neq p_i^{s'}, p_i^s, p_i^{s'} \in P_i$, is either their common boundaries or an empty set. Moreover, if OVR and OVR' are different, there must exist a p_i^s and $p_i^{s'}$ that are different. The boundaries of an OVR are comprised of the boundaries of corresponding Voronoi regions. Hence, the overlapping region of OVR and OVR' is a subset of their common boundaries. □

PROPERTY 5. *Given a type weight function ς^t, and object weight functions $\sigma = \{\varsigma_1^o, ..., \varsigma_n^o\}$, a point q in $OVR(p_1^u, ..., p_n^v)$, the total weighted distance from q to the corresponding object group $G = \{p_1^u, ..., p_n^v\}$ is the minimum weighted distance from q to all object combinations G', where $G' \in P_1 \times ... \times P_n$.*

$$\begin{aligned} &WGD(q, G, \varsigma^t, \sigma) = \min(\\ &\quad \{\ WGD(q, G', \varsigma^t, \sigma) \mid G' \in P_1 \times ... \times P_n\ \}\ ) \end{aligned} \tag{17}$$

PROOF. If $VD(P_1)$ is generated by P_1 and the weight function $\varsigma_1^o \in \sigma$, a point q in $OVR(p_1^u, ..., p_n^v)$ must fall in $Dom(p_1^u)$ of $VD(P_1)$ so that p_1^u is the closest point in P_1 to q. $WD(q, p_1^u, \varsigma^t, \varsigma_1^o)$ is the minimum weighted distance from q to any points in P_1. We can get the same result in other sets $P_i \in \mathbb{E}$. After summing them up, we obtain Property 5 that $WGD(q, G, \varsigma^t, \sigma)$ has the minimum distance. □

PROPERTY 6. *$|MOVD(\mathbb{E})|$ is bigger than or equal to $|VD(P_i)|$, where $P_i \in \mathbb{E}$.*

$$|MOVD(\mathbb{E})| \geq |VD(P_i)| \tag{18}$$

PROOF. Overlapping two Voronoi diagrams is a process in which one Voronoi diagram is decomposed by another Voronoi diagram. Each Voronoi region is divided into a number of subregions, unless two Voronoi regions from different VDs are exactly the same, or one region contains the other. In these extreme cases, the Voronoi region remains unchanged. Thus, after overlapping Voronoi diagrams, the number of overlapping regions in an MOVD is either greater than or equal to the basic Voronoi diagrams. □

PROPERTY 7. *When $\mathbb{E}$ is made up of only one object set $\mathbb{E} = \{P\}$, then*

$$MOVD(\mathbb{E}) = OVD(\mathbb{E}) = VD(P) \tag{19}$$

PROOF. This property is straightforward. If $\mathbb{E}$ has only one object set P, there is no other Voronoi diagram overlapped on $VD(P)$. Obviously $VD(P)$ does not have any empty regions. Therefore, $OVD(\mathbb{E})$ and $MOVD(\mathbb{E})$ are identical to $VD(P)$. This property not only states an extreme case of definitions, but also highlights basic units in the OVD/MOVD model. All OVDs are generated from these building blocks. □

4.3 Algebraic Structure of MOVD

After theoretically introducing the OVD/MOVD model, we will mainly focus on the overlap operation. We create an algebraic structure of MOVD by exploring MOVD space under the overlap operation and discussing its properties. The implementation details of the operation will be presented in Section 5.

4.3.1 MOVD space

MOVD space is a universal set of MOVDs that are fed into and produced by the overlap operation. Given a universal set of object sets $\mathbb{E} = \{P_1, ..., P_n\}$, the universal set of MOVD($\mathbb{E}$) is defined as

$$U(MOVD(\mathbb{E})) = \{MOVD(E_i) \mid E_i \subseteq \mathbb{E}\} \tag{20}$$

PROPERTY 8. *The number of MOVDs existing in the universal space is as follows:*

$$|U(MOVD(\mathbb{E}))| = \sum_{i=0}^{|\mathbb{E}|} \binom{|\mathbb{E}|}{i} \tag{21}$$

PROOF. By definition, MOVD space consists of a number of MOVDs, each of which is generated by a subset of $\mathbb{E}$; thus the number of MOVDs in the space equals the number of subsets in $\mathbb{E}$, which is presented as Equation 21. The case that i equals 0 indicates a special subset, the empty set, defined in Equation 14. □

4.3.2 Overlap operation $\oplus$

We define a binary operation $\oplus$ that overlaps two given MOVDs. The result of $\oplus$ is a new MOVD generated by the union of generator sets of input MOVDs. The formal definition is: given MOVD(E_i) and MOVD(E_j), where $E_i, E_j \subseteq \mathbb{E}$, then

$$MOVD(E_i) \oplus MOVD(E_j) = MOVD(E_i \cup E_j) \tag{22}$$

A general implementation (RRB) of the operation will be discussed in Section 5.2.

4.3.3 $\oplus$ Operation Properties

By properties of the union operation on sets, we can obtain the following three laws.

PROPERTY 9. *Idempotent Law*

$$MOVD(E_i) \oplus MOVD(E_i) = MOVD(E_i) \tag{23}$$

PROPERTY 10. *Commutative Law*

$$\begin{aligned} &MOVD(E_i) \oplus MOVD(E_j) \\ &\quad = MOVD(E_j) \oplus MOVD(E_i) \end{aligned} \tag{24}$$

PROPERTY 11. *Associate Law*

$$\begin{aligned} &(MOVD(E_i) \oplus MOVD(E_j)) \oplus MOVD(E_k) = \\ &\quad MOVD(E_i) \oplus (MOVD(E_j) \oplus MOVD(E_k)) \end{aligned} \tag{25}$$

COROLLARY 4.1. *MOVD(E_i), where $E_i \subseteq \mathbb{E}$, is unique.*

PROOF. According to the commutative and associate laws of operation $\oplus$, the order of overlapping Voronoi diagrams does not cause the result to change. Thus MOVD(E_i) is unique. □

