A Clustering-Based Approach to Kinetic Closest Pair

Timothy M. Chan[1] **and Zahed Rahmati**[2]

1 **Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada**
`tmchan@uwaterloo.ca`

2 **School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran**
`rahmati@ece.ut.ac.ir`

Abstract

Given a set P of n moving points in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant, we present a kinetic data structure (KDS) for maintenance of the closest pair on P. Assuming the closest pair distance is between 1 and Δ over time, our KDS uses $O(n \log \Delta)$ space and processes $O(n^2 \beta \log \Delta \log n + n^2 \beta \log \Delta \log \log \Delta))$ events, each in worst-case time $O(\log^2 n + \log^2 \log \Delta)$. Here, β is an extremely slow-growing function. The locality of the KDS is $O(\log n + \log \log \Delta)$. Our closest pair KDS supports insertions and deletions of points. An insertion or deletion takes *worst-case* time $O(\log \Delta \log^2 n + \log \Delta \log^2 \log \Delta)$.

Also, we use a similar approach to provide a KDS for the all ε-nearest neighbors in $\mathbb{R}^d$.

The complexities of the previous KDSs, for both closest pair and all ε-nearest neighbors, have polylogarithmic factor, where the number of `logs` depends on dimension d. Assuming Δ is polynomial in n, our KDSs obtain improvements on the previous KDSs.

Our solutions are based on a kinetic clustering on P. Though we use ideas from the previous clustering KDS by Hershberger, we simplify and improve his work.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Kinetic Data Structure, Clustering, Closest Pair, All Nearest Neighbors

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.28

1 Introduction

Let P be a set of of points in $\mathbb{R}^d$. The *closest pair* problem is a fundamental, well-studied proximity problem in computational geometry, which is to find a pair of points in P with minimum separation distance. A decision version of the closest pair problem, called the *closest pair decision* problem, is to decide whether the closest pair distance is less than or equal to a given r. In many applications, *e.g.*, collision detection, the closest pair decision problem is more important than the closest pair problem. A general version of the closest pair problem is finding the nearest neighbor in P for each point in P, which is called the *all nearest neighbors* problem. The *all ε-nearest neighbors* problem is to find a point $p \in P$ to each point $q \in P$ such that $d(p, q) \leq (1 + \varepsilon) \cdot d(p^*, q)$, where $p^* \in P$ is the nearest neighbor of q, and $d(.,.)$ denotes the Euclidean distance between two points; p is called an ε-nearest neighbor to q.

The *unit disk covering* problem is to find the minimum cardinality set $\mathcal{S}$ of unit disks such that each point in P is covered by some disk in $\mathcal{S}$. The problem is well-motivated from many applications, *e.g.*, VLSI design and facility location. The unit disk covering problem is NP-hard in the L_2 and L_∞ metrics [7]. There exist polynomial time approximation solutions, of constant factor, to the unit disk covering problem in the L_2 and L_∞ metrics [4, 6, 9, 11].

Consideration of the problems on a set of moving objects has been studied extensively in different communities (*e.g.*, computational geometry, robotics, and computer graphics); see [12] and references therein. In this paper, we focus on the kinetic problems for a set P of n moving points in a *fixed* dimension d. Next, we formally state the kinetic problems.

KDS framework

Basch, Guibas, and Hershberger [2] first introduced the *kinetic data structure* (KDS) framework to maintain an attribute (*e.g.*, closest pair) of a set P of moving points. In the KDS framework, it is assumed that the trajectory of each point in P is given by a polynomial of degree bounded by some constant $\bar{s}$. A set of data structures and algorithms, namely a *kinetic data structure* (KDS), is built to maintain the attribute of interest. A KDS includes a set of *certificates* (boolean functions) that attests the attribute of interest is valid over time, except at some *discrete moments* (failure times of the certificates); when a certificate fails we say an *event* occurs. To track the next event after the current time, we define a *priority queue* of the failure times of the certificates. Note that any change to the events in the priority queue requires $O(\log n)$ for the update, and also note that the response time to an event in the KDS does not include this update time. An important criterion in a KDS is the *locality* of the KDS, which is the number of certificates associated with a particular point at any fixed time. If the locality of a KDS is polylogarithmic in n (or maximum nearest neighbor distance), the KDS is called *local*. A local KDS ensures that when a point changes its trajectory only a small number of changes is needed in the KDS.

Statements of kinetic problems

The *kinetic closest pair* problem is to maintain the closest pair in P over time. The *kinetic closest pair decision problem* is defined as follows: Given a parameter r, build a KDS to determine at any time whether the closest pair distance is less than or equal to r. Maintaining the nearest neighbor in P to each point in P is called the *kinetic all nearest neighbors* problem. The *kinetic all ε-nearest neighbors* problem is to maintain some ε-nearest neighbor $p \in P$ to each point $q \in P$ such that $d(p, q) \leq (1 + \varepsilon) \cdot d(p^*, q)$, where $p^* \in P$ is the nearest neighbor of q.

The *kinetic clustering* problem is to build a KDS that maintains a set S of clusters on the moving points in P, such that each cluster can be covered by a (unit) disk, and such that the cardinality $|S|$ of S is within a small factor of $|\mathcal{S}|$, the minimum possible by the optimal covering $\mathcal{S}$.

Related work

Basch, Guibas, and Hershberger [2] gave the first KDS for the closest pair on a set of n moving points, where the trajectory of each point is a polynomial bounded by some constant $\bar{s}$. Let $s = 2\bar{s} + 2$. Their KDS uses $O(n)$ space and processes $O(n^2\beta_s(n)\log n)$ events, each in time $O(\log^2 n)$; their KDS is local. Here, $\beta_s(n)$ is an extremely slow-growing function, *i.e.*, $\beta_s(n) = \lambda_s(n)/n$, where $\lambda_s(n)$ is the maximum length of Davenport-Schinzel sequences of order s on n symbols. Their KDS was later simplified and extended to higher dimensions d, using multidimensional range trees, by Basch, Guibas, and Zhang [3]. The KDS of [3] uses $O(n \log^{d-1} n)$ space, processes $O(n^2\beta_s(n)\log n)$ events, each in time $O(\log^d n)$, and it is local.

