Kinetic Pie Delaunay Graph and Its Applications

Mohammad Ali Abam[1,2], Zahed Rahmati[3], and Alireza Zarei[4]

[1] Dept. of Computer Engineering, Sharif University of Technology, Tehran, Iran
abam@sharif.edu

[2] Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
abam@ipm.ir

[3] Dept. of Computer Science, University of Victoria, Victoria, BC, Canada
rahmati@uvic.ca

[4] Dept. of Mathematical Science, Sharif University of Technology, Tehran, Iran
zarei@sharif.edu

Abstract. We construct a new proximity graph, called the *Pie Delaunay graph*, on a set of n points which is a super graph of *Yao graph* and *Euclidean minimum spanning tree* (*EMST*). We efficiently maintain the *Pie Delaunay graph* where the points are moving in the plane. We use the kinetic *Pie Delaunay graph* to create a kinetic data structure (*KDS*) for maintenance of the *Yao graph* and the *EMST* on a set of n moving points in 2-dimensional space. Assuming x and y coordinates of the points are defined by algebraic functions of at most degree s, the structure uses $O(n)$ space, $O(n \log n)$ preprocessing time, and processes $O(n^2\lambda_{2s+2}(n)\beta_{s+2}(n))$ events for the *Yao graph* and $O(n^2\lambda_{2s+2}(n))$ events for the *EMST*, each in $O(\log^2 n)$ time. Here, $\lambda_s(n) = n\beta_s(n)$ is the maximum length of Davenport-Schinzel sequences of order s on n symbols. Our *KDS* processes nearly cubic events for the *EMST* which improves the previous bound $O(n^4)$ by Rahmati *et al.* [1].

Keywords: Euclidean minimum spanning tree, Yao graph, Pie Delaunay triangulation, kinetic data structures.

1 Introduction

Investigating geometric problems on moving points, known as kinetic geometric problems, has been studied extensively in the past decade [2, 3, 4, 5]. In this setting, the points are moving in the plane and our goal is to show a data structure maintaining the combinatorial structure of a special attribute during the motion. We assume that the trajectory of the point p_i at time t ($p_i(t)$) is defined by two polynomial functions of maximum degree s for x and y coordinates of p_i ($p_i(t) = (x_i(t), y_i(t))$).

In this paper, we present a simple kinetic data structure for maintenance of the following proximity problems. In Euclidean space, for a set $P = \{p_1, p_2, \ldots, p_n\}$ of n points, there exists the complete graph $G(V, E)$ where $V = P$ and E is the set of edges in the graph such that the weight of each edge is the Euclidean distance between its two endpoints. An *Euclidean minimum spanning tree* (*EMST*) of G is a connected sub-graph of G where the sum of the weights of its edges is the minimum possible. The *Yao graph* [6] of P can be constructed by partitioning the plane, for each point, into k wedges with equal angles $2\pi/k$ and connecting each point to the closest point in each of its k wedges. In the rest of the paper when we talk about the *Yao graph* it means that $k = 6$.

We present a *KDS* for the *Yao graph* and a new *KDS* for the *EMST* which is an improvement of the previous *EMST KDS* by Rahmati *et al.* [1] (our *KDS* processes nearly cubic events but the *KDS* in [1] processes $O(n^4)$ events). Guibas *et al.* [7] presented a *KDS* for Delaunay triangulation based on a *circle*, and Abam *et al.* [2] presented a *KDS* based on a *diamond.* There we partition a disk into six wedges with equal angles which creates six convex shapes; a Delaunay triangulation is then constructed based on each of these wedges. The union of all these triangulations is a sparse proximity graph. This new proximity graph, which we call *Pie Delaunay graph*, is a super graph of the *Yao graph* and the *EMST.*

Notation. $\lambda_s(n)$ is the maximum length of Davenport-Schinzel sequences of order s on n symbols. Intuitively, if we have a set of n moving points where the trajectory of each point is an algebraic function with at most degree s then the number of changes for the lowest point along the y-axis is $\lambda_s(n)$. Here, $\beta_s(n) = \frac{\lambda_s(n)}{n}$ and $\alpha(n)$ is the Inverse Ackermann function.