PROPERTY 12. *$MOVD(\emptyset)$ is an identity element.*

PROOF. $MOVD(\emptyset)$ equals $\{\mathbb{R}\}$ such that it leaves MOVDs unchanged under operation $\oplus$. The following equation can be easily proved by the definition of $\oplus$.

$$MOVD(E_i) \oplus MOVD(\emptyset) = MOVD(E_i) \tag{26}$$

□

PROPERTY 13. *Closure: the universal MOVD space of $\mathbb{E}$ is closed under operation $\oplus$.*

PROOF. By definition, given any MOVD(E_i) and MOVD(E_j), where $E_i, E_j \subseteq \mathbb{E}$, the result of overlapping them is MOVD($E_i \cup E_j$). $E_i \cup E_j$ is still a subset of $\mathbb{E}$, so the result is an element of $U(MOVD(\mathbb{E}))$. □

Definition Sequential Overlap Operations

$$\begin{aligned}\sum_{i=1}^{n} MOVD(E_i) &= MOVD(E_1) \oplus ... \oplus MOVD(E_n) \\ &= MOVD(\bigcup_{i=1}^{n} E_i)\end{aligned} \tag{27}$$

Definition Partial Order
If $MOVD(E_i) = MOVD(E_j) \oplus MOVD(E_k)$ then,

$$\begin{aligned}MOVD(E_i) &\succcurlyeq MOVD(E_j) \\ MOVD(E_i) &\succcurlyeq MOVD(E_k)\end{aligned} \tag{28}$$

The partial order definition formalizes a comparison model for evaluating how much information MOVDs maintain. As Equation 28 shows, $MOVD(E_i)$ is generated by $MOVD(E_j)$ and $MOVD(E_k)$. $MOVD(E_i)$ has more information (*i.e.*, objects) than either $MOVD(E_j)$ or $MOVD(E_k)$. We use $\succcurlyeq$ to denote the relationship.

PROPERTY 14. *$MOVD(E_i) \oplus MOVD(E_j) = MOVD(E_i)$ if $MOVD(E_i) \succcurlyeq MOVD(E_j)$.*

PROOF. The following equation proves Property 14 by applying the partial order definition that decomposes $MOVD(E_i)$ into $MOVD(E_j)$ and $MOVD(E_k)$, and the commutative and idempotent laws of operation $\oplus$. □

$$\begin{aligned}&MOVD(E_i) \oplus MOVD(E_j) \\ &\quad = MOVD(E_j) \oplus MOVD(E_k) \oplus MOVD(E_j) \\ &\quad = MOVD(E_j) \oplus MOVD(E_j) \oplus MOVD(E_k) \\ &\quad = MOVD(E_j) \oplus MOVD(E_k) \\ &\quad = MOVD(E_i)\end{aligned} \tag{29}$$

5. MOVD-BASED ALGORITHMS

After introducing the OVD model, we now illustrate our MOVD-based algorithms for the query. In this research we mainly focus on applying the properties of OVD and MOVD models to solve the proposed novel query type. Therefore, the proposed algorithms primarily rely on main memory for data storage.

5.1 Framework of the MOVD-based Solution

Fig. 3 illustrates the framework of our solution. The inputs are Point of Interest (POI) data sets ($P_i \in \mathbb{E}$), object weight functions $\sigma = \{\varsigma_1^o, ..., \varsigma_n^o\}$, and a type weight function ς^t. The result is an optimal location of the query.

In the evaluation system, the query is sequentially processed by three modules. In particular, based on POIs of particular types and the object weight functions, *VD Generator* generates Voronoi diagrams that are the basic MOVDs used in the next step (see Property 7). Then, a new MOVD is produced by overlapping the basic MOVDs with *MOVD Overlapper* (see Equation 27). A significant number of impossible object combinations are filtered out, which reduces the cost of Fermat-Weber computation in the next step. Finally, *Optimizer* sequentially scans OVRs in the new MOVD, finding a locally optimal location in each OVR, and returns the best of these locations as the query result.

Essentially, two solutions are proposed in Fig. 3, illustrated by two paths from the *VD Generator* to the *Optimizer*. The solutions apply either Real Region as Boundary (RRB) or Minimum Bounding Rectangle as Boundary (MBRB) approaches in the *MOVD Overlapper*. The RRB approach provides real boundaries of OVRs in the new MOVD by calculating the overlapping regions, which is expensive if the regions are complex. The MBRB approach can avoid the real region calculation, but it produces false positives that would incur unnecessary calculation during overlapping the next MOVD. Which approach performs better depends on the number and the complexity of MOVDs generated by the *VD Generator*. The two MOVD overlapping approaches will be described in the following two subsections. A cost-bound approach used in *Optimizer* will be presented in Section 5.4. The Voronoi diagram generation approaches used in the *VD Generator* can be found in [14].

5.2 RRB Approach

In this subsection, we describe the RRB approach for MOVD overlapping operations. Since basic MOVDs are identical to Voronoi diagrams (see Property 7), the generation methods of which have been extensively studied, we will mainly focus on the process of creating an MOVD from two MOVDs. For a better explanation, overlapping two basic MOVDs is illustrated by the simple example in Fig. 4.

A plane-sweep-based algorithm is designed in the RRB approach. As the typical plane sweep approach [4], the RRB approach maintains an event queue and two sweeping statuses. The event queue consists of a number of event points that are the maximum and minimum values of projections of OVRs on the y axis. These maximum and minimum points

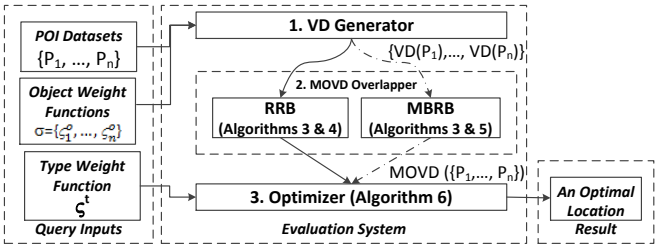


Figure 3: Framework of the MOVD-based solution. The paths of RRB and MBRB solution are indicated by solid and dashed arrows, respectively.