Table 1 The previous (and new) kinetic results for the attribute closest pair (CP). The attribute CP(r) is to decide whether the closest pair distance is at most r.

attribute	dim.	space	#events	proc. time	local
CP [2]	2	$O(n)$	$O(n^2\beta_s(n)\log n)$	$O(\log^2 n)$ /event	Yes
CP [3]	d	$O(n\log^{d-1} n)$	$O(n^2\beta_s(n)\log n)$	$O(\log^d n)$ /event	Yes
CP [1]	d	$O(n\log^{d-1} n)$	$O(n^2\beta_s(n)\log n)$	$O(\log^d n)$ /event	Yes
CP [14]	2	$O(n)$	$O(n^2\beta_s^2(n)\log n)$	$O(n^2\beta_s^2(n)\log^2 n)$	No
CP [here]	d	$O(n\log\Delta)$	$O((n^2\beta_s(n\log\Delta)\log\Delta)\cdot(\log n+\log\log\Delta))$	$O(\log^2 n+\log^2\log\Delta)$ /event	Yes
CP(r) [here]	d	$O(n)$	$O(n^2)$	$O(1)$ /event	Yes

Agarwal *et al.* [1] used multidimensional range trees to provide KDSs for maintenance of the closest pair and all the nearest neighbors in $\mathbb{R}^d$. Their closest pair KDS, which has the same approach and complexity as that of [3], supports insertions and deletions of points, where each operation takes *amortized* time $O(\log^{d+1} n)$. For maintenance of all the nearest neighbors, they implemented multidimensional range trees by randomized search trees (treaps). Their all nearest neighbors KDS uses $O(n\log^d n)$ space and handles $O(n^2\beta_s(n)\log^{d+1} n)$ events, with total processing time $O(n^2\beta_s(n)\log^{d+2} n)$. Each insertion or deletion in this KDS takes *expected* time $O(n)$. Rahmati *et al.* [14] used the kinetic semi-Yao graph (*i.e.*, theta graph) as a supergraph of the nearest neighbor graph to present a simple method for maintenance of the closest pair and all nearest neighbors. Their kinetic approach, which in fact maintains two Delaunay triangulations in $\mathbb{R}^2$, uses linear space and processes $O(n^2\beta_s^2(n)\log n)$ events, with total cost $O(n^2\beta_s^2(n)\log^2 n)$. By taking advantage of multidimensional range trees, the approach of [14] was later extended to higher dimensions to maintain all the nearest neighbors and all the ε-nearest neighbors [13]. None of the KDSs for maintenance of all the *exact* nearest neighbors is local.

Tables 1 and 2 summarize the complexities of the previous KDSs for maintenance of the closest pair, all the nearest neighbors, and all the ε-nearest neighbors. Here, "dim.", "#events", and "proc." stand for "dimension", "number of events", and "processing", respectively. There is also a different track, instead of maintaining an attribute over time, one would be interested in finding a time value for which the attribute is minimized or maximized. For a set of linearly moving points, in fixed dimension d, Chan and Rahmati [5] provided an approach to approximate the *minimum closest pair distance* and *minimum nearest neighbor distances* over time. For any constant $\varepsilon > 0$, their approach computes a $(1+\varepsilon)$-factor approximation to the minimum closest pair distance in time $\tilde{O}(n^{5/3})$. The notation $\tilde{O}$ hides polylogarithmic factors. Assuming $n \le m \le n^5$, their approach builds a data structure, which uses $\tilde{O}(m)$ preprocessing time and space, for answering queries: For any linearly moving query point q, their structure computes in time $\tilde{O}(\frac{n}{m^{1/5}})$ a $(1+\varepsilon)$-factor approximation to the minimum nearest neighbor distance to q over time.

Gao *et al.* [8] presented a *randomized* algorithm to maintain a clustering of moving points in $\mathbb{R}^2$, where each cluster can be covered by a unit square such that the centers of the squares are located at the points of P. The number of squares in their approach is on the order of $10^6\cdot|\mathcal{S}|$. Their KDS uses $O(n\log n\log\log n)$ space, and processes $O(n^2\log\log n)$ events, each in expected time $O(\log^{3.6} n)$. The locality of their KDS is $O(\log\log n)$. They proved that the number of changes of the optimal covering is $\Theta(n^3)$, and any approximate covering with constant factor undergoes $\Omega(n^2)$ changes.

Hershberger [10] gave a *deterministic* solution to the kinetic clustering problem in fixed dimension d in the L_∞ metric, where the number of axis-aligned boxes is at most $3^d\cdot|\mathcal{S}|$.

Table 2 The previous (and new) kinetic results for maintenance of all nearest neighbors (NNs) and all ε-nearest neighbors (ε-NNs).

attribute	dim.	space	#events	proc. time	local
all NNs [1]	d	$O(n \log^d n)$	$O(n^2 \beta_s(n) \log^{d+1} n)$	$O(n^2 \beta_s(n) \log^{d+2} n)$	No
all NNs [14]	2	$O(n)$	$O(n^2 \beta_s^2(n) \log n)$	$O(n^2 \beta_s^2(n) \log^2 n)$	No
all NNs [13]	d	$O(n \log^d n)$	$O(n^2 \beta_s^2(n) \log n)$	$O(n^2 \beta_s(n) \log^{d+1} n)$	No
all ε-NNs [13]	d	$O(n \log^d n)$	$O(n^2 \log^d n)$	$O(\log^d n \log \log n)$ /event	Yes
all ε-NNs [here]	d	$O(n \log \Delta')$	$O((n^2 \beta_s(\log \Delta'))\cdot$ $(\log \Delta' \log \log \Delta'))$	$O(\log^2 \log \Delta')$ /event	Yes

His KDS uses linear space, and processes $O(n^2)$ events, each in $O(\log^2 n)$ time. The locality of the KDS is $O(\log n)$. His approach uses a dimensional reduction technique: It partitions the points into 1-dimensional clusters, covered by strips (of width at most one) perpendicular to x_1-axis, then partitions the points *in each of these clusters* into 2-dimensional clusters, covered by strips (of width at most one) perpendicular to x_2-axis, and so on. Each event at one level of this hierarchy creates $O(1)$ dynamic changes to the clusters at next level of the hierarchy. Handling an event in his approach requires dynamic maintenance, which in fact involves checking many *complicated* cases. For each strip at each level of the hierarchy, his approach uses two *dynamic and kinetic tournament trees* to track the leftmost point and rightmost point of the strip. He posed the problem of providing a smooth kinetic maintenance for clustering on P *without dimension reduction.*

Main contributions

For a set P of n moving points, in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant $\bar{s}$, we provide clustering-based solutions to the kinetic closest pair decision problem and kinetic closest pair problem. Our kinetic clustering approach in $\mathbb{R}^d$ uses the kinetic 1-dimensional clustering by Hershberger.