Related work. Fu and Lee (1991) [8] proposed the first algorithm for maintenance the *EMST* on a set of moving points. The algorithm uses $O(sn^4 \log n)$ preprocessing time where s is the maximum degree of the algebraic functions defining the trajectory of the points and uses $O(m)$ space where m is the maximum number of the changes of the *EMST* from time $t = 0$ to $t = \infty$. At any given time, the algorithm constructs the *EMST* in linear time. Agarwal *et al.*(1998) [9] proposed an algorithm for a restricted kinetic version of the *EMST* over a graph where the distance between each pair of points in the graph is defined by linear function of time. Processing time of the algorithm for each combinatorial change of the *EMST* is $O(n^{\frac{1}{2}} \log^{\frac{3}{2}} n)$.

The kinetic data structure (*KDS*) framework was introduced by Basch (1999) [10]. To maintain a special attribute of a set of moving points, a *KDS* defines a set of *certificates* which certify the correctness of the attribute. During the time when a certificate fails, one must update the value of the attribute and then, build the new set of certificates to satisfy the correctness of the attribute. Therefore, it suffices to compute the failure times of these certificates, *events*, and put them in an event queue. Whenever the time of the next event in the queue is equal to the current time, one invokes a repair mechanism to update the value of the attribute and replace the failed certificate(s) with new valid one(s). The set of data structures and the update mechanism used to update these certificates and maintain the attributes is called a *KDS.* The kinetic data structure framework has been used for solving many of the geometric problems in kinetic environments.

Basch *et al.* (1999)[11] presented an approximation algorithm for $(1+\epsilon)$-*EMST*. Their *KDS* uses $O(\epsilon^{\frac{-(d-1)}{2}} n \log^{d-1} n)$ space, $O(\epsilon^{\frac{-(d-1)}{2}} n \log^{d-1} n)$ preprocessing time, and processes $O(\epsilon^{-(d-1)} n^3)$ events, each in $O(\log^d n)$ time where d is the dimension of the points. Recently, Rahmati *et al.* (2011)[1] improved the previous result by Fu and Lee [8]. They presented the exact *KDS* for maintenance of the *EMST* on a set of n moving points in 2-dimensional space. They build a *KDS* of space $O(n)$ in $O(n \log n)$ preprocessing time and their *KDS* processes $O(n^4)$ events, each in $O(\log^2 n)$ time.

Our Results. We introduce the *Pie Delaunay graph* which is a super graph of *Yao graph.* Since the set of *EMST* edges is a subset of the set of *Yao graph* edges the *Pie Delaunay graph* includes the *EMST.* We maintain the *Pie Delaunay graph* which enables us to maintain *Yao graph* and *EMST.* Our *KDS* uses $O(n)$ space, $O(n \log n)$ preprocessing time, and processes $O(n^2 \lambda_s(n) \beta_{s+2}(n))$ events for the *Yao graph* and $O(n^2 \lambda_{2s+2}(n))$ events for the *EMST*, each in $O(\log^2 n)$ time.

We describe our *KDS* in two sections. Section 2 contains the construction of the *Pie Delaunay graph* and its *KDS*; the *KDS* in this section maintains the *Pie Delaunay graph* during the motion. Next, we show the application of the *Pie Delaunay graph* in Section 3: first we construct the *Yao graph* and the *EMST* and then in Subsection 3.2, we maintain the *Yao graph* based on the kinetic *Pie Delaunay graph* and in Subsection 3.3, we present the kinetic *EMST* based on the kinetic *Yao graph.*

2 Pie Delaunay Graph

In this section, we summarize the construction of the *Pie Delaunay graph* and then we present a *KDS* for it.

