2013 IEEE 54th Annual Symposium on Foundations of Computer Science

On Clustering Induced Voronoi Diagrams

Danny Z. Chen
Department of Computer Science and Engineering
University of Notre Dame
Email: dchen@cse.nd.edu

Ziyun Huang Yangwei Liu Jinhui Xu
Department of Computer Science and Engineering
State University of New York at Buffalo
Email: {ziyunhua, yangweil, jinhui}@buffalo.edu

***Abstract*—In this paper, we study a generalization of the classical Voronoi diagram, called *clustering induced Voronoi diagram (CIVD)*. Different from the traditional model, CIVD takes as its sites the power set U of an input set P of objects. For each subset C of P, CIVD uses an *influence function* $F(C, q)$ to measure the total (or joint) *influence* of all objects in C on an arbitrary point q in the space $\mathbb{R}^d$, and determines the influence-based Voronoi cell in $\mathbb{R}^d$ for C. This generalized model offers a number of new features (*e.g.,* simultaneous clustering and space partition) to Voronoi diagram which are useful in various new applications. We investigate the general conditions for the influence function which ensure the existence of a small-size (*e.g.*, nearly linear) approximate CIVD for a set P of n points in $\mathbb{R}^d$ for some fixed d. To construct CIVD, we first present a standalone new technique, called *approximate influence (AI) decomposition*, for the general CIVD problem. With only $O(n \log n)$ time, the AI decomposition partitions the space $\mathbb{R}^d$ into a nearly linear number of cells so that all points in each cell receive their approximate maximum influence from the same (possibly unknown) site (*i.e.,* a subset of P). Based on this technique, we develop assignment algorithms to determine a proper site for each cell in the decomposition and form various $(1-\epsilon)$-approximate CIVDs for some small fixed $\epsilon > 0$. Particularly, we consider two representative CIVD problems, vector CIVD and density-based CIVD, and show that both of them admit fast assignment algorithms; consequently, their $(1-\epsilon)$-approximate CIVDs can be built in $O(n \log^{d+1} n)$ and $O(n \log^2 n)$ time, respectively.**

***Keywords*-Voronoi diagram; clustering; clustering induced Voronoi diagram; influence function; approximate influence decomposition;**

I. Introduction

Voronoi diagram is a fundamental geometric structure with numerous applications in many different areas [5], [6], [33]. The ordinary Voronoi diagram is a partition of the space $\mathbb{R}^d$ into a set of cells induced by a set P of points (or other objects) called sites, where each cell c_i of the diagram is the union of all points in $\mathbb{R}^d$ which have a closer (or farther) distance to a site $p_i \in P$ than to any other sites. There are many variants of Voronoi diagram, depending on the types of objects in P, the distance metrics, the dimensionality of the space, the order of Voronoi diagram, etc. In some sense, the cells in a Voronoi diagram are formed by competitions among all sites in $\mathbb{R}^d$, such that the winner site for any point q in $\mathbb{R}^d$ is the one having a larger influence on q defined by its distance to q.

A common feature shared by most known Voronoi diagrams is that the influence from every site object is independent from one another and does not combine together. However, it is quite often in real world applications that the influence from multiple sources can be "added" together to form a *joint influence*. For example, in physics, a particle q may receive forces from a number of other particles and the set of such forces jointly determines the motion of q. This phenomenon also arises in other areas, such as social networks where the set of actors (*i.e.*, nodes) in a community may have joint influence on a potential new actor (*e.g.,* a twitter account with a large number of followers may have a better chance to attract more followers). In such scenarios, it is desirable to identify the subset of objects which has the largest joint influence on one or more particular objects.

To develop a geometric model for joint influence, in this paper, we generalize the concept of Voronoi diagram to *clustering induced Voronoi diagram* (CIVD). In CIVD, we consider a set P of n points (or other type of objects) and a non-negative influence function F which measures the joint influence $F(C, q)$ from each subset C of P to any point q in $\mathbb{R}^d$. The Voronoi cell of C is the union of all points in $\mathbb{R}^d$ which receive a larger influence from C than from any other subset $C' \subseteq P$. This means that CIVD considers all subsets in the power set $U = 2^P$ of P as the sites (called *cluster sites*), and partitions $\mathbb{R}^d$ according to their influences. For some interesting influence functions, it is possible that only a small number of subsets in U have non-empty Voronoi cells. Thus the complexity of a CIVD is not necessarily exponential.

CIVD thus defined considerably generalizes the concept of Voronoi diagram. To our best knowledge, there is no previous work on the general CIVD problem. It obviously extends the ordinary Voronoi diagrams [6], where each site is a one-point cluster. (Note that the ordinary Voronoi diagrams are all special CIVDs equipped with proper influence functions.) Some Voronoi diagrams [33], [34] allow a site to contain multiple points, but the distance functions used are often defined by the closest (or farthest) point in such a site, not by a collective effect of all points of the site. The k-th order Voronoi diagram [33], where each cell is the union of points in $\mathbb{R}^d$ sharing the same k nearest neighbors

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.49

in P, may be viewed as having clusters of points as sites, and the "distance" functions are defined on all points of each site; but all cluster "sites" of a k-th order Voronoi diagram have the same size k and its "distance" function is quite different from the influence function in our CIVD problem. Some two-point site Voronoi diagrams were also studied [8], [9], [18], [19], [25], [28], [36], in which each site has exactly two points and the distance functions are defined by certain "combined" features of point pairs. In some sense, such Voronoi diagrams may be viewed as special CIVDs.

CIVD enables us to capture not only the spatial proximity of points, but more importantly their aggregation in the space. For example, a cluster site C with a non-empty Voronoi cell may imply that the points in C form a local cluster inside that cell. This provides an interesting connection between clustering and space partition and a potential to solve clustering and space partition problems simultaneously. Such new insights could be quite useful for applications in data mining and social networks. For instance, in social networks, clustering can be used to determine communities in some feature space, and space partition may allow to identify the nearest (or best-fit) community for any new actor. Furthermore, since each point in P may appear in multiple cluster sites with non-empty Voronoi cells, this could potentially help find all communities in a social network without having to apply the relatively expensive overlapping clustering techniques [1], [7], [10], [17] or to explicitly generate multiple views of the network [14], [16], [21], [32].

Of course, CIVD in general can have exponentially many cells, and an interesting question is what meaningful CIVD problems have a small number (say, polynomial) of cells. Thus, generalizing Voronoi diagrams in this way brings about a number of new challenges: (I) How to efficiently deal with the exponential number of potential cluster sites; (II) how to identify those non-empty Voronoi cells so that the construction time of CIVD is proportional only to the actual size of CIVD; (III) how to partition the space and efficiently determine the cluster site for each non-empty Voronoi cell in CIVD.