Given a parameter r, we present a KDS for deciding in time $O(1)$ whether the closest pair distance is less than or equal to r. This KDS uses $O(n)$ space and processes $O(n^2)$ events, each in $O(1)$ time. The KDS can support insertions and deletions of points, where each operation can be performed in worst-case time $O(\log n)$.

To solve the optimization problem of maintaining the closest pair, we assume the closest pair distances is between 1 and Δ. This assumption is related to the assumption that the ratio between the maximum closest pair distance and the minimum closest pair distance is bounded by some parameter Δ. In many applications, the maximum closest pair distance over time is small, which makes our assumption and results reasonable. However, we can use our kinetic solution for the closest pair decision problem to detect if the closest pair distance is less than 1 or greater than Δ.

Our KDS for maintenance of the closest pair in $\mathbb{R}^d$ uses $O(n \log \Delta)$ space and processes $O(n^2 \log \Delta \log n \beta_s(n \log \Delta) + n^2 \log \Delta \log \log \Delta \beta_s(n \log \Delta))$ events, each in time $O(\log^2 n + \log^2 \log \Delta)$. We can dynamize our closest pair KDS such that each insertion or deletion takes *worst-case* time $O(\log \Delta \log^2 n + \log \Delta \log^2 \log \Delta)$. Note that, the space and processing time of each event in both previous closest pair KDSs by Basch *et al.* [3] and Agarwal *et al.* [1] are $O(n \log^{d-1} n)$ and $O(\log^d n)$, respectively, and also each insertion or deletion in the closest pair KDS by Agarwal *et al.* takes *amortized* time $O(\log^{d+1} n)$, where the number of `logs` is dependent on dimension d, whereas in our KDS it does not grow with d.

In addition, assuming the nearest neighbor distance to each point is between 1 and Δ', we provide a KDS (similar to that of the closest pair) for maintenance of all the ε-nearest neighbors in $\mathbb{R}^d$. This KDS uses $O(n \log \Delta')$ space and handles $O(n^2 \log \Delta' \log\log \Delta' \beta_s(\log \Delta'))$ events, each in *worst-case* time $O(\log^2 \log \Delta')$. The locality of the KDS is $O(\log\log \Delta')$. Our KDS for all ε-nearest neighbors supports insertions and deletions of points, where each operation takes *worst-case* time $O(\log \Delta' \log n + \log \Delta' \log^2 \log \Delta')$. In the previous KDS for maintenance of all the exact nearest neighbors by Agarwal *et al.* [1], an insertion or deletion takes *expected* time $O(n)$.

Tables 1 and 2 give the comparisons between our results and the previous results.

Our approach is based on a clustering on moving points. We use the 1-dimensional clustering KDS by Hershberger to provide a d-dimensional clustering KDS. Our KDS uses $O(n)$ space and processes $O(n^2)$ events, each in $O(1)$ time. Each point participates in $O(1)$ certificates. At any time, each cluster can be covered by a d-dimensional axis-aligned box of maximum side-length one, and the number of boxes is $|S| \leq 3^d \cdot |\mathcal{S}|$. For the $\mathbb{R}^d$ case, our approach is simpler than the approach by Hershberger: We in fact do the future work stated in his paper; we solve the problem *without dimension reduction*, which is a need to his KDS. Our KDS uses d *kinetic sorted lists*, but his KDS uses (order of) $2 \cdot d \cdot 3^d \cdot |\mathcal{S}|$ dynamic and kinetic tournament trees. Also, we obtain improvements on his KDS: Processing an event in our KDS takes constant time, but the processing time in his KDS is $O(\log^2 n)$. The locality of our KDS is $O(1)$, but it is $O(\log n)$ in his KDS.

2 Kinetic Clustering

Section 2.1 provides a kinetic 1-dimensional clustering for a set P of moving points in $\mathbb{R}^d$, where the trajectory of each point is a polynomial of degree bounded by some constant. In Section 2.2, we give a simple generalization that allows us to maintain a d-dimensional clustering, where each cluster can be covered by a d-dimensional axis-aligned box of maximum side-length one.

2.1 The 1-d Case

Hershberger [10] provided a kinetic approach for clustering a set P of moving points by strips, perpendicular to x-axis, of width at most one. We call an *x-cluster* the set of points in P covered by some strip B. Denote by $lpt(C)/rpt(C)$ the leftmost/rightmost point of the x-cluster C. The *diameter* of an x-cluster C is $x(rpt(C)) - x(lpt(C))$. Let $lb(C)/rb(C)$ denote the x-coordinate of the left/right boundary of B, the strip corresponding to C. Let C_ℓ/C_r denote the next x-cluster on the left/right side of C.

Hershberger's kinetic approach uses three types of x-clusters (*right set*, *left set*, and *gap set*) with the following properties to obtain a smooth kinetic maintenance of x-clusters.

$$(lb(C), rb(C)) = \begin{cases} (x(lpt(C)), x(lpt(C)) + 1) & \text{when } C \text{ is a right set,} \\ (lb(C_r) - 1, lb(C_r)) & \text{when } C \text{ is a left set,} \\ (x(lpt(C)), x(rpt(C))) & \text{when } C \text{ is a gap set.} \end{cases}$$

His approach maintains the following invariants over time, where each of them can be considered as a KDS certificate, called an *invariant certificate*. An invariant certificate fails when the distance between two points is zero, one, or two.