2.1 The Construction of the Pie Delaunay Graph

Partition a disk into six wedges $\sigma_0, ..., \sigma_5$, each of angle $\pi/3$ and the origin as their common apex, where σ_i spans the orientation $[i\pi/3, (i+1)\pi/3]$, and call any translated and scaled copy of σ_i an *i-pie*—see Fig. 1. Let P be a set of points in the plane. We denote the constructed Delaunay triangulation of the point set

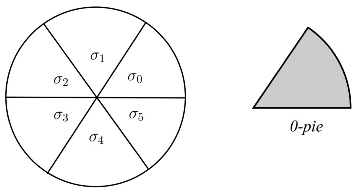

Fig. 1. Partitioning a circle into six pies and the 0-pie

P based on σ_i by $\mathcal{DT}_i(P)$ and define it as follow: For two points $p, q \in P$, the edge pq is an edge in $\mathcal{DT}_i(P)$ if and only if there is an i-pie where p and q are on its boundary and it does not contain any other points from P. Fig. 2 shows $\mathcal{DT}_0(P)$. In this figure, pq, qr, rp and s_1s_2 are the edges of $\mathcal{DT}_0(P)$ because for each of these edges there is a *0-pie* which dose not contain any other point from P. The *Pie Delaunay graph* ($\mathcal{DG}(P)$) is the union of all $\mathcal{DT}_i(P)$ for $i = 0, ..., 5$. We denote the set of $\mathcal{DG}(P)$ edges by $\mathcal{E}(\mathcal{DG}(P))$. Then, pq is an edge of $\mathcal{DG}(P)$ if and only if it is an edge in $\mathcal{DT}_i(P)$ where $0 \leq i \leq 5$. Each wedge σ_i is a convex shape and so, using the approach of [12], its corresponding Delaunay triangulation $\mathcal{DT}_i(P)$ can be constructed (based on σ_i) in $O(n \log n)$ time. Therefore, we can construct the *Pie Delaunay graph* in $O(n \log n)$ time.

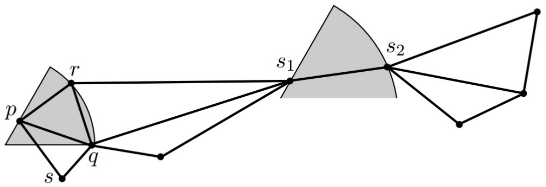

Fig. 2. Delaunay Triangulation based on 0-pie

2.2 Kinetic Pie Delaunay Graph

Given $\mathcal{E}(\mathcal{DG}(P))$, we show how to maintain the *Pie Delaunay graph* with processing time $O(\log n)$ for each certificate failure.

In our *KDS* we define two certificates *NotInPie* and *NotInCone*. Call the edges on the boundary of $\mathcal{DT}_i(P)$ *i-hull* edges. Every interior edge pq, which is not an i-hull edge, is incident to two triangles. Call the two triangles a quadrilateral, and let r and s be the two other vertices of the quadrilateral. For the convex shape i-pie that passes through p, q, and r, we have a *NotInPie* certificate which certifies that point s is outside of the pie, as shown in Fig. 2. When the certificate fails, we replace pq by rs. In general, when a corresponding certificate of each quadrilateral fails, we perform an edge flip.

Each i-pie has three edges. By removing one of the edges and extending the other two edges a cone can be created, call these cones i-cones, see Fig. 3(a). An edge pq is an i-hull edge if and only if there exists an i-cone such that p and q are on its boundary and the i-cone does not contains any other points—see the edge pq in Fig. 3(b). Each edge pq of the i-hull is incident to at most four other i-hull edges, call them by ps_1, ps_2, qs_3, qs_4, and incident to at most one triangle. Let r be the third vertex of this triangle if it exists; r can be one of the s_i where $0 \leq i \leq 4$. For the i-cone passing through p and q, we maintain at most five *NotInCone* certificates certifying that r and s_i's are outside of the i-cone. Whenever a *NotInCone* certificate fails we either delete or insert a vertex into the i-hull and then, we delete or insert an edge into the triangulation.