We consider in this paper the CIVD model for a set P of n points in $\mathbb{R}^d$ for some fixed d, aiming to resolve the above difficult issues. We first investigate the general sufficient conditions which allow the influence function to yield only a small number of non-empty approximate Voronoi cells. Our focus is thus mainly on the family of influence functions satisfying these conditions. We then present a standalone new technique, called *approximate influence decomposition* (or AI decomposition), for general CIVD problems. In $O(n \log n)$ time, this technique partitions the space $\mathbb{R}^d$ into a nearly linear number (*i.e.,* $O(n \log n)$) of cells so that for each such cell c, there exists a (possibly unknown) subset $C \subseteq P$ whose influence to any point $q \in c$ is within a $(1-\epsilon)$-approximation of the maximum influence that q can receive from any subset of P, where $\epsilon > 0$ is a fixed small constant. For this purpose, we develop a new data structure called *box-clustering tree*, based on an extended quad-tree decomposition and guided by a *distance-tree* built from the well separated pair decomposition [11]. In some sense, our AI decomposition may be viewed as a generalization of the well separated pair decomposition.

The AI decomposition partially overcomes challenges (II) and (III) above. However, we still need to assign a proper cluster site (selected from the power set U of P) to each resulting cell. To illustrate how to resolve this issue, we consider some important CIVDs and make use of both the AI decomposition and the specific properties of the influence functions of these problems to build approximate CIVDs. Particularly, we study two representative CIVD problems. The first problem is *vector CIVD* in which the influence between any two points p and q is a force-like vector (*e.g.*, gravity force) and the joint influence is the vector sum. Clearly, this problem can be used to construct Voronoi diagrams in some force-induced fields. The second problem is *density-based CIVD* in which the influence from a cluster C to a point q is the density of the smallest ball centered at q and containing C. This problem enables us to generate all density-based clusters as well as their approximate Voronoi cells. Since density-based clustering is widely used in many areas such as data mining, computer vision, pattern recognition, and social networks [12], [13], [15], [30], [35], we expect that the density-based CIVD is also applicable in these areas. For both these problems, we show efficient assignment algorithms that determine a proper cluster site for each cell in the AI decomposition in polylogarithmic time. Thus, $(1-\epsilon)$-approximate CIVDs for both problems can be constructed in $O(n \log^{d+1} n)$ and $O(n \log^2 n)$ time, respectively.

Since the conditions and the AI decomposition are all quite general and do not require to know the exact form of the influence function, we expect that our techniques will be applicable to many other CIVD problems.

It is worth pointing out that although significant differences exist, several problems/techniques can be viewed as related to CIVD. The first one is the *approximate Voronoi diagram or nearest neighbor* problem [2], [3], [4], [26], [27], [29], which shares with our approximate CIVD the same strategy of using regular shapes to approximate the Voronoi cells. However, since its sites are all single-point, it is thus quite different from our approximate CIVD problem. The second one is the *Fast Multipole Method (FMM)* for the N-body problem [22], [23], [24] which shares with the Vector CIVD a similar idea of modeling joint force by influence functions. The difference is that FMM mainly relies on simple functions (*i.e.,* kernels) to reduce the computational complexity, while Vector CIVD uses perturbation and properties of the influence function to achieve faster computation.

Due to space limit, all proofs are left to the full paper.

II. Overview of Approximate CIVD

In this section, we give an overview of the main ideas in computing approximate CIVDs. In subsequent sections, we will unfold the details of our ideas.

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in $\mathbb{R}^d$ for some fixed d, C be a subset of P, and q be an arbitrary point in $\mathbb{R}^d$ (called a *query point*). The influence from C to q is a function $F(C, q)$ on the vectors from every point $p \in C$ to q (or from q to p). Let $C_m(P, q) \subseteq P$ denote the cluster site which has the maximum influence, $F_{max}(q)$, on q, called the *maximum influence site* of q. Below we define the $(1-\epsilon)$-approximate CIVD induced by F.

Definition 1: Let $\mathcal{R} = \{c_1, c_2, \ldots, c_N\}$ be a partition of the space $\mathbb{R}^d$, and $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$ be a set (possibly multiset) of cluster sites of P. The set of pairs $\{(c_1, C_1), (c_2, C_2), \ldots, (c_N, C_N)\}$ is a $(1-\epsilon)$-approximate CIVD with respect to the influence function F if for each $c_i \in \mathcal{R}$, $F(C_i, q) \geq (1-\epsilon)F_{max}(q)$ for any point $q \in c_i$, where $\epsilon > 0$ is a small constant. Each c_i is an *approximate Voronoi cell*, and C_i is the *approximate maximum influence site* of c_i.

From the above definition, we know that to compute an approximate CIVD, there are two major tasks: (1) partition $\mathbb{R}^d$ into a set $\mathcal{R} = \{c_1, c_2, \ldots, c_N\}$ of cells, and (2) determine $C_i \subseteq P$ for each c_i. We call task (2) the *assignment problem*, which finds an approximate maximum influence site C_i from the power set U of P for each cell c_i of $\mathcal{R}$. Since the choice of C_i often depends on the properties of the specific influence function, we need to develop a specific assignment algorithm for each CIVD problem. In Sections V and VI, we present efficient assignment algorithms for two CIVD problems.

We call task (1) the *space partition problem*. For this problem, we develop a standalone technique, called *Approximate Influence (AI) Decomposition*, for general CIVDs. The size of a CIVD (or an approximate CIVD) in general can be exponential. Thus, in Section III, we investigate the general and sufficient conditions that ensure the existence of a small-size approximate CIVD. The AI decomposition makes use of only these general conditions and need not know the exact form of the influence function.

Roughly speaking, the general conditions aim to achieve simultaneously two objectives at the resulting cells of the space partition: (a) Cells that are far away from the input points should be of as "large" sizes as possible (where "far away" means that the size of the cell is small comparing to the distance from the cell to the nearest input point), and (b) cells that are close to the input points should not be too small. Each objective helps reduce the number of cells in the space partition from a different perspective.

Corresponding to the two objectives above, the AI decomposition presented in Section IV partitions $\mathbb{R}^d$ into two types of cells: type-1 cells and type-2 cells. Type-1 cells are those close to some input points (*i.e.,* corresponding to objective (b)), and type-2 cells are those far away from the input points (*i.e.,* corresponding to objective (a)). The AI decomposition has the following properties.

1) The space $\mathbb{R}^d$ is partitioned (in $O(n \log n)$ time) into a total of $O(n \log n)$ type-1 and type-2 cells.
2) A type-1 cell c is a box region (*i.e.,* an axis-aligned hypercube) or the difference of two box regions, and is associated with a known approximate maximum influence site C.
3) A type-2 cell c is a box region with a diameter of $D(c) \leq 2r\lambda_1(\epsilon)/3$, where r is the minimum distance between c and any point in P. All points in a type-2 cell c share a (not yet identified) cluster site $C \subseteq P$ as their common approximate maximum influence site.

As stated above, every type-1 cell in the AI decomposition is associated with a known approximate maximum influence site. Thus, our assignment algorithms only need to focus on determining approximate maximum influence sites for the type-2 cells.

III. Influence Function

In this section, we discuss the general conditions for the influence function to yield a small-size approximate CIVD.