(I1) If $p \in C$, then either $lb(C) \leq x(p) < rb(C)$, or $x(p) = rb(C)$ and C is a gap set.
(I2) For all C, $rb(C) \leq lb(C_r)$.
(I3) If C is a gap (resp. left) set, then C_r is not a gap (resp. left) set.
(I4) If C is a gap set, then $lb(C_r) - lb(C) < 1$.
(I5) If C is a gap set, and C_r and C_ℓ are right sets, then $lb(C_r) - rb(C_\ell) < 1$.

▶ **Lemma 1** ([10]). *Each point in P participates in $O(1)$ invariant certificates. When an invariant fails, the corresponding certificates can be updated in $O(1)$ time, by a constant number of x-cluster type changes, point transfers between the x-clusters, and singleton x-cluster creations. The number of x-clusters, at any time, is within a factor of 3 of the minimum possible by the optimal covering.*

Let L be a *kinetic sorted list* of the points in P, in increasing order according to their x-coordinates. For any two consecutive points in L, an *ordering certificate* is defined that attests the order of the two points along the x-axis; an ordering event occurs when two consecutive points in L exchange their order. We use the kinetic sorted list L to maintain $lpt(C)$ and $rpt(C)$ of all the x-clusters C.[1]

▶ **Lemma 2.** *Each point participates in two ordering certificates. When an ordering event occurs, the corresponding certificates can be updated in $O(1)$ time.*

Note that, for some x-cluster C, an update to $lpt(C)/rpt(C)$ implies $O(1)$ updates to the invariant certificates. Also note that updating some invariant certificate may create $O(1)$ changes to $lpt(\cdot)/rpt(\cdot)$ of the x-clusters. From this, together with Lemmas 1 and 2, we conclude:

▶ **Lemma 3.** *There exists a KDS that maintains a set S of x-clusters, such that each x-cluster can be covered by a strip of width at most one, where $|S| \leq 3 \cdot |\mathcal{S}|$. The KDS uses $O(n)$ space and handles $O(n^2)$ events, each in $O(1)$ time. The locality of the KDS is $O(1)$.*

2.2 The General Case: Any Fixed d

Denote the d coordinate axes by x_j, $j = 1, \ldots, d$. Hershberger's approach uses a dimension reduction approach: It first creates the x_1-clusters, then for each x_1-cluster it creates the x_2-clusters, and so on. This approach needs to extend the smooth maintenance of Lemma 1 to support insertions and deletions. The dynamic maintenance of his approach considers many complicated cases to update the clusters; each event at one level of the hierarchy creates dynamic changes to the clusters at next level of the hierarchy. Here, we show how simply we can maintain a set S of d-dimensional clusters on a point set P, without the dimension reduction and without the need of dynamic maintenance used by Hershberger.

Notation

Denote by $i_1, \ldots, i_d$ the indices that we use to refer to the strips perpendicular to the $x_1, \ldots, x_d$-axes, respectively. Let $B(i_j)$ denote some strip perpendicular to the x_j-axis, and let $C(i_j) = P \cap B(i_j)$ be the corresponding x_j-cluster for $B(i_j)$. Denote by $C(i_j + k)$ (resp. $C(i_j - k)$) the kth x_j-cluster right after (resp. before) the x_j-cluster $C(i_j)$. Let $B(i_1, \ldots, i_d)$ denote the d-dimensional axis-aligned box which is formed by the intersection of the strips $B(i_1), \ldots, B(i_d)$; let $C(i_1, \ldots, i_d) = P \cap B(i_1, \ldots, i_d)$. We denote by S the set of *non-empty* clusters $C(i_1, \ldots, i_d)$, for all $i_1, \ldots, i_d$. Figure 1 shows the strips of x_1-clusters and x_2-clusters of a set of points in $\mathbb{R}^2$; the nine non-empty boxes give a covering of the point set.

By application of Lemma 3, corresponding to each x_j-axis, we maintain a set of x_j-clusters. When an event associated with some x_j-cluster occurs, $O(1)$ points transfer between

[1] The KDS of Hershberger uses two *kinetic tournament trees* to maintain $lpt(C)$ and $rpt(C)$ for each cluster C. Thus his KDS includes a set of tournament certificates, where each point participates in $O(\log n)$ such certificates.

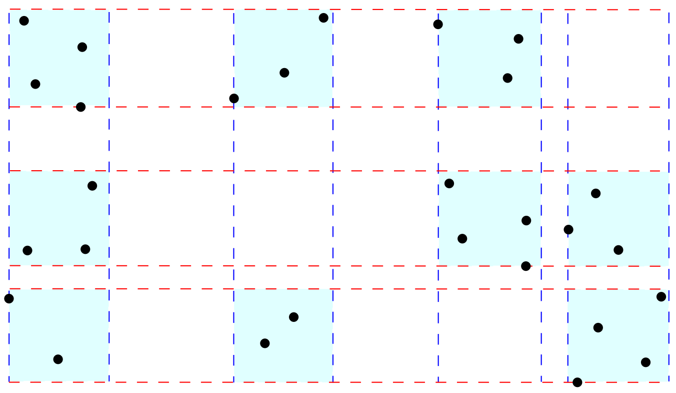

Figure 1 A set of 2-dimensional clusters of a point set in $\mathbb{R}^2$.