Thus, when a *NotInCone* certificate or a *NotInPie* certificate fails we replace the invalid certificates with the new valid ones which causes a constant number

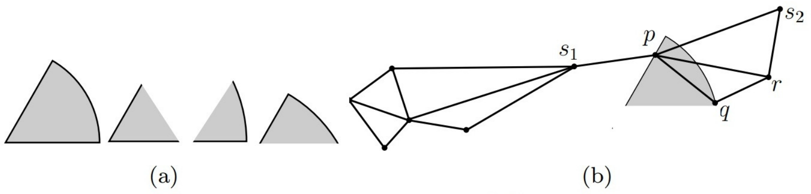

Fig. 3. (a) An i-pie and the three types of cones defined by it. (b) A hull edge pq corresponds to an i-cone.

of changes to the data structure, because the number of the invalid certificates is constant. After the updating, we also have to calculate the next failure times of the new valid certificates and place them in the queue which takes $O(\log n)$ time; the first element of the queue shows the next time that a certificate will be invalid. Thus, the following lemma results from the above discussion.

Lemma 1. *A change in the Pie Delaunay graph happens when a* NotInPie *certificate or a* NotInCone *certificate is invalid. The Pie Delaunay graph can be maintained kinetically in $O(\log n)$ time per event.*

Proof. Each edge of the *Pie Delaunay graph* is either an interior edge or an external (i-hull) edge. For the interior edge pq there exists an i-pie which p and q are on its boundary and it does not contain any other points. If we scale the i-pie such as p and q are on its boundary then a new point (r) will be incident to the boundary of the i-pie. In this case we need to define a certificate certifying p, q, and r are on the boundary of the i-pie and it does not contain any other points from P (*NotInPie* certificate). Similarly, for the external edge $p'q'$, we define a certificate certifying p' and q' are on the boundary of an i-cone and it does not contain any other points (*NotInCone* certificate). Thus, it satisfies our definition of the *Pie Delaunay graph* and for maintenance of the *Pie Delaunay graph* and we just need to define two certificates *NotInPie* and *NotInCone.*

When one of these events happens we apply a constant number of edge insertions and edge deletions into the *Pie Delaunay graph* and a constant number of changes in the event queue. So, we maintain the *Pie Delaunay graph* kinetically in $O(1)$ time per each of these events, plus $O(\log n)$ time to update the event queue. □

For a set of n points in the Euclidean plane, Guibas *et al.* [7] have shown that the number of the combinatorial changes in the Delaunay triangulation based on *circle* is $O(n^2\lambda_s(n))$. We have the following theorem about the number of the combinatorial changes of the $\mathcal{DG}(P)$ which is based on the i-pie.

Theorem 1. *The number of all changes (edge insertions and edge deletions) of the Pie Delaunay graph on a set of n moving points with trajectory of algebraic function with at most degree s is $O(n^2\lambda_{2s+2}(n))$.*

Proof. The number of convex-edge changes is $O(n^3)$ as three points are involved in any convex change. Since $n^3 = O(n^2\lambda_{2s+2}(n))$, we focus on the number of triangle changes in $\mathcal{DT}_i(P)$. For each edge pq of a triangle, four different cases are imaginable as shown in Fig. 4. It is easy to see for any triangle Δ, the case (a) of Fig. 4 happens to one of its edge. We charge any change to Δ to this edge. Therefore, we consider the number of the combinatorial changes of $\mathcal{DT}_i(P)$ for an arbitrary edge pq that satisfies case (a) of Fig. 4.

Two edges of an i-pie are line segments and one of them is an arc; call the line segments by ow_1 and ow_2. Let W_i be a wedge whose sides are created by removing the arc w_1w_2 of i-pie and extending the two line segments; the wedge W_i is the area between two half-lines $\overrightarrow{ow_1}$ and $\overrightarrow{ow_2}$. Let $\mathcal{V}(W_i)$ be the set of all points in the wedge W_i. In Fig. 4(a), a change for triangle pqr corresponding to pq happens when for some $t \in \mathcal{V}(W_i)$, the length of the edge ot becomes smaller than the length of the edge or.