By the definition of CIVD, for a given influence function, it is possible that most cluster sites in 2^P have a non-empty Voronoi cell, and hence the resulting CIVD is of exponential size. Fortunately, many influence functions in applications have certain good properties that induce CIVDs of much smaller sizes. Thus, it is desirable to understand how an influence function affects the size of the corresponding CIVD. For this purpose, we investigate the general and sufficient conditions of the influence functions which allow to yield a small-size (approximate) CIVD.

Note that since an influence function can be arbitrary, we shall focus on its general properties rather than its exact form. We will make some reasonable and self-evident assumptions about the influence function.

Let q be an arbitrary query point in $\mathbb{R}^d$ and C be a subset of P. The influence from C to q is defined as follows.

Definition 2: The influence from C to q is a function $F(C, q)$ satisfying the following condition: $F(C, q) = f(G(C, q))$, where $G(C, q) = \{p - q \mid p \in C\}$ is the multiset of vectors defined by C and q and $f(\cdot)$ is a non-negative function defined over all possible multisets of vectors in $\mathbb{R}^d$. For convenience, f is also called the influence function.

In the above definition, the influence depends solely on the set of vectors pointing from q to each point $p \in C$ or from p to q. This implies that the influence of C on q remains the same under translation.

The influence function is also required to have good properties on scaling and rotation, as follows.

Property 1 (Similarity Invariant): Let ϕ be a transformation of scaling or rotation about q, and C be any set (possibly

multiset) of points in $\mathbb{R}^d$. The ratio $F(\phi(C), q)/F(C, q)$ (if $F(C, q) \neq 0$) only depends on ϕ.

The above property implies that the maximum influence site $C_m(P, q)$ of q remains the same under any similarity transformation (including translation, scaling, and rotation about q). Thus Property 1 is also called the *similarity invariant* property, and is necessary for the following locality property.

As discussed in the last section, to ensure a small-size approximate CIVD, we expect that the cells (in the CIVD) that are far away from the input points should be "large" (*i.e.,* objective (a)) and the cells that are close to the input points should not be too small (*i.e.,* objective (b)). This means that many spatially close points in $\mathbb{R}^d$ would have to share the same approximate maximum influence site, which implies that the influence function must have a certain degree of locality (to achieve objective (a)). Below we give the precise meaning of the locality property.

Definition 3: Let q be a point and C be a set (possibly multiset) of points in $\mathbb{R}^d$. A 1-to-1 mapping ψ from C to $\psi(C)$ in $\mathbb{R}^d$ is called an *ϵ-perturbation* with respect to q if $\|p - \psi(p)\| \leq \epsilon \|p - q\|$ for every point $p \in C$, where $0 < \epsilon < 1$ is the *error ratio* and q is the *witness point* of ψ.

Intuitively speaking, from the witness point's view, an ϵ-perturbation only changes slightly the position of a point it maps from.

Definition 4: Let q be a point and C be a set (possibly multiset) of points in $\mathbb{R}^d$. For any γ with $0 < \gamma < 1$, let δ be a continuous and monotone function with $\delta(\gamma) < 1$ and $\lim_{x \to 0} \delta(x) = 0$. An influence function F is said to be (δ, γ)-*stable* at (C, q) if for any ϵ-perturbation C' of C with $\epsilon \leq \gamma < 1$, $(1 - \delta(\epsilon))F(C, q) \leq F(C', q) \leq (1 + \delta(\epsilon))F(C, q)$.

In the above definition, (C, q) is called a (δ, γ)-stable pair or simply stable pair.

Definition 5: Let C be a set (possibly multiset) of points in $\mathbb{R}^d$, q be a query point, and F be the influence function. (C, q) is a *maximal pair* of F if for any subset C' of C, $F(C', q) \leq F(C, q)$.

From the above definition, we know that any maximum influence site and its corresponding query point form a maximal pair. Since each maximal pair could potentially correspond to a non-empty Voronoi cell and any locality requirement on the influence function has to ensure stability on all Voronoi cells, it is sufficient to define the locality based on the stability of all maximal pairs.

Property 2 (Locality): The influence function F is (δ, γ)-stable at any maximal pair (C, q) for some monotone function δ and a small constant $0 < \gamma < 1$.

The locality property above means that a small perturbation on P only changes slightly the maximum influence on a query point q. This implies that we can use the perturbed points of P to determine an approximate maximum influence site for each q. The following lemma further shows that a good approximation of the maximum influence site for q is still a good approximation after an ϵ-perturbation.

Lemma 1: Let F be any influence function satisfying Property 2, and ψ be an ϵ-perturbation on P (with a witness point q and $\epsilon \leq \gamma$). Let C be any subset of P with influence $F(C, q) \geq (1 - \epsilon)F_{max}(q)$. If F is (δ, γ)-stable at (C, q), then there exist a constant $\epsilon < \gamma' < 1$ and a monotone function Δ with $\Delta(\gamma') < 1$ and $\lim_{x \to 0} \Delta(x) = 0$ such that $F(C', q) \geq (1 - \Delta(\epsilon))F'_{max}(q)$, where $P' = \psi(P)$, $C' = \psi(C)$, and $F'_{max}(q) = F(C_m(P', q), q)$.

In the above lemma, the error caused by the perturbation can be estimated by the function Δ. Thus Δ is also called the *error estimation function*. Since Δ is a monotone function around 0, for a sufficiently small value $\epsilon > 0$, $\Delta^{-1}(\epsilon)$ exists (this fact will be used later on). For ease of analysis, we assume that ϵ is sufficiently small so that $\Delta^{-1}(\epsilon) < 1/2$.

By Property 2, we know that the locality of an influence function is defined based on perturbation. Since perturbation uses relative error, locality is not uniform throughout the entire space. Such non-uniformity enables us to achieve objective (a) (discussed in Section II), but does not help attain objective (b). This means that if the influence function has only the locality property, then two query points, say q_1 and q_2, close to an input point p and having a distance larger than $2\epsilon \max\{\|p - q_1\|, \|p - q_2\|\}$ cannot be grouped into the same Voronoi cell. Since there are infinitely many query points arbitrarily close to p, we would need an infinite number of Voronoi cells to approximate their influences. Thus some additional property is needed to obtain a small-size CIVD (*i.e.,* to achieve objective (b)).

To get around this problem, we may imagine a situation that when a query point q is very close to a subset $C \subseteq P$, it is reasonable to assume that the influence from C completely "dominates" the influence from all other points in $P \setminus C$. This means that when determining the influence for q, we can simply ignore all points in $P \setminus C$, without losing much accuracy. This suggests that the influence function should also have the following *Local Domination* property.

Property 3 (Local Domination): There exists a polynomially bounded function $\mathcal{P}(\cdot)$ such that for a point q in $\mathbb{R}^d$ and any subset $P' \subseteq P$, if there is a point $p \in P'$ with $\mathcal{P}(n)\|q - p\| < \epsilon \cdot \|q - p'\|$ for all $p' \in P \setminus P'$, then $F'_{max}(q) > (1 - \epsilon)F_{max}(q)$, where $F'_{max}(q) = F(C_m(P', q), q)$.