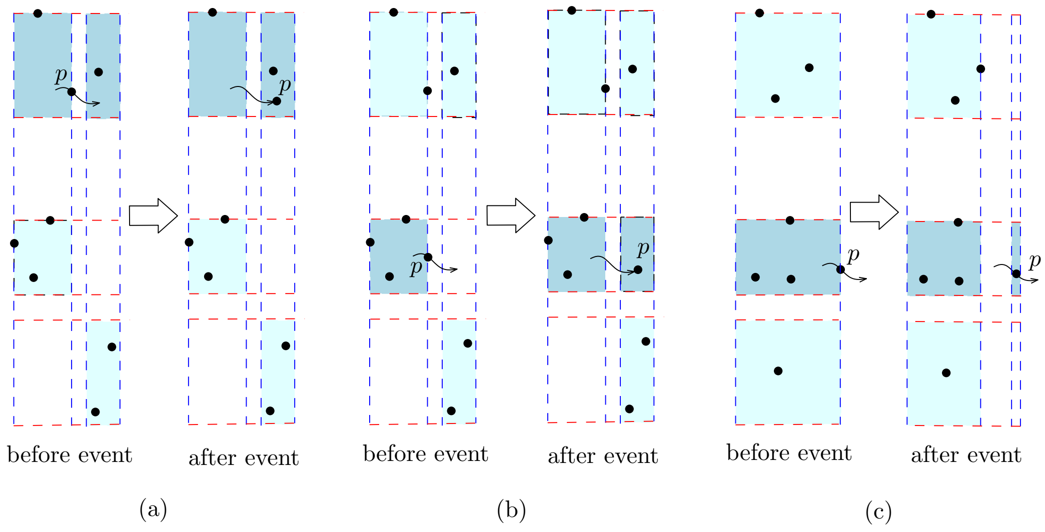

Figure 2 Updating the 2-dimensional clusters. (a) and (b) When p moves to an existing x_1-cluster. (c) When p moves to a new x_1-cluster.

the x_j-clusters, and $O(1)$ singleton x_j-clusters are created (from Lemma 1). Fix some $j \in \{1, \ldots, d\}$. Assume p is in some x_j-cluster $C(i_j)$, before an event. We update the set S of the d-dimensional clusters as follows.

- If p moves to an *existing* x_j-cluster, after the event: We delete p from the previous d-dimensional cluster and add to an existing/new d-dimensional cluster. For example, in Figures 2(a), we add p to an *existing* 2-dimensional cluster; in Figure 2(b), we create a *new* singleton 2-dimensional cluster for p.
- If p moves to a *new* x_j-cluster, after the event: We delete p from the previous d-dimensional cluster and add to a new singleton d-dimensional cluster; see Figure 2(c).

Consider $B(i_1, \ldots, i_j, \ldots, i_d)$, the axis-aligned box of $C(i_1, \ldots, i_j, \ldots, i_d)$, which contains p. (For simplicity, we use the notation C and B instead of $C(i_1, \ldots, i_j, \ldots, i_d)$ and $B(i_1, \ldots, i_j, \ldots, i_d)$, respectively.) The left/right boundary of B along the x_j-axis follows the left/right boundary of the strip $B(i_j)$ of $C(i_j)$, where $p \in C(i_j)$; *i.e.*, $lb(C) = lb(C(i_j))$ and $rb(C) = rb(C(i_j))$. Note that the left and right boundaries of B along other x_ℓ-axes

($\ell \neq j$) are the same as those of the box to which p belonged before the event. Also note that we delete a cluster C when its cardinality becomes equal to zero.

From the above discussion, there are $O(1)$ creations of new d-dimensional clusters of constant size in S, and $O(1)$ point insertions into or deletions from the clusters of S. Therefore, together with Lemma 3, we obtain:

▶ **Theorem 4.** *For a set P of moving points in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant, there exists a KDS that maintains a set S of d-dimensional clusters, such that each cluster can be covered by an axis-aligned box of maximum side-length one, where $|S| \leq 3^d \cdot |\mathcal{S}|$. The KDS uses $O(n)$ space and handles $O(n^2)$ events, each in $O(1)$ time* (plus $O(\log n)$ time to update the priority queue)*. The locality of the KDS is $O(1)$.*

3 KDS for Closest Pair

In Section 3.1, we first build a KDS to solve the *closest pair* decision problem. Then, in Section 3.2, we solve the optimization problem maintaining the closest pair over time. Finally, we dynamize the KDSs in Section 3.3.

3.1 Kinetic Closest Pair Decision Problem

Consider the following decision problem:

▶ **Decision Problem 1.** *Given a parameter r, determine at any time whether the closest pair distance is less than or equal to r.*

By application of Theorem 4, build a kinetic data structure $\mathcal{D}(r)$ for maintaining a set S of clusters on the moving points in P in $\mathbb{R}^d$, such that the maximum side-length of the axis-aligned boxes corresponding to the clusters is $r/\sqrt{d}$. Let $C'(i_j)$ be some x_j-cluster in the neighborhood of the x_j-cluster $C(i_j)$. We call $C'(i_1, \ldots, i_d)$ (or C' for short) a *neighbor* cluster to $C(i_1, \ldots, i_d)$ (or C for short) if $C'(i_j)$ is between $C(i_j - \lceil 2\sqrt{d} \rceil - 1)$ and $C(i_j + \lceil 2\sqrt{d} \rceil + 1)$ for all j, $1 \leq j \leq d$, *i.e.*,

▶ **Condition 1.** $C'(i_j) \in \{C(i_j - \lceil 2\sqrt{d} \rceil - 1), \ldots, C(i_j + \lceil 2\sqrt{d} \rceil + 1)\}$*, for all j ($1 \leq j \leq d$).*

If there exist two points of P in the same cluster $C \in S$, then the closest pair distance is less than or equal to r. Otherwise, for each singleton cluster $C = \{p\}$, we need to find the points q in the neighborhood, and check the possible candidate pairs (p, q) for the closest pair. In other words:

▶ **Lemma 5.** *The answer to Decision Problem 1 is yes iff the following disjunction is true:*

$$\left(\bigvee_{c} A_c\right) \vee \left(\bigvee_{c,c'} E_{c,c'}\right), \tag{1}$$

where

$$A_c = \begin{cases} \mathit{true} & \text{if } |C| \geq 2, \\ \mathit{false} & \text{if } |C| = 1, \end{cases} \qquad E_{c,c'} = \begin{cases} \mathit{true} & \text{if } d(p,q) \leq r, \text{ where } p \in C \text{ and } q \in C', \\ \mathit{false} & \text{if } d(p,q) > r, \text{ where } p \in C \text{ and } q \in C'. \end{cases}$$

Let κ be the number of true expressions among A_c and $E_{c,c'}$ in the disjunction of (1). We do the following updates during the changes to the clusters.

(U1) When a new cluster C is created, that in fact contains a single point, we define a new expression A_c with value false. Then we update the neighbors C'' of C' (neighbors of neighbors for C) as C might violate them, and define the corresponding edges (p, q) and expressions $E_{i',i''}$ between singleton clusters C' and C''. We set the value of $E_{i',i''}$ to true (resp. false) if $d(p, q) \leq r$ (resp. $d(p, q) > r$), where $C' = \{p\}$ and $C'' = \{q\}$.