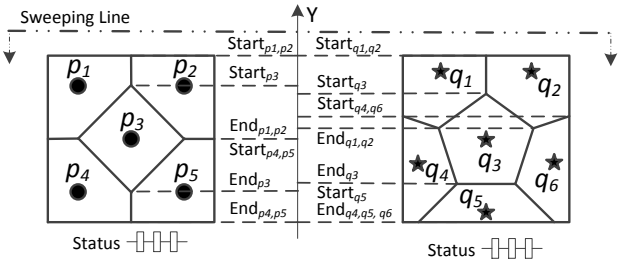


Figure 4: Overlapping two MOVDs.

are called start and end points, which indicate that when the sweeping line arrives at these points, the corresponding OVR starts or ends its intersection with the sweeping line. The event points of both MOVDs are sorted by their y-coordinates in descending order. The sweeping line vertically scans the plane from top to bottom, so that the start point of an OVR will be reached before its end point. The status structures are set up to record OVRs that intersect with the sweeping line. Two status structures are maintained individually and respectively for MOVDs. OVRs also have a range (minimum and maximum values) of projections on the x axis. The event points are pre-determined before the overlap calculation.

During the sweeping process, when an end point is arrived at, the corresponding OVR is removed from the status structure. When the sweeping line reaches a start point, the corresponding OVR is inserted into the status structure. Moreover, overlapping regions of the new OVR and OVRs in the other status structures are required to be detected. The detection process first identifies potential OVRs, the range of which overlaps with the new OVR on the x axis. Then, the overlapped region of the two OVRs is calculated. The details are described in Algorithms 2 and 3.

The essential idea of the algorithms is that the minimum and maximum values on the x and y axes are an outer boundary of OVR. Two OVRs cannot overlap each other if the area inside their outer boundaries do not overlap. Overlapped outer boundary detection significantly reduces overlapping region calculations by avoiding the overlapping of two OVRs (*e.g.,* regions of p_1 and q_5 in Fig. 4), which are actually far away from each other.

As shown in Algorithm 2, the overlap operation receives two MOVDs as input parameters and produces a new MOVD. From lines 1-4, *Result*, *EventQueue*, *Status* and *Status*$'$ are initialized to be empty sets. *Status* keeps the status for $MOVD(E)$, and *Status*$'$ for $MOVD(E')$. Then, in lines 5-6, events are inserted into *EventQueue* and sorted. Finally, from lines 7-14, all events are iteratively handled by Algorithm 3.

Algorithm 3 describes the event handler that receives the following four parameters. e is an event object. *Current* is the status structure of MOVD from which the event occurs. *Other* refers to the other status structure. *Result* is the MOVD produced by the overlap operation. As shown in Fig. 6, an MOVD manages a list of OVRs, each of which is represented as <*region*, *pois*>, where *region* maintains the shape of the OVR and *pois* is a list of objects associated with the OVR. If a start event occurs, the corresponding OVR is first inserted into the *Current* status. Then, potentially overlapped OVRs in *Other* are detected by comparing their $Range_X$ with the current OVR. $Range_X$ denotes the range of possible x-coordinates of OVRs. If their $Range_X$ overlap, the overlapped region is calculated in line 5. If the newly generated overlapped region is not empty, a pair of the region and its associated *pois* will be appended to *Result*. In the second branch, an end event takes place and the corresponding OVR is removed from *Current*.

It is worth noting that due to the space limitation of this paper, a general overlapping approach is not presented; however, the RRB approach can be modified to be a general approach used for the OVD model if line 7 is removed and only *region* is appended to *Result* in line 8. *pois* contains the additional information for our specific query type. Algorithm 3 does not specify any methods for overlapping region calculation in line 5. The reason is that the shape of OVRs in a general model is difficult to predict. The case is worse after overlapping because the OVRs become more complex. Furthermore, overlap methods for regions vary greatly as well. The overlap methods for polygons are different from the ones for circles. The overlap methods applied in the model cannot be determined until the shapes of regions have been decided. We will discuss this issue in Section 5.3.

The RRB approach is an output-sensitive algorithm, the complexity of which depends on the size of the results, or more exactly the number of OVRs existing in the new MOVD. We denote the average size of input MOVDs by n. There are $4 \times n$ events in total, and sorting them in order takes $O(n \lg n)$ time. There are $2 \times n$ start and end events handled by Algorithm 3. If status structures are organized as a balanced search tree that sorts OVRs in order by their start x-coordinates, inserting or deleting an OVR from the status can be completed in $O(\lg n)$ time. The total cost of maintaining the status is $O(n \lg n)$ as well. If status structures record the start and end x-coordinates of OVRs, a range specified by the points that are either immediately smaller than the minimum or greater than the maximum x-coordinate of the current OVR can be figured out in $O(\lg n)$ time. The OVRs, whose event points are located at the range, are potentially required to overlap the current OVR. Moreover, we denote the number of OVRs in the result by I and the cost of overlapping region computation by θ. The cost of calculating the overlap regions is $\theta \times I$. In the worst case, I becomes n^2, so that the total cost of operation $\oplus$ is $O(\theta \times n^2)$.