The above property suggests that there is a dominating region for each input point p of P. Consider a ball centered at p and with a radius $\frac{\epsilon\|p - p'\|}{2(\mathcal{P}(n) + \epsilon)}$, where p' is the nearest neighbor of p in P. By Property 3, for any query point q inside this ball, its influence mainly comes from p.

The above properties and Lemma 1 suggest a way to construct an approximate CIVD. By Property 2, we know that it suffices to use a perturbation of P to construct an approximate CIVD. Since the influence function considers only the vectors between a query point q and the input points of P, we can equivalently perturb all query points

(i.e., the entire space $\mathbb{R}^d$), instead of the input points, and still obtain an approximate CIVD. This means that we can first approximate the space by partitioning it into small enough regions, and then associate each such region with a cluster site having an (approximate) maximum influence on it. The set of regions together forms an approximate CIVD. During the partition process, we use Property 3 to avoid generating regions of too small sizes, hence preventing a large number of regions. This leads us to the approximate influence decomposition.

IV. Approximate Influence Decomposition

In this section, we present a general space-partition technique called *approximate influence (AI) decomposition* for constructing an approximate CIVD. We assume that the influence function satisfies the similarity invariant, locality, and local domination properties.

To build an approximate CIVD, we utilize the locality and local domination properties to partition the space $\mathbb{R}^d$ into two types of cells (*i.e.,* type-1 and type-2 cells). Our idea for partitioning $\mathbb{R}^d$ is based on a new data structure called *box-tree*, which is constructed by an extended quadtree decomposition and guided by another new data structure called *distance-tree* built by the well separated pair decomposition [11]. Roughly speaking, the box-tree construction begins with a big enough bounding box of the input point set P (*i.e.,* an axis-aligned hypercube), recursively partitions each box into smaller boxes, and stops the recursion on a box when a certain condition is met. There are two types of boxes in the partition: One is a box generated by the normal quadtree decomposition, and the other involves the intersection or difference of two boxes. The stopping condition for a box B is that either B is small enough comparing to its distance to the closest point in P (and hence can be viewed as a type-2 cell), or B is inside the dominating region of some cluster site $C \subseteq P$ (and thus B can be viewed as a type-1 cell). For the first case, by Property 2, we know that all points in B can be viewed as perturbations of a single query point and hence share the same approximate maximum influence site. For the second case, by Property 3, we know that the approximate maximum influence site for all points in B is C. During the above space-partition process, a box becomes a *cell* if no further decomposition of it is needed.

In order for the resulted cells to have the desired properties (see Section II), we need to overcome a number of difficulties: (1) *How to efficiently maintain the distances between all potential cells (*i.e., *the boxes) and the input points in P so that their types can be distinguished?* (2) *How to efficiently generate the two types of cells?* (3) *How to bound the total number of cells and the running time of the space-partition process?* Below we discuss our ideas for resolving these difficulties.

For Difficulty (1), we know that the type of a cell is determined mainly by its distance to the input points in P. Thus, corresponding to the two types of cells, we need to maintain two types of distances for a box B generated by the space-partition process: (i) the distance, denoted by r_{min}, between B and the closest input point, and (ii) the distance, denoted by r_c, between B and the second closest input point or cluster site. A straightforward way to maintain such information is to explicitly determine the values of r_{min} and r_c for each generated box B. But, this would be rather inefficient because the number of possible different values to check could be very large. A seemingly possible way for this problem is to keep track of only the distances between B and the closest and second closest input points. This means that we consider only the dominating region of a single input point (*i.e.,* only checking whether B is in the dominating region of its closest input point). Unfortunately, this could cause the space-partition process to generate unnecessarily many boxes.

To understand this, consider an example in the 2D space which has only three input points, $(0, 0)$, $(1, 0)$, and $(M, 0)$, for some large number M. The size of the dominating region of $(1, 0)$ is small since its nearest neighbor is $(0, 0)$. The space between $(1, 0)$ and $(M, 0)$ is then decomposed into many (small) boxes in order to generate small enough boxes that are fully contained in the dominating region of $(1, 0)$.

One way to avoid this pitfall is that when a subset C of P is far away from other points of P, we treat C as a single point. In the above example, we may view $(0, 0)$ and $(1, 0)$ as forming a "heavy" point (with a "weight"). The dominating region size is then based on the distance between the "heavy" point and $(M, 0)$, which is significantly larger than 1. In this manner, we can reduce the total number of boxes.

To implement the idea, we employ a data structure called *distance-tree*, in which every node stores the location of one input point together with a value (whose exact meaning will be discussed later).

For easy of analysis, we assume the error tolerance $\beta < \frac{1}{2}$. Below are the main steps of our preprocessing algorithm (**Algorithm 1**).

Note that in **Algorithm 1**, $G(W)$ forms a spanner of P with a stretch factor of 2 [11]. In the resulting distance-tree T_p, each node v (called a *distance-node*) is associated with a point set, P_v, with a diameter upper-bounded by $s(v)$ and with $l(v)$ as its *representative point*. P_v is the subset of input points in P associated with the leaves of the subtree of T_p rooted at v. When a query point q is far away from P_v, each point in P_v can be viewed as a perturbation of any others. Thus, it will not cause too much error if we simply treat them as one "heavy" point, represented by $l(v)$. $E(v)$ gives the boundary for the query point q, *i.e.*, when q is outside $E(v)$, it is safe to view P_v as a single point (*i.e.,* when q is outside the bounding box $E(v)$, q is viewed as **far away** from P_v). $E'(v)$ is defined only for analysis purpose.

After constructing T_p, we now discuss our idea for

Algorithm 1 Preprocessing(P, β)

Input: A set P of n points in $\mathbb{R}^d$, and an error tolerance $0 < \beta < 1/2$.

Output: A tree structure T_p, in which every node v stores a value $s(v)$, an input point $l(v)$, and is associated with a bounding box $E(v)$ in $\mathbb{R}^d$.

1: Compute a 12-well separated pair decomposition $W = \{(A_1, B_1), (A_2, B_2), \ldots, (A_{m_0}, B_{m_0})\}$ of P.
2: Construct a graph $G(W)$ by connecting the representatives of A_i and B_i, for every $(A_i, B_i) \in W$.
3: Build a min-priority queue Q for all edges in $G(W)$, base on their edge lengths.
4: Build a tree T_p in the following bottom-up manner.

For each $p \in P$, there is a leaf node v_p in T_p (*i.e.*, T_p is initially a forest of $|P|$ single-node trees), with $s(v_p) = 0$, $l(v_p) = p$, and $E(v_p)$ and $E'(v_p)$ both being 0-sized bounding boxes containing p.