(U2) When the cardinality of some $C \in S$ becomes equal to one, we find all the singleton clusters $C' \in S$ which satisfy Condition 1, and define the edges (p, q) and their corresponding expressions $E_{c,c'}$ with a valid value true/false, where $C_i = \{p\}$ and $C' = \{q\}$.

(U3) When the cardinality of some cluster C becomes bigger than one, the value of A_c becomes true, which implies that the disjunction of (1) is true.

We can easily track the value of κ over time: We increase (resp. decrease) κ by one if the cardinality of some $C \in S$ gets > 1 (resp. $= 1$). Also, we increase (resp. decrease) κ by one if $d(p, q) \leq r$ (resp. $d(p, q) > r$), where $C = \{p\}$ and $C' = \{q\}$; we can define a boolean function for each edge (p, q) attesting its length is less than or equal to r. Note that when $|C|$ gets bigger than one, since we do not need to track the values of $E_{c,c'}$ for all neighbors C', we delete all the expressions $E_{c,c'}$ and the edges (p, q), and decrease κ by the number of edges (p, q) such that $d(p, q) \leq r$. Now, we can conclude:

▶ **Theorem 6.** *Let r be a positive real parameter. For a set P of moving points in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant, there exists a KDS $\mathcal{D}(r)$ that decides in $O(1)$ time whether the closest pair distance is less than or equal to r. $\mathcal{D}(r)$ uses $O(n)$ space and handles $O(n^2)$ events, each in $O(1)$ time* (plus $O(\log n)$ time to update the priority queue)*. The locality of $\mathcal{D}(r)$ is $O(1)$.*

Proof. By the invariant certificates (I1)-(I5), for each x_j-axis, $rb(C(i_j + \lceil 2\sqrt{d}\rceil + 1)) - lb(C(i_j + 1)) > r$, which implies that (if exists) we can find a pair (p, q) in our KDS such that $d(p, q) \leq r$.

Condition 1 of the definition of a *neighbor* cluster insures that we check only a constant number of neighbor clusters. Thus the updates (U1)-(U3) can be done in time $O(1)$. At any time, deciding whether $\kappa > 1$ is equivalent to deciding whether the disjunction of (1) is true. From this together with Theorem 4, the proof obtains. ◀

3.2 Kinetic Closest Pair Problem

Assume the Euclidean distance between any two points in P at any time is at least 1 and at most Δ. Let $r_\ell = 2^\ell$, $0 \leq \ell \leq \log \Delta$.

Fix some $\ell \in \{0, \ldots, \log \Delta\}$. In a similar way to that of Section 3.1, we build a kinetic data structure $\mathcal{D}(r_\ell)$. Let E_ℓ denote the set of edges (p, q) between the clusters $C = \{p\}$ and the neighbor clusters $C' = \{q\}$ satisfying Condition 1. Let e_ℓ be the edge with minimum length in E_ℓ. At any time, the edge with minimum length among all e_ℓ, $\ell = 0, \ldots, \log \Delta$, gives the closest pair, which can be maintained over time using a *dynamic and kinetic tournament tree* $\mathcal{T}$ over the $O(n \log \Delta)$ edges in $\cup_\ell E_\ell$. Next, we summarize the main result of this section.

▶ **Theorem 7.** *For a set P of moving points in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant $\bar{s}$, our closest pair KDS uses $O(n \log \Delta)$ space and handles $O((n^2 \log \Delta \beta_s(n \log \Delta)) \cdot (\log n + \log\log \Delta))$ events, each in worst-case time $O(\log^2 n + \log^2 \log \Delta)$. Here, $s = 2\bar{s} + 2$. The total processing time of all events and the locality of the KDS are $O((n^2 \log \Delta \beta_s(n \log \Delta)) \cdot (\log^2 n + \log^2 \log \Delta))$ and $O(\log n + \log\log \Delta)$, respectively.*

Proof. By Theorem 3.1 of [1], for a sequence of m insertions/deletions into $\mathcal{T}$ whose maximum size at any time is $\tilde{n}$ ($m \geq \tilde{n}$), $\mathcal{T}$ handles $O(m\beta_{2\bar{s}+2}(\tilde{n})\log\tilde{n})$ events. The total processing time for handling all the events is $O(m\beta_{2\bar{s}+2}(\tilde{n})\log^2\tilde{n})$, each event can be handled in worst-case time $O(\log^2\tilde{n})$, and each point participates in $O(\log\tilde{n})$ tournament certificates.

From Theorem 6, and the fact that $\tilde{n} = |\cup_\ell E_\ell| = O(n\log\Delta)$ and the number of events is $m = O(n^2\log\Delta)$ for all the levels ℓ, $0 \leq \ell \leq \log\Delta$, the proof obtains. ◀

3.3 Dynamizing the KDSs

Here, we present that how our KDSs in Sections 3.1 and 3.2 support insertions and deletions of points.

Hershberger showed that the smooth kinetic maintenance of x_j-clusters (Lemma 1) can support insertions and deletions of points: When a point p is inserted into or deleted from P, the invariant certificates (I1)-(I5) can be updated by a constant number of x_j-cluster type changes and point transfers between the x_j-clusters. In Section 2.1, we use kinetic sorted lists L_j on the point set P, in increasing order along the x_j-axes, in order to track the $lpt(\cdot)/rpt(\cdot)$ of the x_j-clusters. We dynamize the kinetic sorted lists L_j to support point insertions and deletions; each operation can be handled in time $O(\log n)$. This implies that our KDS (of Section 2.2) for maintenance of a set S of d-dimensional clusters can easily support insertions and deletions of points.