Algorithm 2 Overlap($MOVD(E)$, $MOVD(E')$)

```
1.  Result = ∅
2.  EventQueue = ∅
3.  Status = ∅
4.  Status' = ∅
5.  Push events of MOVD(E) and MOVD(E') into
    EventQueue
6.  Sort(EventQueue)
7.  while ( EventQueue ≠ ∅ ) do
8.    e = EventQueue.pop()
9.    if ( e is from MOVD(E) ) then
10.     EventHandler(e, Status, Status', Result)
11.   else
12.     EventHandler(e, Status', Status, Result)
13.   end if
14. end while
15. return Result
```

Algorithm 3 EventHandler(e, *Current*, *Other*, *Result*)

```
1. if e is a start event then
2.    Insert e.ovr into Current
3.    for ovr ∈ Other do
4.       if Range_X(e.ovr) ∩ Range_X(ovr) ≠ ∅ then
5.          region = e.ovr.region ∩ ovr.region
6.          if region ≠ ∅ then
7.             pois = e.ovr.pois ∪ ovr.pois
8.             Result.append(< region, pois >)
9.          end if
10.      end if
11.   end for
12. else
13.   Remove e.ovr from Current
14. end if
15. return
```

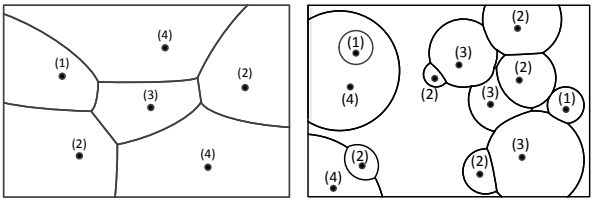
5.3 MBRB Approach

According to the variety of weight functions specified in the query inputs, various Voronoi diagrams are generated by the *VD Generator*. In addition to the ordinary Voronoi diagrams, two typical weighted Voronoi diagrams are displayed in Fig. 5. The generation methods of additively and multiplicatively Voronoi diagrams have been presented in [1, 10, 5, 13]. More practical Voronoi diagrams can be found in [14].

Although the generation methods of weighted Voronoi diagrams have been extensively studied, efficiently maintaining the shape of OVRs is extremely difficult since they are not in regular shapes. In general, their boundaries have to be modelled by a number of curves. More importantly, overheads of overlapping region calculations would be highly expensive due to the complexity of boundary representation.

To overcome this difficulty, we propose the MBRB approach that combines Algorithm 2 with an alternative event handler, MBRBHandler, for the overlap operation. The MBRB approach is motivated by an observation that the shapes of OVRs are not used in *Optimizer*. Instead, the POI locations and their weights are the criteria for optimal location selection; therefore, we set the Minimum Bounding Rectangles (MBR) of OVRs as their shapes in this approach. Two OVRs will be treated as overlapped if their MBRs are overlapped. This approach is able to significantly reduce the cost of the overlap operation by simplifying boundary maintenance and avoiding real region overlapping calculations (line 5 in Algorithm 3 is replaced by line 5 in Algorithm 4); however, the approach suffers from the issue that unnecessary OVRs (false positives which are not really overlapped) would be appended to the new MOVD.

The data structure used in MBRBHandler is shown in



(a) Additively (b) Multiplicatively

Figure 5: Weighted Voronoi diagrams (the numbers indicate weights).

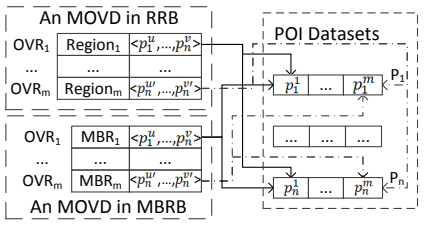


Figure 6: Data structure.

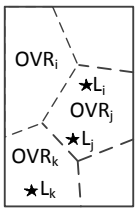


Figure 7: Optimal locations.

Fig. 6. An OVR is indicated as $<MBR$, $pois>$, where an MBR is comprised of minimum and maximum points on the x and y axes, and $pois$ is a list of objects associated with the OVR.

The MBRBHandler is described in Algorithm 4. In the branch that handles start event processing, the handler only detects whether two MBRs are overlapped. If this is the case, the MBRs are overlapped and the objects associated with the two OVRs are merged. The new OVR is appended to the result. The final branch remains unchanged.

Compared to the RRB approach, the complexity of region overlapping θ decreases in constant time, but the size of output I increases, the performance impact of which is difficult to evaluate. The upper bound of I is n^2; therefore, the complexity of the MBRB approach becomes $O(n^2)$ in the worst case.

It is worth noting that the basic principle of our solutions is that the search space is decomposed into a number of OVRs, in which a locally optimal location is found by *Optimizer*; however, the shapes of OVRs are not calculated in the MBRB approach. How does the MBRB solution determine an optimal location in an OVR?

The MBRB solution does not limit the locally optimal location in a particular OVR. Instead, we look for it in the entire search space. As shown in Fig. 7 (next to Fig. 6), if an optimal location L_k is found in OVR_k, L_k will undoubtedly be appended to the candidate list. If the optimal location L_i is outside of OVR_i, according to Property 3, L_i must be located in another OVR, for example OVR_j, which must have an optimal location L_j. L_j must be identical or better than L_i. Appending them to the candidate list does not change the global optimum since only the best one will be returned as the query result. Thus, appending L_i to the candidate list does not change the global optimum.

5.4 A Cost-Bound approach in Optimizer

Algorithm 4 MBRBHandler(e, *Current*, *Other*, *Results*)

```
1. if e is a start event then
2.    Insert e.ovr into Current
3.    for ovr ∈ Other do
4.       if Range_X(e.ovr) ⋂ Range_X(ovr) ≠ ∅ then
5.          mbr = e.ovr.MBR ⋂ ovr.MBR
6.          pois = e.ovr.pois ⋃ ovr.pois
7.          Results.append(<mbr, pois>)
8.       end if
9.    end for
10. else
11.   Remove e.ovr from Current
12. end if
13. return
```

An optimal location q that minimizes $MWGD(q, \mathbb{E}, \varsigma^t, \sigma)$ is found in the third step of the proposed framework. The framework does not specify a weight function for type weight calculations; however, we mainly focus on a multiplicatively-based weight function, which is one of the practical methods used in real applications. Other weight functions can be applied in the framework as well.