While T_p is not a single tree **Do**

- Extract from Q the shortest edge $e = (p_1, p_2)$ with edge length $w(e)$. If v_{p_1} and v_{p_2} are leaves of two different trees in T_p rooted at v_1 and v_2, then create a new node v in T_p as the parent of v_1 and v_2, and let $s(v) = s(v_1) + s(v_2) + w(e)$, $l(v)$ be either $l(v_1)$ or $l(v_2)$, $E'(v)$ be the box centered at $l(v)$ and with size $\frac{4 \cdot s(v)}{\beta}$, and $E(v)$ be the box centered at $l(v)$ and with size $\frac{8 \cdot s(v)}{\beta}$.

efficiently building the *box-tree* T_q (*i.e.,* Difficulty (2)).

To build T_q, consider an arbitrary box-node u of T_q. The key question regarding u is to determine whether its associated box $B(u)$ should be further decomposed. To resolve this issue, we maintain for u a list $L = \{v_1, v_2, \ldots, v_k\}$ of distance-nodes in T_p (inherited from u's parent or predetermined if u is the root of T_p). Each distance-node $v_i \in L$ is associated with a subset P_{v_i} of input points which may possibly give arise to the distance r_{min} for $B(u)$ (and also possibly the distance r_c). The value of r_c is recursively maintained to approximate the closest distance from $B(u)$ to all points in $P \setminus \cup_{i=1}^{k} P_{v_i}$ (*i.e.,* all input points not in L).

To determine whether $B(u)$ should be decomposed, we examine all distance-nodes in L. There are three cases to consider for each $v_i \in L$. The first case is that the bounding box $E(v_i)$ of v_i significantly overlaps with $B(u)$. In this case, the region $B(u) \cap E(v_i)$ is not far away from P_{v_i} and we cannot view P_{v_i} as a single "heavy" point. This means that we cannot use $l(v_i)$ (*i.e.,* the representative point of P_{v_i}) to compute the value of r_{min}. To handle this case, we replace v_i in L by its two children, say $v_{i,1}$ and $v_{i,2}$, in the distance-tree T_p. This can potentially increase the distance between $B(u)$ and each of $P_{v_{i,1}}$ and $P_{v_{i,2}}$, and hence enhance the chance for $B(u)$ to be far away from these two child nodes.

The second case is that $B(u)$ is far away from P_{v_i}. For this case, we remove v_i from L and save its distance (to $B(u)$) in r_c if it is smaller than the current value of r_c. If all distance-nodes are removed from L in this way, then it means that $B(u)$ is far away from all input points and therefore becomes a type-2 cell. In this case, the value of r_{min} is the value of r_c at the time when L becomes empty.

The third case is that v_i does not fall in any of the above two cases. In this scenario, if v_i is the only distance-node left in L and $B(u)$ (or part of $B(u)$) is inside the dominating region of P_{v_i}, then the part of $B(u)$ outside $E(v_i)$ becomes a type-1 cell, and the part of $B(u)$ inside $E(v_i)$ will be recursively determined for its decomposition. Otherwise, either more than one distance-node are still in L or $B(u)$ is not in the dominating region of P_{v_i}. For both cases, we decompose $B(u)$ (or the remaining part of $B(u)$) into 2^d sub-boxes and recursively process each sub-box.

To generate the box-tree T_q, we use a recursive algorithm called *AI-Decomposition*, in which $\mathcal{P}(\cdot)$ is the polynomially bounded function in Property 3. The core of this algorithm is a procedure, called *Decomposition*, which produces the box-subtree rooted at a box-node u that is part of the input to the procedure. In the procedure Decomposition, Step 1 corresponds to the first case; Steps 2 and 3 are for the second case; Steps 4 and 5 handle the third case.

Algorithm 3 AI-Decomposition(P, β)

Input: A set P of n points in $\mathbb{R}^d$, and a small error tolerance $\beta > 0$.

Output: A box-tree T_q.

1: Run the preprocessing algorithm on P and obtain a distance-tree T_p. Let u be the root of T_p. View $E(u)$ as a box-tree node. Run Decomposition($E(u), \beta, \{u\}, T_p, \infty$).
2: Output the box-tree rooted at $E(u)$ as T_q.

A. Algorithm Analysis (Difficulty (3))

We show some interesting properties of the T_p as well as the correctness and running time.

Definition 6: A distance-node $v \in T_p$ is said to be *recorded* for a box-node u if v is removed from the list L in Step 2.2 of **Algorithm 2** when processing u or one of u's ancestors. The value of r_{min} in the iteration when v is removed from L is the *recorded distance* of v for u. If v is recorded for u, then any point $p \in P_v$ is also *recorded* for u with the same recorded distance as v.

Lemma 2: If $p \in P$ is recorded for a box-node u with a recorded distance x, then for any point $q \in B(u)$,

$$(1 - \beta)x \leq \|p - q\| \leq (1 + \beta)x.$$

The following two lemmas show some important properties of the type-2 cells.

Algorithm 2 Decomposition(u, β, L, T_p, r_c)

Input: A box-node u with box $B(u)$, error tolerance $\beta > 0$, distance-tree T_p, linked list L, and a value r_c. **Output:** A subtree of T_q rooted at u.

1: **While** $\exists$ v in L such that the length of at least one edge of $B(u) \cap E(v)$ is no smaller than $\frac{size(B(u))}{2}$ **do**
- Replace v in L by its two children in T_p, if any.

2: Let $D(u)$ be the diameter of $B(u)$. For each node v in L **do**
- 2.1 Let r_{min} be the distance between $B(u)$ and $l(v)$.
- 2.2 If $D(u) < r_{min}\beta/2$, remove v from L, and if $r_c > r_{min}$, let $r_c = r_{min}$.

3: If L is empty, return, and $B(u)$ becomes a **type-2** cell.

4: If there is only one element v in L, let r_{min} be the smallest distance between $l(v)$ and $B(u)$.
- 4.1 If $\frac{r_{min}+D(u)}{r_c} < \frac{\beta}{2\mathcal{P}(n)}$,
  - 4.1.1 If $E(v) \cap B(u) = \phi$ or v is a leaf node in T_p, $B(u)$ is a **type-1** cell dominated by v. Return.
  - 4.1.2 Let B' be the smallest hyper-cubic box in $B(u)$ fully containing $B(u) \cap E(v)$. Create two box nodes u_0 and u_1, with u_0 corresponding to B' and u_1 corresponding to the difference of $B(u)$ and B'. Let u_0 and u_1 be children of u in T_q. In this case, u_1 is a **type-1** cell dominated by v.
  - 4.1.3 Replace v in L by its two children v_1 and v_2 in T_p. Call Decomposition$(u_0, \beta, L, T_p, r_c)$, and return.

5: Decompose $B(u)$ into 2^d smaller boxes, and make the corresponding nodes $u_1, u_2, \ldots, u_{2^d}$ as the children of u in T_q. Call Decomposition$(u_i, \beta, L, T_p, r_c)$ for each u_i. Return.

Lemma 3: For any type-2 cell c produced by **Algorithm 3**, the set of distance-nodes (also viewed as subsets of the input points) recorded for c forms a partition of P.