Note that since the degree of each function describing each point's motion is at most s, each point of P except p and q, can be inserted inside the wedge W_i s times. Summing over all points in P there are $O(sn)$ insertion into $\mathcal{V}(W_i)$. The distance of these points from the apex o creates $O(sn)$ partial functions with at most degree $2s$. The number of the combinatorial changes corresponding to an arbitrary edge pq equals $\lambda_{2s+2}(sn)$ which is equal to the number of the breakpoints in the lower envelope of sn partial functions of at most degree $2s$ (Theorem 2.5. [13]). Since the maximum degree s is a constant, $\lambda_{2s+2}(sn) = O(\lambda_{2s+2}(n))$.

The number of all possible edges is $O(n^2)$ and therefore, the number of the combinatorial changes corresponding to all edges is $O(n^2\lambda_{2s+2}(n))$.

Besides the above changes for the edge pq, there exist other changes that happen when a point, such as s passes through the segment op or the segment oq and enters inside the area opq, see Fig. 4(a). Map each point $p = (x_p(t), y_p(t))$ to a point $p' = (u_p(t), v_p(t))$ in a new parametric plane where $u_p(t) = x_p(t) + \sqrt{3}y_p(t)$ and $v_p(t) = x_p(t) - \sqrt{3}y_p(t)$. Passing the point s through the segment op or the segment oq means that the point s' changes its u-coordinate or its v-coordinate with the u-coordinate or v-coordinate of p' or q', call these changes *swap-changes.* That is, the number of all swap-changes for all possibles is bounded with the number of all swaps between points in their ordering with respect to u-axis and v-axis. The number of the all u-swaps and v-swaps between points is $O(sn^2)$. $\square$

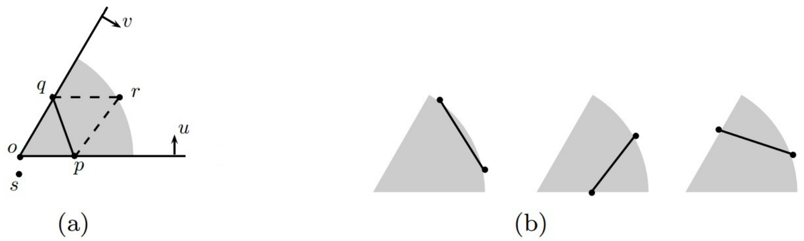

Fig. 4. Combinatorial changes for an arbitrary edge pq

From this discussion, Lemma 1, and the Theorem 1:

Theorem 2. *For a set of n points in the plane with the trajectories of algebraic functions with maximum degree s, the kinetic Pie Delaunay graph uses linear space and processes $O(n^2\lambda_{2s+2}(n))$ events, each in $O(\log n)$ time.*

3 The Applications

In this section we introduce new constructions for the *Yao graph* and the *EMST* and then, we consider the kinetic version of them.

3.1 The Constructions

For each point $p \in P$, partition the plane into k wedges $W_0(p), ..., W_{k-1}(p)$ of angle $2\pi/k$ where p is origin of the wedges and W_i spans the orientation $[2\pi i/k, 2\pi(i+1)/k]$. The *Yao graph* can be constructed by finding the closest point to p inside the wedge $W_i(p)$ where $0 \leq i \leq k-1$. For constructing the *EMST*, a version of the *Yao graph* where $k = 6$ is needed—we denote it by $\mathcal{YG}(P)$ and the set of its edges by $\mathcal{E}(\mathcal{YG}(P))$. The following lemma shows that the *Pie Delaunay graph* is a super graph of the *Yao graph.*

Lemma 2. $\mathcal{E}(\mathcal{YG}(P)) \subseteq \mathcal{E}(\mathcal{DG}(P))$.