If applying a multiplicatively-based weight function to type weights, the problem of finding an optimal location in each OVR is converted into a typical Fermat-Weber problem in two-dimensional space. The objects associated with OVRs are the points in the Fermat-Weber problems. The weights of the points are specified by the type weight function ς^t. The object weights are integrated into the distance from a location to points. As mentioned in Section 2.3, the problem has been solved theoretically. The optimal location in three-point cases and multiple-collinear-point cases can be found in constant and linear time, respectively. An approximate iterative approach has been proposed for other cases [24].

In the RRB and MBRB approaches, we observe that a large number of OVRs will be created by *MOVD Overlapper* (see Property 2). In addition, the number of the Fermat-Weber problems increases rapidly when the number of objects grows. A basic approach is to sequentially calculate the optimal locations of these Fermat-Weber problems and select the best one as the query result; however, applying the iterative method to the Fermat-Weber problems is very expensive. Therefore, we propose a cost-bound approach in which an optimal cost is set as a global lower bound. During the processing of a Fermat-Weber problem, a local lower bound of the cost in each iteration will be calculated. If the local lower bound is greater than the global lower bound, no matter how many iterations will be processed, its local optimal cost cannot be better than the global lower bound. Thus the following iterations can be avoided, even though the stopping condition has not been satisfied. The definition and the cost-bound approach of the problem are formally described as below.

5.4.1 Optimum Location of Multiple Fermat-Weber Problems

Given a set of object groups $\mathbb{E} = \{G_1, ..., G_n\}$, where G_i ($|G_i| \geq 3$) contains points of a Fermat-Weber problem, a type weight functions ς^t and object weight functions σ, let l_j denote the optimal location of G_j under a stopping condition γ. The optimal location of $\mathbb{E}$ is a location $l \in \{l_j | 1 \leq j \leq n\}$ that minimizes WGD(l_j, G_j, ς^t, σ).

5.4.2 A Cost-Bound Approach

The cost-bound approach receives a set of object groups $\mathbb{E}$, a type weight function ς^t, object weight functions σ, and a stopping condition γ. Setting an error bound ϵ is one of the typical stopping conditions (see Section 2.3). The weights of the objects are indicated by ς^t. The distance from a location to points is calculated by their Euclidean distance and σ. The number of points in the Fermat-Weber problems ($|G_i|$) is unnecessarily fixed.

In Algorithm 5, the global lower bound, *Cbound*, is initialized to infinity and reduced to the minimum cost of the optimal location found so far. The algorithm sequentially checks the Fermat-Weber problems, each of which have a local optimal location found in lines 4-17. In the branch of

Algorithm 5 CostBoundApproach($\mathbb{E}$, ς^t, σ, γ)

```
1.  Cbound = ∞
2.  l = < 0, 0 >
3.  for Gi ∈ E do
4.    Initialize li to the center of Gi
5.    if |Gi| = 3 or Gi is a collinear case then
6.      Calculate the optimal location li of Gi
7.    else
8.      Let Gi =< p_1^u, ..., p_n^v >
9.      Calculate the optimal location l' of < p_1^u, p_2^s >
10.     if WGD(l', {p_1^u, p_2^s}, ς^t, σ) > Cbound then
11.       Continue
12.     end if
13.     repeat
14.       li = f (li, Gi)    /* Iterating, see Equation 8 */
15.       Lbound = lb(li)    /* see Equation 10 */
16.     until γ is satisfied or Lbound ≥ Cbound
17.   end if
18.   Cost = WGD(li, Gi, ς^t, σ)
19.   if Cbound > Cost then
20.     Cbound = Cost
21.     l = li
22.   end if
23. end for
24. return l
```

the iterative method inside the loop, an optimal location of the first two points in G_i is first detected in lines 8-12, as SSC solution does. If a better result of G_i potentially exists, a local lower bound is calculated in each iteration in line 15. If the local lower bound is greater than *Cbound*, the iteration will stop in line 16. The complexity of Algorithm 5 is $O(\mu \times |\mathbb{E}|)$, where μ denotes average number of iterations processed for Fermat-Weber problems. The Cost-bound approach can be used in the SSC solution as well.

6. EXPERIMENTAL VALIDATION

In this section, we evaluate the performance of the OVD model and proposed query solutions with real-world data sets. We implemented the proposed algorithms in C++. All data was loaded into the main memory during the execution of the simulations. All the experiments were conducted on a Red Hat Enterprise Linux server equipped with four Intel Xeon X5550 2.67 GHz processors and 24 GB of memory. All results were recorded after the system model reached a steady state.

In our experiments, the data sets were downloaded from GeoNames[1]. We retrieved the largest five object types, 230,762 streams (STM), 225,553 churches (CH), 200,996 schools (SCH), 166,788 populated places (PPL), and 110,289 buildings ($BLDG$), in the United States. By default, we set the type weight w^t and object weight w^o to 1. The multiplicatively-based weight functions are used as ς^t and σ. GPC library[2] is used for polygon overlapping calculations.

6.1 MOLQ Evaluation

We evaluate the solutions for MOLQ queries with three and four object types that are popular applications in the

[1]http://www.geonames.org/

[2]http://www.cs.man.ac.uk/∼toby/gpc/

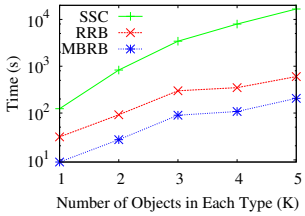


Figure 8: Three object types.

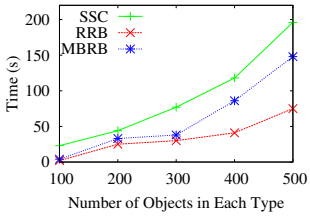


Figure 9: Four object types.

real world. The type weights are randomly generated from 0 to 10. We use the largest object types, $\mathbb{E}$={STM, CH, SCH} for the three-type case and $\mathbb{E}$={STM, CH, SCH, PPL} for the four-type case. The objects are randomly selected from the data sets.