Lemma 4: For any type-2 cell c and $p \in P$, let $D(c)$ be the diameter of c and r be the shortest distance between c and p. Then

$$D(c) \leq \frac{2r\beta}{3}.$$

The lemma below characterizes the type-1 cells.

Lemma 5: If c is a type-1 cell dominated by a distance-node v, then for any $q \in c$ and $p' \in P \setminus P_v$,

$$\frac{\|q - l(v)\|}{\|q - p'\|} \leq \frac{\beta}{\mathcal{P}(n)}.$$

The following definition is mainly for Theorem 1.

Definition 7: In $\mathbb{R}^d$, let C be a set of k coincident points and q be any query point. The *maximum duplication function* ρ for an influence function F satisfying Property 1 is defined as $\rho(k) = |C_m(C, q)|$ (*i.e.*, the cardinality of $C_m(C, q)$). For any set C' of k points in $\mathbb{R}^d$ (not necessarily coincident points), the *selection mapping* η maps C' to an arbitrary subset $\eta(C')$ of C' with cardinality $\rho(k)$.

Note that in the above definition, it is possible that, for some influence function F, the maximum influence of a set C of k coincident points on a query point q is attained by a subset of C. By Property 1, we know that $\rho(k)$ depends only on the influence function F and is independent of C and q.

The following theorem ensures that all points in each cell generated by the AI decomposition have a common approximate maximum influence site (*i.e.,* the correctness of the AI decomposition).

Theorem 1: Let c be any cell generated by the AI-Decomposition algorithm with an error tolerance $\beta = \Delta^{-1}(\epsilon)$, where Δ is the error estimation function. Then the following holds.

1) If c is a type-1 cell dominated by a distance-node v, then $F(\eta(P_v), q) \geq (1 - \epsilon)F(C_m(P, q), q)$ for any query point q in c.
2) If c is a type-2 cell and q' is an arbitrary point in c, then $F(C_m(P, q'), q) \geq (1-\epsilon)F(C_m(P, q), q)$ for any $q \in c$. Furthermore, if there exists a subset $C \subseteq P$ such that $F(C, q') \geq (1 - \beta)F(C_m(P, q'), q')$ and (C, q') is a stable pair, then $F(C, q) \geq (1-\epsilon)F(C_m(P, q), q)$ for any q in c.
3) For any query point q outside the bounding box $B(u_{root})$, $F(\eta(P_{v_{root}}), q) \geq (1 - \epsilon)F(C_m(P, q), q)$, where u_{root} is the root of T_q and v_{root} is the root of T_p.

The following packing lemma is a key to bounding the total number of type-1 and type-2 cells and the running time of the AI decomposition (*i.e.,* Theorem 2).

Lemma 6 (Packing Lemma): Let o_c be any point in $\mathbb{R}^d$, and S_{in} and S_{out} be two d-dimensional boxes (*i.e.,* axis-aligned hypercubes) co-centered at o_c and with edge lengths $2r_{in}$ and $2r_{out}$, respectively, with $0 < r_{in} < r_{out}$. Let $\mathcal{B}$ be a set of mutually disjoint d-dimensional boxes such that for any $B \in \mathcal{B}$, B intersects the region $S' = S_{out} - S_{in}$ (i.e., the region sandwiched by S_{in} and S_{out}) and its edge length $L(B) \geq C \cdot r$, where r is the minimum distance between B and o_c and C is a positive constant. Then $|\mathcal{B}| \leq C'(C, d) \log(r_{out}/r_{in})$, where $C'(C, d)$ is a constant depending only on C and d.

Theorem 2: For any set of n input points in $\mathbb{R}^d$ and an influence function F satisfying the three properties in Section III, the AI-Decomposition algorithm yields $O(n \log n)$ type-1 and type-2 cells in $O(n \log n)$ time, where the constant hidden in the big-O notation depends on the error tolerance β and d.

V. Density-based CIVD

In this section, we show how to augment the AI decomposition algorithm to generate a $(1 - \epsilon)$-approximate CIVD

for the density-based CIVD problem.

The density-based CIVD problem for a set P of n points in $\mathbb{R}^d$ is to partition the space into cells so that all points in each cell share the same subset C of P as their *densest cluster*. For a given query point $q \in \mathbb{R}^d$, the densest cluster $C_m(P,q)$ of q is the subset C of P which maximizes the influence $F(C,q) = |C|/V(C,q)$ over all subsets of P, where $V(C,q) = \frac{\pi^{\frac{d}{2}} l^d}{\Gamma(\frac{d}{2}+1)}$ is the volume of the smallest ball centered at q and containing all points in C, l is the maximum distance from q to any point in C, and Γ is the gamma function. In other words, $C_m(P,q)$ is the cluster with the highest density around q.

Clearly, density-based CIVD is closely related to the widely used density-based clustering problem [12], [13], [15], [30], [35]. Since density-based clustering is used in many data mining, pattern recognition, medical imaging, and social network applications, we expect that the density-based CIVD is also applicable in these areas.

The following theorem shows that the problem satisfies the three properties in Section III.

***Theorem** 3:* The density-based CIVD Problem satisfies the three properties in Section III.

The above theorem indicates that AI Decomposition can be used to build an approximate density-based CIVD. To solve the associated assignment problem, our idea is to modify the AI-Decomposition algorithm (**Algorithm 3**) so that some additional information is maintained for us to assign a cluster to each resulting type-2 cell. (Note that by Theorem 1, for each type-1 cell c, we can simply use the distance-node v which dominates it as its densest cluster.) In this way, we can obtain the approximate CIVD at the same time of completing the AI decomposition.

Recall that an input point p is recorded in the AI-Decomposition algorithm only when its distance to the current to-be-decomposed box is large enough. Therefore, for a cell c, it is most likely that an input point recorded earlier is farther away from c than an input point recorded later. Intuitively, recorded distances (of input points) should be roughly in a decreasing order with respect to the order in which they are recorded. Below we discuss how to use this observation to modify the AI-Decomposition algorithm.

To understand how to modify the AI-Decomposition algorithm, we first consider an example. Let q be a query point, and $P = \{p_1, p_2, \ldots, p_n\}$ be a set of input points in the decreasing order of their distances to q (*i.e.*, $\|p_i - q\| > \|p_j - q\|$ for all $1 \le i < j \le n$, and no two points have the same distance to q). To find $C_m(P,q)$, we can use the following approach which scans P only once in its sorted order and uses $O(1)$ additional space. For each $1 \le i \le n$, we compute $D_i = \frac{c_d(n-i+1)}{\|p_i-q\|^d}$, and store the largest D_i during the scanning process, where $c_d = \frac{\Gamma(\frac{d}{2}+1)}{\pi^{\frac{d}{2}}}$. Since the ball centered at q with radius $\|p_i - q\|$ contains exactly $n-i+1$ points, $\{p_i, p_{i+1}, \ldots, p_n\}$, the largest D_i value, along with the corresponding i value, gives us the desired densest cluster $C_m(P,q) = \{p_i, p_{i+1}, \ldots, p_n\}$.