Proof. Let W_i be a wedge whose sides are parallel to the sides of σ_i. For each point p, qp is an edge of $\mathcal{YG}(P)$ where q is the closest point to p inside $W_i(p)$, see Fig. 5. This means that, there is an i-pie where p and q are on its boundary and it dose not contain any other points of P. Therefore, $pq \in \mathcal{E}(\mathcal{DG}(P))$ and so, the *Pie Delaunay graph* includes the *Yao graph.* $\square$

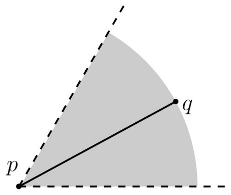

Fig. 5. An edge of a Yao graph is an edge of the Pie Delaunay Graph

Denote the *EMST* edges by $\mathcal{E}(EMST)$. In previous section, we noticed, using the approach of [12], $\mathcal{DG}(P)$ can be constructed in $O(n \log n)$ time. Cardinality of the $\mathcal{E}(\mathcal{DG}(P))$ is $O(n)$ and so, by a trace over the edges incident to each point of $\mathcal{DG}(P)$, we can construct the $\mathcal{YG}(P)$ in $O(n)$ time. $\mathcal{E}(EMST) \subseteq \mathcal{E}(\mathcal{YG}(P))$ [6] and since the number of edges in $\mathcal{E}(\mathcal{YG}(P))$ is linear, the *EMST* can be constructed in $O(n \log n)$ time using the Prim or Kruskal algorithm [14, 15] and so, the following lemma results.

Lemma 3. *Using linear space, the Yao graph and the EMST can be constructed in $O(n \log n)$ time.*

3.2 Kinetic Yao Graph

Now, assume the points start moving. To maintain $\mathcal{YG}(P)$ during the motion we introduce the *kinetic tournament tree* [4] which is a preliminary tool in the kinetic data structure framework.

Using a kinetic tournament tree, we can maintain the lowest point among a set of n moving points along the y-axis. The tournament tree on a set of n points is a balanced tree with the points stored at its leaves (in an arbitrary order). An internal node of the tournament tree maintains the lowest point between two children; the root of the tournament tree maintains the lowest point among all points. This tournament tree is known as *kinetic tournament tree* and the number of changes to the value at the root of the *kinetic tournament tree* is $\lambda_s(n)$ [3, 10]. We use a kind of tournament structure which supports insertions and deletions of points (*dynamic kinetic tournament tree*) [3]. The following theorem can be concluded from the Theorem 3.1. in [3] and it bounds the total number of events that may occur while inserting and deleting at most m points, at arbitrary locations, into a dynamic kinetic tournament.

Theorem 3. *A dynamic kinetic tournament, with a sequence of m insertions and deletions whose maximum size at any time is n (assuming $m \geq n$), generates at most $O(m\beta_{s+2}(n))$ events at the root. Processing an update or a tournament event takes $O(\log^2 n)$ worst-case time; the tournament on n elements can be constructed in $O(n)$ time.*

Let $\mathcal{E}(W_i(p))$ be the set of edges of the $\mathcal{DG}(P)$ inside the wedge $W_i(p)$ and incident to the point p. For each wedge $W_i(p)$, we have to maintain the closet point to p and so, corresponding to each $W_i(p)$ we construct a dynamic kinetic tournament $(DKT_i; i = 1, ..., 6n)$ whose elements are $\mathcal{E}(W_i(p))$. Therefore, at any time, the root of all dynamic kinetic tournaments are the edges of the *Yao graph* and so, the following theorem is resulted.

Theorem 4. *The KDS for maintenance of the Yao graph uses $O(n)$ space with preprocessing time $O(n \log n)$, and processes $O(n^2\lambda_{2s+2}(n)\beta_{s+2}(sn))$ events, each in $O(\log^2 n)$ time.*

Proof. We know that *Pie Delaunay graph* can be constructed in $O(n \log n)$ time and the cardinality of $\mathcal{E}(\mathcal{DG}(P))$ is $O(n)$. Each edge of the $\mathcal{DG}(P)$ is inserted into at most two of the DKT_i's which $i = 1, ..., 6n$. Let n_i be the number of elements in DKT_i. From Theorem 3, the construction time of DKT_i on $O(n_i)$ elements is $O(n_i)$ and so the construction time over all DKT_i's is $O(n)$. Thus, the *KDS* uses linear space with preprocessing time $O(n \log n)$.