Fig. 8 displays the performance of SSC, the proposed RRB and MBRB solutions. The cost-bound approach is used in all the three solutions. As Fig. 8 shows, RRB and MBRB run 4-28 times and 13-81 times faster than SSC, respectively, because they avoid a significant number of object combinations. Overlapping Voronoi diagrams is a process of filtering out combinations that cannot be the closest objects of any location. Another observation is that MBRB takes only 1/3 of the execution time of query processing in RRB. The evidence has been shown in Fig. 11 and 14; the benefit obtained by *MOVD Overlapper* in MBRB is greater than the overhead paid in *Optimizer*.

In the query with four object types, only approximate results can be provided by the three approaches. The error bound ϵ is set to be 0.001. Fig. 9 shows the execution time of the three solutions, in which the RRB solution has the best performance (2.8 times faster than SSC and 50% faster than MBRB, on average). Although the execution time of overlapping processing in the MBRB approach is slightly shorter than RRB as shown in Fig. 14(b), a large number of OVRs (21 times more than that in RRB) makes MBRB expensive in Fermat-Weber calculation, finding an optimal location in each OVRs. In the general cases, the overlapping processing takes nearly 90% of execution time in the query evaluation.

6.2 Cost-Bound Approach Evaluation

We evaluate the basic (Original) and cost-bound (CB) approaches by varying the number of Fermat-Weber problems and the error bound ϵ. The basic approach sequentially calculates the optimum locations of all Fermat-Weber problems, and selects the best location for the result. The number of points in each Fermat-Weber problem is fixed to 5. The coordinates and type weights (from 0 to 10) of points are randomly generated. The iterative method for a Fermat-Weber problem will stop when the deviation from the optimal cost is less than the error bound ϵ (see Section 2.3) [17].

Fig. 10 displays the execution time of the two approaches. As either the problem size increases or ϵ decreases, the execution time of both approaches rises. Obviously, the growth rate of the original approach is higher than the cost-bound approach because a significant number of unnecessary iterations can be avoided by setting a cost bound, which makes the cost-bound approach more efficient (4 to 34 times faster than the original approach), even though it has to pay extra overhead on lower bound calculations in each iteration.

6.3 Overlapping Two Voronoi Diagrams

Two overlap approaches, RRB and MBRB, on two regular Voronoi diagrams are evaluated with various data set sizes. The Voronoi diagrams are generated by two object sets, which are randomly selected from STM and CH. Their sizes are indicated by the x and y axes in Fig. 11-13.

From Fig. 11, we observe that the execution time of the overlapping processing in MBRB is shorter than that of RRB. In particular, the speedup of MBRB ranges from 16.3 times in two 10K data sets to 100.3 times in two 160K data sets. The reason is that the regions of OVRs generated by RRB are determined by real region overlapping calculation (polygon overlapping calculation in this experiment). The complexity of overlapping two polygons is proportional to the number of vertices in the polygons, which is more expensive than the MBR detection (rectangle overlapping calculation) that can be completed in constant time in MBRB. Also, Fig. 12 shows the evidence that due to replacing real regions of OVRs with their MBRs, MBRB generates around 150% more OVRs, on average, than RRB. Two OVRs that are not really overlapped with each other may be determined to be overlapped by the MBR detection. However, Fig. 13 shows that MBRB consumes 26%-29% less memory than RRB. Although MBRB generates more OVRs, the regions (MBRs) of which can be represented by just two points, all vertices of polygons have to be recorded in RRB. According to Fig. 13, the total number of points managed by the MBRB approach is smaller than RRB.

6.4 Overlapping Multiple Voronoi Diagrams

In this experiment, we examine the overlap operation by varying the number of Voronoi diagrams. These Voronoi diagrams are generated by objects randomly selected from $\mathbb{E} = \{STM, CH, SCH, PPL, BLDG\}$. For object type selection, we follow the sequence in $\mathbb{E}$ (*i.e.*, $\mathbb{E} = \{STM, CH\}$ for the two-type case, $\mathbb{E} = \{STM, CH, SCH\}$ for the three-type case, and so on). In addition to performance evaluation, we explore the availability of the overlap operation, which is described by the maximum size of objects in a particular number of object types that can be processed on the test bed. All data is assumed to be loaded into the main memory.

Fig. 14(a) demonstrates the availability of the overlap

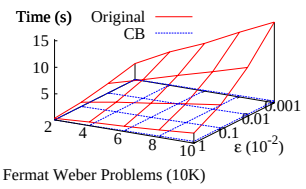


Figure 10: CB approach evaluation.

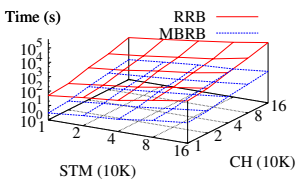


Figure 11: Execution time.

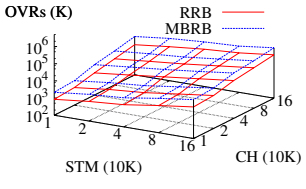


Figure 12: Number of OVRs.

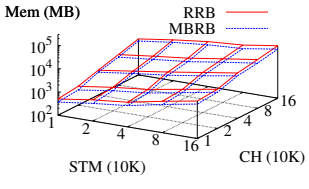
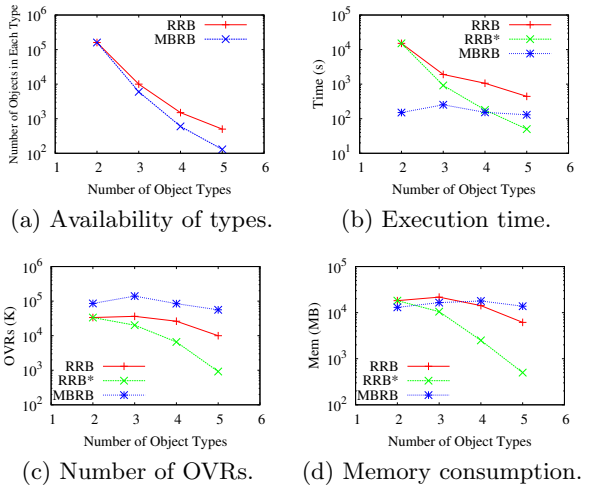


Figure 13: Memory consumption.