With the above understanding, we can now modify the Decomposition algorithm (**Algorithm 2**) as follows. Particularly, we change Step 2 of the Decomposition algorithm, since this is the step distance-nodes are removed from L. Before the execution of Step 2, we sort the distance-nodes in L by the decreasing order of their distances to the current box-node u (if there are nodes with the same distance, we order them arbitrarily). Then, we execute Step 2 and try to remove distance-nodes from L according to this sorted order. We assume that in the Decomposition algorithm, a number M is maintained for storing the total number of input points which are recorded for the current box-node u. During the execution of Step 2, after removing each distance-node v, we compute a value $D = \frac{c_d(n-M)}{r^d}$ and then update M (*i.e.,* increase M by the cardinality $|P_v|$ of v). We save the largest value D along with the corresponding distance-node v and the box-node u. In each recursive call to the Decomposition algorithm, we pass the stored D, u, and v to the next level of the recursion.

Clearly, such a modification on Step 2 of the Decomposition algorithm resembles the computation in the above example. The only difference is that in the above example, input points are considered strictly in the decreasing order of their distances to the query point q, but in the modified Decomposition algorithm distance-nodes are not always removed by the decreasing order of their distances to some query point q. This is because at different recursion levels, distance-nodes may not be removed in a strictly decreasing order. Below we show that despite such a difference, it is still possible to use the modified AI Decomposition procedure to obtain a $(1-\epsilon)$-approximate densest cluster CIVD.

Let c be a type-2 cell generated by the AI decomposition and q be a query point in c. Consider the root-to-leaf path in the recursion tree of the Decomposition algorithm for c. Let $v_1, v_2, \ldots, v_m$ be the sequence of distance-nodes removed from L along this recursion path (sorted by the increasing order of the time that they are removed), and $x_1, x_2, \ldots, x_m$ be the closest distances to their corresponding box-node u at the time when they are removed. Let D_{max} be maximum value of D passing through the recursion path and v_{max}, and u_{max} be the corresponding box-node and distance-node when D achieves its maximum value.

Below we prove the claim that if $v_{max} = v_i$, then the union of $v_i, v_{i+1}, \ldots, v_m$ is almost the densest cluster for q for a properly chosen β. Since $v_1, v_2, \ldots, v_m$ are all distance-nodes recorded for the type-2 cell c, by Lemma 3, we know that they form a partition of P. For any $p \in v_j$, by Lemma 2 we have

$$(1-\beta)x_j \le \|q-p\| \le (1+\beta)x_j. \quad (1)$$

Let ψ be a mapping defined as follows: For any $p \in v_j$, $\psi(p)$ is on the ray that emits from q and passes through

p, and with $\|p - q\| = x_j$. Let C denote the union of $v_i, v_{i+1}, \ldots, v_m$. By Lemma 1, we know that to prove the claim, it is sufficient to show that $F(\psi(C), q)$ is almost as large as $F(C_m(P', q), q)$, where $P' = \psi(P)$. Clearly, P' can be partitioned into subsets $\psi(v_1), \psi(v_2), \ldots, \psi(v_m)$, with all points in each $\psi(v_i)$, for $i = 1, \ldots, m$, having the same distance x_i to q. From the above discussion, we know that if $x_1, x_2, \ldots, x_m$ is in decreasing order, $\psi(C)$ is exactly $C_m(P', q)$. The following lemma shows that $x_1, x_2, \ldots, x_m$ are actually in roughly sorted order, which is sufficient for us to obtain an approximate densest cluster.

Lemma 7: In the modified AI-Decomposition algorithm with error tolerance β, $x_j \leq (1+\beta)x_i$ for any $1 \leq i < j \leq m$.

Lemma 8: $F(\psi(C), q) \geq (1+\beta)^{-d} F(C_m(P', q), q)$.

Theorem 4: For any β satisfying the conditions $1 - (1+\beta)^{-d} \leq \Delta^{-1}(\epsilon)$ and $\beta \leq \Delta^{-1}(\epsilon)/3$, the modified AI decomposition algorithm outputs a $(1-\epsilon)$ approximate density-based CIVD in $O(n \log^2 n)$ time.

VI. Vector CIVD

In this section, we show that the AI decomposition can be combined with an assignment algorithm to build a $(1-\epsilon)$-approximate CIVD for the vector CIVD problem.

Let P be a set of n points in $\mathbb{R}^d$ and F be the influence function. For each point $p \in P$ and a query point q in $\mathbb{R}^d$, the influence $F(\{p\}, q)$ is a vector in the direction of $p - q$ (or $q - p$) and with a magnitude of $\|p - q\|^{-t}$ for some constant $t \geq 1$. Such a function may represent force-like influence between objects, such as the gravity force between planets and stars (with $t = d - 1$) or electric force between physical bodies like electrons and protons (with $t = 2$). For a cluster site C of P, the influence from C to a query point q is the vector sum of the individual influences from each point of C to q. The vector CIVD problem is to partition the space into Voronoi cells such that each cell is the union of points sharing the same maximum influence site.

Theorem 5: The vector CIVD problem satisfies the three properties in Section III for any constant $t \geq 1$.

The above theorem suggests that AI decomposition can be used to build an approximate Vector CIVD. To complete the construction, we still need an assignment algorithm to assign a cluster site to each type-2 cell. For this purpose, we choose β to be $\Delta^{-1}(\epsilon)$. By Theorem 1, we know that to determine an approximate maximum influence site for a type-2 cell c, it is sufficient to pick an arbitrary point $q \in c$ and find the cluster site which gives q the maximum influence.

The following observation is the key to determining the approximate maximum influence site.

Observation 1: In the vector CIVD problem, if a subset C is the maximum influence site of a query point q, there exists a hyperplane H passing through q and with all points of C locating at one side of H and all points in $P \setminus C$ at the other side of H.

The above observation suggests that $C_m(P, q)$ can be obtained by enumerating all possible partitions of P induced by a hyperplane passing through q and finding the *Optimal Hyperplane Partition (OHP)*. Since there are n input points, a total of $O(n^d)$ such hyperplanes need to be considered. Thus straightforwardly solving the OHP problem could be too costly. To improve the running time, our idea is to significantly reduce the number of input points involved in the OHP problem. The main strategy for reducing the number of points is to perturb the aggregated input points so that each aggregated point cluster is mapped to a single point. Also, those points that are far away from q and have little influence on q are ignored. In this way, we can reduce the number of input points from n to $O(\log n)$.

A quad-tree decomposition based *aggregation-tree* T is built to help identify those point clusters that can be perturbed. The to-be-perturbed point clusters form an *effective cover* (*i.e.,* a set of disjoint point clusters not too far away from q) in the aggregation-tree T. Straightforwardly computing the effective cover takes $O(n)$ time. To improve the time complexity, a key problem is to avoid searching a long-path (with a possible length of $O(n)$) in T. We use a number of techniques, such as the majority path decomposition, to build some auxiliary data structures for T so that we can perform binary search on this long path and therefore speed up the computation from $O(n)$ time to $O(\log^2 n)$. With this and a fact that the effective cover has a size of $O(\log n)$, we obtain an assignment algorithm which assigns a $(1-\epsilon)$-approximate maximum influence site to any type-2 cell in $O(\log^d n)$ time.