Let m_i be the number of insertions/deletions into the DKT_i. From Theorem 1 we know that $\Sigma_{i=1}^{6n} m_i = O(n^2\lambda_{2s+2}(n))$. According to the Theorem 3, the number of all changes at the root of all DKT_i for $i = 1, ..., 6n$ is $\Sigma_{i=1}^{6n} O(m_i\beta_{s+2}(n)) = O(\beta_{s+2}(n)\Sigma_{i=1}^{6n} m_i) = O(n^2\lambda_{2s+2}(n)\beta_{s+2}(n))$ which each one can be handled in $O(\log^2 n)$ time. □

In our algorithm, we processed a nearly cubic number of events for maintenance of the *Yao graph* but the exact number of changes to the *Yao graph* is nearly square. For linearly moving points in the plane, Katoh *et al.* [16] showed the changes to the *Yao graph* is $O(n\lambda_4(n))$. In the following theorem we bound the number of the combinatorial changes of the *Yao graph* of a set of moving points with the trajectory of algebraic function of at most degree s.

Theorem 5. *The number of all changes in the Yao graph, when the points move with polynomial trajectory of at most degree s, is $O(n\lambda_{2s+2}(n))$.*

Proof. For an arbitrary point $p \in P$, each of other points of P can be inserted inside the wedge $W_i(p)$ s times and so, there exist $O(sn)$ insertion into the wedge $W_i(p)$. The distance of these points from p creates $O(sn)$ partial functions with at most degree $2s$; the lowest envelope of these partial functions corresponds to the closest point to p inside the wedge $W_i(p)$. The number of all changes in the lower envelope of sn partial functions with at most degree $2s$ corresponding to the point p is $\lambda_{2s+2}(n)$ (Theorem 2.5. [13]). Hence, the number of all changes to the *Yao graph* on a set of n moving points is $O(n\lambda_{2s+2}(n))$. $\square$

3.3 Kinetic EMST

Our approach to maintain the *EMST* is based on the fact that the edges of the *EMST* are a subset of the edges of the *Yao graph*; A change in the combinatorial structure of the *EMST* depends on the orderings of the edge weights of the *Yao graph* edges.

Here, we maintain the edges of $\mathcal{YG}(P)$ (which are the root of DKT_i where $i = 1, ..., 6n$) in a sorted list ($L_{\mathcal{YG}}$) and whenever the ordering of two edges in this list is changed, we apply the required changes to the *EMST*. Therefore, we need to track these changes to update and maintain the *EMST* of a set of moving points. In particular, to maintain the *EMST* there exists two kinds of events that we should consider:

(a) edge insertion and edge deletion from $L_{\mathcal{YG}}$, and
(b) the change between two consecutive edges in $L_{\mathcal{YG}}$.

In the case (a), as soon as an edge is deleted from $L_{\mathcal{YG}}$ the new one is inserted; both of the deleted edge and the inserted edge are in the same dynamic kinetic tournament and have a common endpoint, call them by pq and pr. In this case, the deleted edge pq can be one of the *EMST* edges and so, we have to find a new edge reconstructing the *EMST*. It's easy to show that the new edge reconstructing the *EMST* is pr.