(a) Availability of types. (b) Execution time.

(c) Number of OVRs. (d) Memory consumption.

Figure 14: Varying number of object types.

operation by varying the number of object types. When the number of object types increases, the maximum numbers of objects in both RRB and MBRB approaches drop rapidly, from 10^5 objects in two types to 10^3 objects in four types. The more Voronoi diagrams overlap, the more OVRs are generated which requires more memory. Moreover, the dropping rate of the MBRB approach is higher than RRB because the MBRB approach consumes more memory when the number of object types is greater than three, as shown in Fig. 14(d).

Fig. 14(b), 14(c), and 14(d) display corresponding execution time, the number of OVRs, and memory consumption of both approaches with parameters that lie on the availability lines (in Fig. 14(a)). Due to different data sizes and the number of object types configured in the two groups of evaluation, we conduct another group of experiments that evaluate the RRB approach with the parameters used in MBRB evaluation for fair comparison. The experimental results are highlighted by RRB*.

As we expect, the MBRB approach always produces relatively larger set of OVRs than RRB*, as shown in Fig. 14(c). The more OVRs generated by MBRB than RRB* increase from 1.5 times in two object types, 21 times in four object types, to 74 times in five object types, since the false positives in the overlapping processing will be fed in the next overlapping processing, which generates more false positives. Moreover, in Fig. 14(b), when the number of object types is greater than 4, RRB* runs faster than the MBRB approach because the computation complexity induced by a surprisingly large number of OVRs dominates the entire process in the MBRB approach, which has a greater impact than the benefits obtained from the region overlapping calculation. In addition, a turning point in terms of memory consumption is observed between 2 and 3 in Fig. 14(d). When overlapping three or more Voronoi diagrams, the MBRB approach consumes more memory due to the large number of OVRs, in which the total number of points is more than the vertices managed by RRB*.

7. RELATED WORK

In this section, we review previous works related to reverse nearest neighbor queries and optimal location queries.

7.1 Reverse Nearest Neighbor Query

Korn and Muthukrishnan [11] proposed the influence set notion based on reverse nearest neighbor (RNN) queries. They presented a precomputation-based approach for solving RNN queries and an R-tree based method (RNN-tree) for large data sets. In order to decrease index maintenance costs in [11], Yang and Lin [27] presented the Rdnn-tree which combines the R-tree with the RNN-tree and leads to significant savings in dynamically maintaining the index structure. The solutions in [11, 27] can be employed to evaluate both the monochromatic RNN query and the bichromatic RNN query; however, these precomputation-based techniques incur extra maintenance costs for data updates. Therefore, several solutions without precomputation were proposed. For discovering influence sets in dynamic environments, Stanoi *et al.* [18] presented techniques to process bichromatic RNN queries without precomputation. The design is to dynamically construct the influence region of a given query point q where the influence region is defined as a polygon in space which encloses all RNNs of q. For the monochromatic RNN query, Tao *et al.* [19] developed algorithms for evaluating RkNN with arbitrary values of k on dynamic multidimensional data sets by utilizing a data-partitioning index. The algorithms were later extended to support continuous RkNN searches [20], which return the RkNN results for every point on a line segment.

There are some other works related to RNN query evaluation. Retrieving RNN aggregations (such as COUNT or MAX DISTANCE) over data streams was introduced in [12]. Yiu *et al.* [28] proposed pruning-based methods to find RNNs in large graphs. The algorithms for efficient RNN search in generic metric spaces were presented in [21]. The techniques require no detailed representations of objects and can be applied as long as the similarity between two objects can be computed and the similarity metric satisfies the triangle inequality. Cheema *et al.* [3] studied the problem of continuous monitoring of reverse k nearest neighbors queries in Euclidean space as well as in spatial networks. While the aforementioned approaches work well for R(k)NN queries, they cannot be utilized directly to evaluate the unique query type studied in this paper due to the fundamental differences between query definitions.

7.2 Optimal Location Query

One group of optimal location queries (OLQ) is defined with an optimization function which maximizes the influence of a facility. Given a set of sites, a set of weighted objects, and a spatial region Q, the optimal-location query defined in [6] returns a location in Q with a maximum influence based on the L_1 distance, where the influence of a location is the total weight of its RNNs. Xia *et al.* [25] proposed pruning techniques based on a metric named $minExistDNN$ to retrieve the top-t most influential sites according to the total weights of their RNNs inside a given spatial region Q. The Optimal Location Selection (OLS) search was introduced in [7], which retrieves target objects in a target object set D_B that are outside a spatial region R but have maximal optimality with a given data object set D_A and a critical distance d_c. Here, the optimality of a target object $b \in D_B$ located outside R is defined as the number of the data objects from D_A that are inside R and have their distances to b not exceeding d_c.

Another group of location optimization queries is defined with a different optimization function which minimizes the

average distance between a client and the nearest facility. Zhang *et al.* [29] proposed the Min-Dist Optimal Location Query (MDOLQ). Given a set S of sites, a set O of weighted objects, and a spatial region Q, MDOLQ returns a location for building a new site in Q, which minimizes the average distance from each object to its closest site according to the L_1 distance. They provide a progressive algorithm that quickly suggests a location, tells the maximum error the outcome may have, and continuously refines the result. When the algorithm finishes, the exact answer can be found. Because user movements are usually confined to underlying spatial networks in practice, Xiao *et al.* [26] extended OLQ to support queries on road networks. They designed a unified framework that addresses three variants of optimal location queries. By observing that users can only choose from some candidate locations to build a new facility in many real applications, Qi *et al.* [16] introduced the Min-dist Location Selection Query (MLSQ) based on the studies in [29, 26]. Given a set of clients and a set of existing facilities, MLSQ finds a location from a given set of potential locations for establishing a new facility where the average distance between a client and her nearest facility is minimized. MND, a method for efficiently solving MLSQ, employs a single value to describe a region that encloses the nearest existing facilities of a group of clients. The MND method is presented in [16]. However, these studies differ from the proposed query type in definition and optimization functions. Consequently, we cannot use them for answering our novel query type.