Theorem 6: A $(1-\epsilon)$-approximate vector CIVD can be constructed in $O(n \log^{d+1} n)$ time, where n is the number of points in P and $d \geq 2$ is the dimensionality of the space.

Acknowledgment: The research of the first author was supported in part by NSF under Grant CCF-1217906, and the research of the last three authors was supported in part by NSF under grant IIS-1115220.

References

[1] R. Andersen, D.F. Gleich, and V. Mirrokni, "Overlapping Clusters for Distributed Computation," *Proc. 5th ACM International Conference in Web Search and Data Mining*, 2012, pp. 273-282.

[2] S. Arya and T. Malamatos, "Linear-Size Approximate Voronoi Diagrams," *Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms (SODA'02)*, pp. 147155, 2002.

[3] S. Arya, T. Malamatos, and D. M. Mount, "Space-Efficient Approximate Voronoi Diagrams," *Proc. 34th ACM Symp. on Theory of Computing (STOC 2002)*, pp. 721730, 2002.

[4] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching," *Journal of the ACM,* 45 (1998), pp. 891923.

[5] F. Aurenhammer, "Power Diagrams: Properties, Algorithms and Applications," *SIAM J. on Computing*, 16(1) (1987), 78-96.

[6] F. Aurenhammer, "Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, 23 (1991), 345-405.

[7] A. Banerjee, C. Krumpelman, S. Basu, R.J. Mooney, and J. Ghosh, "Model-based Overlapping Clustering," *Proc. 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005, pp. 532–537.

[8] G. Barequet, M.T. Dickerson, and R.L.S. Drysdale III, "2-Point Site Voronoi Diagrams," *Discrete Applied Mathematics*, 122(1-3) (2002), 37-54.

[9] G. Barequet, M.T. Dickerson, D. Eppstein, D. Hodorkovsky, and K. Vyatkina, "On 2-Site Voronoi Diagrams under Geometric Distance Functions," *Proc. 8th International Symp. on Voronoi Diagrams in Science and Engineering*, 2011, pp. 31-38.

[10] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping Correlation Clustering," *Proc. IEEE 11th International Conference on Data Mining*, 2011, pp. 51-60.

[11] P. Callahan and R. Kosaraju, "A Decomposition of Multidimensional Point Sets with Applications to k-nearest-neighbors and n-body Potential Fields," *JACM*, 42(1) (1995), 67-90.

[12] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based Clustering over an Evolving Data Stream with Noise," *Proceedings of the 6th SIAM International Conference on Data Mining*, 2006, pp. 328-339.

[13] D.Z. Chen, M.H.M. Smid, and Bin Xu, "Geometric Algorithms for Density-based Data Clustering," *Int. J. Comput. Geometry Appl.*, 15(3) (2005), 239-260.

[14] N. Chen, J. Zhu, F. Sun, and E.P. Xing, "Large-margin Predictive Latent Subspace Learning for Multi-view Data Analysis," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 34(12) (2012), 2365-2378.

[15] Y. Chen and L. Tu, "Density-based Clustering for Real-time Stream Data," *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 133-142.

[16] C.M. Christoudias, R. Urtasun, and T. Darrell, "Multi-view Learning in the Presence of View Disagreement," arXiv:1206.3242, June 2012.

[17] G. Cleuziou, L. Martin, and C. Vrain, "Poboc: An Overlapping Clustering Algorithm," *Proc. 16th European Conference on Artificial Intelligence*, 2004, pp. 440-444.

[18] M.T. Dickerson and D. Eppstein, "Animating a Continuous Family of Two-site Voronoi Diagrams (and a Proof of a Bound on the Number of Regions)," *Proc. 25th ACM Symp. Computational Geometry*, 2009, pp. 92-93.

[19] M.T. Dickerson and M.T. Goodrich, "Two-site Voronoi Diagrams in Geographic Networks", *Proc. 16th ACM SIGSPATIAL International Conf. Advances in Geographic Information Systems*, 2008, doi:10.1145/1463434.1463504.

[20] H. Ding and J. Xu, "Solving Chromatic Cone Clustering via Minimum Spanning Sphere," *Proc. 38th International Colloquium on Automata, Languages and Programming (ICALP)*, 2011, pp. 773-784.

[21] D. Greene and P. Cunningham, "Multi-view Clustering for Mining Heterogeneous Social Network Data," *Proc. 31st European Conference on Information Retrieval, Workshop on Information Retrieval over Social Networks*, LNCS, Vol. 5478, 2009.

[22] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, Cambridge (1988).

[23] L. Greengard. "The Numerical Solution of the N-body Problem," *Computers in Physics,* 4, pp. 142152 (1990).

[24] L. Greengard. "Fast Algorithms for Classical Physics." *Science* 265, 909 914 (1994).

[25] I. Hanniel and G. Barequet, "On the Triangle-Perimeter Two-site Voronoi Diagram," *Trans. on Computational Science*, 9 (2010), 54-75.

[26] S. Har-Peled, "A Replacement for Voronoi Diagrams of Near Linear Size," *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 2001)*, pp. 94103, 2001.

[27] Sariel Har-Peled and Nirman Kumar, "Down the Rabbit Hole: Robust Proximity Search and Density Estimation in Sublinear Space." *FOCS 2012*: 430-439.

[28] D. Hodorkovsky, "2-Point Site Voronoi Diagrams," M.Sc. Thesis, Technion, Haifa, Israel, 2005.

[29] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *Proc. 30th ACM Symp. on Theory of Computing (STOC 1998)*, pp. 604 613, 1998.

[30] H.-P. Kriegel and M. Pfeifle, "Density-based Clustering of Uncertain Data," *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 672-677.

[31] D.T. Lee and R.L.S. Drysdale, III, "Generalization of Voronoi Diagrams in the Plane," *SIAM J. Comput.*, 10(1) (1981), 73-87.

[32] A.Y. Liu and D.N. Lam, "Using Consensus Clustering for Multi-view Anomaly Detection," *Proc. IEEE CS Security and Privacy Workshops*, 2012, pp. 117-125.

[33] A. Okabe, B. Boots, K. Sugihara, and S.N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd Eds., John Wiley & Sons, 2000.

[34] E. Papadopoulou, "The Hausdorff Voronoi Diagram of Point Clusters in the Plane," *Algorithmica*, 40 (2004), 63-82.

[35] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Mining and Knowledge Discovery*, 2(2) (1998), 169-194.

[36] K. Vyatkina and G. Barequet, "On 2-Site Voronoi Diagrams under Arithmetic Combinations of Point-to-Point Distances," *Proc. 7th International Symp. Voronoi Diagrams in Science and Engineering*, 2010, pp. 33-41.