Given the dynamic and kinetic clustering S in $\mathbb{R}^d$, we can perform the updates (U1)-(U3), after each cluster change, to decide whether the closest pair distance is less than or equal to r; each update can be done in time $O(1)$. This implies:

▶ **Lemma 8.** *Our KDS $\mathcal{D}(r)$ of Theorem 6* (for deciding, at any time, whether the closest pair distance in $\mathbb{R}^d$ is less than or equal to r) *supports insertions and deletions of points. Each operation can be performed in worst-case time $O(\log n)$.*

Assume that the Euclidean distance between the *inserted point* q and any other point $p \in P$, at any time, is at least 1 and at most Δ. When q is inserted into (resp. deleted from) P, we insert q into (resp. delete q from) the $\log\Delta + 1$ levels of our closest pair KDS of Section 3.2. Since we can dynamize each $\mathcal{D}(r_\ell)$, $0 \leq \ell \leq \log\Delta$ (by Lemma 8), and each insertion into or deletion from $\mathcal{T}$ can be done in $O(\log^2(n\log\Delta))$, we obtain the following.

▶ **Lemma 9.** *Our KDS of Theorem 7* (for maintenance of the closest pair in $\mathbb{R}^d$) *supports insertions and deletions of points. Each operation can be performed in worst-case time $O(\log\Delta\log^2 n + \log\Delta\log^2\log\Delta)$.*

4 KDS for All ε-Nearest Neighbors

Here, we first consider a decision version of the all ε-nearest neighbors problem, and then provide a KDS for maintenance of some ε-nearest neighbor to each point in P.

4.1 Kinetic All ε-Nearest Neighbors Decision Problem

Consider the following decision problem:

▶ **Decision Problem 2.** *Given parameters ε and r, (for each point $q \in P$) determine at any time whether there exists some point $p \in P$ such that its distance to q is less than or equal to $(1+\varepsilon)\cdot r$.*

In a similar way to that of Section 3.1, build a kinetic data structure $\mathcal{D}(\varepsilon r)$ for maintaining a set S of clusters, such that the maximum side-length of the boxes corresponding to the clusters is $\varepsilon r/\sqrt{d}$. For each cluster $C \in S$, we maintain some point z_c in C as the *representative point* of C. The distance between z_c and any other point in C is at most εr.

Recall that $C(i_j + k)$ (resp. $C(i_j - k)$) denote the kth x_j-cluster after (resp. before) the x_j-cluster $C(i_j)$, along the x_j-axis. Here, we define a new condition for a *neighbor* cluster $C'(i_1, \ldots, i_d)$ to $C(i_1, \ldots, i_d)$. We say C' is the neighbor cluster of C if:

▶ **Condition 2.** *$C'(i_j) \in \{C(i_j - \lceil 2\sqrt{d}/\varepsilon\rceil - 1), \ldots, C(i_j + \lceil 2\sqrt{d}/\varepsilon\rceil + 1)\}$, for all x_j-axes, $j = 1, \ldots, d$.*

Fix some point $q \in P$, and assume $q \in C$. Let $E(q)$ be the set of edges $(q, z_{c'})$, where $z_{c'}$ are the representative points of the neighbor clusters C' satisfying Condition 2.

▶ **Lemma 10.** *The answer to Decision Problem 2 (for each $q \in P$) is yes iff the following disjunction is true:*

$$D(q) = A_c(q) \vee \left(\bigvee_{c'} E_{c,c'}(q) \right), \tag{2}$$

where (assuming $q \in C$)

$$A_c(q) = \begin{cases} true & \text{if } |C| \geq 2, \\ false & \text{if } |C| = 1, \end{cases} \qquad E_{c,c'}(q) = \begin{cases} true & \text{if } d(q, z_{c'}) \leq (1+\varepsilon)\cdot r, \\ false & \text{if } d(q, z_{c'}) > (1+\varepsilon)\cdot r. \end{cases}$$

Let $\mathcal{T}(q)$ be a dynamic and kinetic tournament tree over the edges in $E(q)$, which maintains the edge $e(q)$ with minimum length in $E(q)$. From Lemma 10, if $|C| \geq 2$, then the value of $D(q)$ would be true; otherwise, the value of $D(q)$ is the answer to whether $\|e(q)\| \leq (1+\varepsilon)\cdot r$. For each $q \in P$, we define $\mathcal{T}(q)$, and maintain the values of $D(q)$. We do the following updates to $D(q)$, during the changes to the clusters in S.

(U1) When p is deleted from some C such that $z_c = p$, we select a point v in $C - \{p\}$ as the new representative point. If after the event $C = \{v\}$, we first build $\mathcal{T}(v)$ to determine the value of $D(v)$. Then we find all the singleton neighbor clusters $C' = \{q\}$, and in $\mathcal{T}(q)$ we replace the edge (q, p) with (q, v). Note that, if after the event $C = \emptyset$, we update the neighbors C'' of neighbors C', for C; for the singleton neighbor clusters $C' = \{q\}$, we update $\mathcal{T}(q)$ if the neighbors C'' of C' change.

(U2) When a point p is inserted into some C, we ignore $\mathcal{T}(p)$ and set the value of $D(p)$ to true. Note that, if before the event C is a singleton cluster (say $C = \{q\}$), we also delete $\mathcal{T}(q)$ and set the value of $D(q)$ to true.

(U3) When a new cluster C is created, that contains a single point (say $C = \{p\}$), we build $\mathcal{T}(p)$. We then update the neighbors C'' of neighbors C', for C, and also for the singleton neighbor clusters $C' = \{q\}$, we apply the required changes to $\mathcal{T}(q)$ if the neighbors of C'' of C' change.

▶ **Lemma 11.** *Let ε and r be positive real parameters, and let P be a set of moving points in fixed dimension d, where the trajectory of each point is a polynomial of degree bounded by some constant. There exists a KDS that decides, for any point $q \in P$ at any time, in time $O(1)$ whether there is a point $p \in P$ such that $d(p, q) \leq (1+\varepsilon)\cdot r$. The KDS uses $O(n)$ space and handles $O(n^2)$ events, each in $O(1)$ time* (plus $O(\log n)$ time to update the priority queue)*. The locality of the KDS is $O(1)$.*

Proof. From the invariant certificates (I1)-(I5), for the x_j-axis, $rb(C(i_j + \lceil 2\sqrt{d}/\varepsilon \rceil + 1)) - lb(C(i_j + 1)) > (1 + \varepsilon) \cdot r$. This implies that, for each q, we can find a point p in a neighbor cluster such that $d(p, q) \le (1 + \varepsilon) \cdot r$, if any such p exists.