8. CONCLUSION

In this research, we formulated a novel optimal location selection problem. In addition to designing a straightforward approach that sequentially scans all object combinations, we propose an MOVD-based approach (RRB) that efficiently answers the query. Moreover, in order to minimize the costs induced by region overlapping, we propose the MBRB approach, in which MBRs are used as the boundaries of OVRs, since overlapping two rectangles is much cheaper than overlapping two arbitrary regions. In addition, a cost-bound iterative approach is proposed to efficiently process a large number of Fermat-Weber problems. We demonstrate the excellent performance of the proposed approaches through extensive simulations.

For the future work, we plan to optimize the proposed solutions by either using disk-based techniques that load a portion of data into the main memory, or pruning the search space by filtering out the impossible POI combinations during the MOVD overlapping.

9. REFERENCES

[1] J.-D. Boissonnat and C. Delage. Convex Hull and Voronoi Diagram of Additively Weighted Points. In *ESA*, pages 367–378, 2005.

[2] R. Chandrasekaran and A. Tamir. Algebraic Optimization: The Fermat-Weber Location Problem. *Math. Program.*, 46:219–224, 1990.

[3] M. A. Cheema, W. Zhang, X. Lin, Y. Zhang, and X. Li. Continuous reverse k nearest neighbors queries in euclidean space and in spatial networks. *VLDB J.*, 21(1):69–95, 2012.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[5] P. Dong. Generating and updating multiplicatively weighted Voronoi diagrams for point, line and polygon features in GIS. *Computers & Geosciences*, 34(4):411–421, 2008.

[6] Y. Du, D. Zhang, and T. Xia. The Optimal-Location Query. In *SSTD*, pages 163–180, 2005.

[7] Y. Gao, B. Zheng, G. Chen, and Q. Li. Optimal-Location-Selection Query Processing in Spatial Databases. *IEEE Trans. Knowl. Data Eng.*, 21(8):1162–1177, 2009.

[8] J. B. S. Haldane. Note on the Median of a Multivariate Distribution. *Biometrika*, 35:414–415, 1948.

[9] G. Jalal and J. Krarup. Geometrical Solution to the Fermat Problem with Arbitrary Weights. *Annals OR*, 123(1-4):67–104, 2003.

[10] M. I. Karavelas and M. Yvinec. Dynamic Additively Weighted Voronoi Diagrams in 2D. In *ESA*, pages 586–598, 2002.

[11] F. Korn and S. Muthukrishnan. Influence Sets Based on Reverse Nearest Neighbor Queries. In *SIGMOD Conference*, pages 201–212, 2000.

[12] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse Nearest Neighbor Aggregates Over Data Streams. In *VLDB*, pages 814–825, 2002.

[13] L. Mu. Polygon Characterization With the Multiplicatively Weighted Voronoi Diagram. *The Professional Geographer*, 56(2):223–239, 2004.

[14] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Probability and Statistics. Wiley, NYC, 2nd edition, 2000.

[15] F. Pagliara, J. Preston, and D. Simmonds. *Residential Location Choice: Models and Applications*. Springer, 2010.

[16] J. Qi, R. Zhang, L. Kulik, D. Lin, and Y. Xue. The Min-dist Location Selection Query. In *ICDE*, 2012.

[17] J. G. M. Robert F. Love and G. O. Wesolowsky. Facilities location, models and methods. 1988.

[18] I. Stanoi, M. Riedewald, D. Agrawal, and A. E. Abbadi. Discovery of Influence Sets in Frequently Updated Databases. In *VLDB*, pages 99–108, 2001.

[19] Y. Tao, D. Papadias, and X. Lian. Reverse kNN Search in Arbitrary Dimensionality. In *VLDB*, pages 744–755, 2004.

[20] Y. Tao, D. Papadias, X. Lian, and X. Xiao. Multidimensional reverse k NN search. *VLDB J.*, 16(3):293–316, 2007.

[21] Y. Tao, M. L. Yiu, and N. Mamoulis. Reverse Nearest Neighbor Search in Metric Spaces. *IEEE Trans. Knowl. Data Eng.*, 18(9):1239–1252, 2006.

[22] Y. Vardi and C.-H. Zhang. A modified Weiszfeld algorithm for the Fermat-Weber location problem. *Mathematical Programming*, 90:559–566, 2001.

[23] G. Weisbrod, M. Ben-Akiva, and S. Lerman. Tradeoffs in Residential Location Decisions: Transportation versus Other Factors. *Transportation Policy and Decision-Making*, 1(1), 1980.

[24] E. Weiszfeld and F. Plastria. On the point for which the sum of the distances to n given points is minimum. *Annals OR*, 167(1):7–41, 2009.

[25] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On Computing Top-t Most Influential Spatial Sites. In *VLDB*, pages 946–957, 2005.

[26] X. Xiao, B. Yao, and F. Li. Optimal location queries in road network databases. In *ICDE*, pages 804–815, 2011.

[27] C. Yang and K.-I. Lin. An Index Structure for Efficient Reverse Nearest Neighbor Queries. In *ICDE*, pages 485–492, 2001.

[28] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao. Reverse Nearest Neighbors in Large Graphs. *IEEE Trans. Knowl. Data Eng.*, 18(4):540–553, 2006.

[29] D. Zhang, Y. Du, T. Xia, and Y. Tao. Progressive Computation of the Min-Dist Optimal-Location Query. In *VLDB*, pages 643–654, 2006.