Assuming ε and d are constants, Condition 2 implies that the number of neighbor clusters for each cluster in S is $O(1)$. Therefore, each of the updates ($\overline{\text{U}}1$)-($\overline{\text{U}}3$) can be done in time $O(1)$. The number of changes to the representative points, which is on the order of the number of changes to the clusters in S, is $O(n^2)$. This implies that the number of all events for all the constant size tournament trees $\mathcal{T}(q)$, for all $q \in P$, is $O(n^2)$. From this, together with the complexity of a $\mathcal{D}(\varepsilon r)$, the proof obtains. ◀

4.2 Kinetic All ε-Nearest Neighbors Problem

Assume the nearest neighbor distance to any point $q \in P$ is at least 1 and at most Δ'.

Let $r_\ell = 2^\ell$, $0 \le \ell \le \log \Delta'$. Fix some $\ell \in \{0, \ldots, \log \Delta'\}$. In a similar way to that of Section 4.1, we build $\mathcal{D}(\varepsilon r_\ell)$. Let $E_\ell(q)$ denote the set of edges $(q, z_{c'})$ between $C = \{q\}$ and its neighbor clusters C' satisfying Condition 2. We build a dynamic and kinetic tournament tree over the edges in $\cup_\ell E_\ell(q)$ to maintain the edge with minimum length in $\cup_\ell E_\ell(q)$, which in fact gives some ε-nearest neighbor to q.

▶ **Theorem 12.** *For a set P of moving points in fixed dimension d such that the trajectory of each point is a polynomial of degree bounded by some constant $\bar{s}$, our KDS for maintenance of all the ε-nearest neighbors uses $O(n \log \Delta')$ space and handles $O(n^2 \log \Delta' \beta_s(\log \Delta') \log\log \Delta')$ events, each in worst-case time $O(\log^2 \log \Delta')$. The total processing time for all the events and the locality of the KDS are $O(n^2 \log \Delta' \beta_s(\log \Delta') \log^2 \log \Delta')$ and $O(\log\log \Delta')$, respectively. Here, $s = 2\bar{s} + 2$.*

Proof. The proof is similar to the proof of Theorem 7. The dynamic and kinetic tournament tree corresponding to the point $q \in P$ handles $O(m_q \beta_{2\bar{s}+2}(\log \Delta') \log\log \Delta')$ events, each in worst-case time $O(\log^2 \log \Delta')$. Here, m_q is the number of insertions and deletions performed on the tournament tree of q. The total processing time of the events (associated with the tournament tree of q) and the locality are $O(m_q \beta_{2\bar{s}+2}(\log \Delta') \log^2 \log \Delta')$ and $O(\log\log \Delta')$, respectively. Since the number of insertions/deletions to all the tournament trees, for all the points in P, is $\sum_q m_q = O(n^2 \log \Delta')$, the proof obtains. ◀

▶ Remark. Our KDS of Theorem 12 *(for maintenance of all the ε-nearest neighbors in $\mathbb{R}^d$)* supports insertions and deletions of points. When a point q is inserted into (resp. deleted from) the point set P, we insert q into (resp. delete q from) the $\log \Delta' + 1$ levels of the kinetic data structures $\mathcal{D}(\varepsilon r_\ell)$. At each level, there exist $O(1)$ changes to the dynamic and kinetic sorted lists (where each one takes time $O(\log n)$; see Section 3.3), and $O(1)$ changes to the dynamic and kinetic tournament trees of the points (where each one takes time $O(\log^2 \log \Delta')$; see Theorem 12). Therefore, each operation can be performed in worst-case time $O(\log \Delta' \log n + \log \Delta' \log^2 \log \Delta')$.

▶ Remark. Our approach of Theorem 12 works to solve the bichromatic version of the problem. Given a set B of blue points and a set G of green points, for each green point $g \in G$, we want to maintain some blue point $b \in B$ as the bichromatic ε-nearest neighbor to g. Using a similar approach to that of Section 4.1, for each cluster C at each level ℓ ($0 \le \ell \le \log \Delta'$), we maintain a blue representative point. Then, for each green point g, where $g \in C$, we track some blue representative point in the neighbor clusters C' satisfying Condition 2.

References

1 Pankaj K. Agarwal, Haim Kaplan, and Micha Sharir. Kinetic and dynamic data structures for closest pair and all nearest neighbors. *ACM Transactions on Algorithms*, 5:4:1–37, 2008.

2 Julien Basch, Leonidas J. Guibas, and John Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31:1–19, 1999.

3 Julien Basch, Leonidas J. Guibas, and Li Zhang. Proximity problems on moving points. In *Proceedings of the 13th Annual Symposium on Computational Geometry (SoCG'97)*, pages 344–351, New York, NY, USA, 1997. ACM.

4 H. Brönnimann and M.T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(1):463–479, 1995.

5 Timothy M. Chan and Zahed Rahmati. Approximating the minimum closest pair distance and nearest neighbor distances of linearly moving points. *Computational Geometry*, 2016.

6 Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC'88)*, pages 434–444, New York, NY, USA, 1988. ACM.

7 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.

8 Jie Gao, Leonidas Guibas, John Hershberger, Li Zhang, and An Zhu. Discrete mobile centers. *Discrete & Computational Geometry*, 30(1):45–63, 2003.

9 Teofilo F. Gonzalez. Covering a set of points in multidimensional space. *Information Processing Letters*, 40(4):181–188, 1991.

10 John Hershberger. Smooth kinetic maintenance of clusters. *Computational Geometry*, 31(1–2):3–30, 2005.

11 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985.

12 Zahed Rahmati. *Simple, Faster Kinetic Data Structures*. PhD thesis, University of Victoria, 2014.

13 Zahed Rahmati, Mohammad Ali Abam, Valerie King, and Sue Whitesides. Kinetic k-semi-Yao graph and its applications. *Computational Geometry*, 2016.

14 Zahed Rahmati, Mohammad Ali Abam, Valerie King, Sue Whitesides, and Alireza Zarei. A simple, faster method for kinetic proximity problems. *Computational Geometry*, 48(4):342–359, 2015.