Now, we consider the case (b). Let $path(p_i, p_j)$ be the simple path between p_i and p_j in the *EMST* and $|e|$ be the Euclidean length of e. A change in $L_{\mathcal{YG}}$ corresponds to a pair of edges e and e' in $\mathcal{E}(\mathcal{YG}(P))$ where at time t^-, $|e| < |e'|$, and at time t^+, $|e| > |e'|$. Then, at time t, e may be replaced by e' in $\mathcal{E}(EMST)$. It is simple to prove the following lemma:

Lemma 4. *EMST changes if and only if at time t^-, $|e| < |e'|$, $e \in \mathcal{E}(EMST)$, $e' \notin \mathcal{E}(EMST)$, and $e \in path(p_i, p_j)$ where p_i and p_j are the end points of e' and at time t^+, $|e| > |e'|$.*

Such events can be detected and maintained within $O(\log n)$ time per operation using the link-cut tree data structure of [17].

Theorem 6 below bounds the number of the events of the *EMST* in our *KDS*.

Theorem 6. *The number of the combinatorial changes of the EMST in our KDS is* $O(n^2\lambda_{2s+2}(n))$.

Proof. The set of *Yao graph* edges is a superset of the set of the *EMST* edges and any change in the order of the consecutive edges in the sorted list of the *Yao graph* edges may change the *EMST*. Precisely, any change in the *Yao graph* causes insertion/deletion into the sorted list and each insertion may causes $O(n)$ changes in the *EMST*. The number of all insertions and deletions into the sorted list $L_{\mathcal{YG}}$ is $O(n\lambda_{2s+2}(n))$, see Theorem 5, and therefore, in our data structure, the number of the combinatorial changes of the *EMST* is $O(n^2\lambda_{2s+2}(n))$. □

We summarize all results of our *KDS* in the following theorem.

Theorem 7. *For a set of n moving points with polynomial trajectory of at most degree s, our KDS uses linear space and requires* $O(n \log n)$ *preprocessing time. The KDS processes* $O(n^2\lambda_{2s+2}(n)\beta_{s+2}(n))$ *events for the Yao graph and* $O(n^2\lambda_{2s+2}(n))$ *events for the EMST and each of these events can be handled in the worst case time* $O(\log^2 n)$.

4 Conclusion

In the paper, we presented the new proximity graph *Pie Delaunay graph* and then we maintained the kinetic data structure for the *Yao graph.* In our *KDS*, we process the number of nearly cubic events for the kinetic *Yao graph* but the exact number of the changes in the *Yao graph* is nearly square and so, finding a *KDS* which processes only nearly square events is a future direction.

For kinetic *EMST*, we handle a nearly cubic upper bound of topological changes but the tight upper bound is not known. For linearly moving points in the plane, Katoh *et al.* [16] proved an upper bound of $O(n^3 2^{\alpha(n)})$ for the number of the combinatorial changes of the *EMST* which was later improved to $O(n^{8/3} 2^{\alpha(n)} \log^{4/3} n)$ by combining the results of Chan [18], and Marcus and Tardos [19]. Finding the tight upper bound for the combinatorial changes (events) of the *EMST* and finding a *KDS* for *EMST* processing the number of sub-cubic events are other future directions.

Acknowledgments. We thank Valerie King, Frank Ruskey, and Sue Whitesides for their useful conversations and comments.

References

[1] Rahmati, Z., Zarei, A.: Kinetic Euclidean Minimum Spanning Tree in the Plane. In: Iliopoulos, C.S., Smyth, W.F. (eds.) IWOCA 2011. LNCS, vol. 7056, pp. 261–274. Springer, Heidelberg (2011)

[2] Abam, M.A., de Berg, M., Gudmundsson, J.: A simple and efficient kinetic spanner. Comput. Geom. Theory Appl. 43, 251–256 (2010)
[3] Alexandron, G., Kaplan, H., Sharir, M.: Kinetic and dynamic data structures for convex hulls and upper envelopes. Comput. Geom. Theory Appl. 36(2), 144–158 (2007)
[4] Basch, J., Guibas, L.J., Hershberger, J.: Data structures for mobile data. In: Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1997, pp. 747–756. Society for Industrial and Applied Mathematics, Philadelphia (1997)
[5] Kaplan, H., Rubin, N., Sharir, M.: A kinetic triangulation scheme for moving points in the plane. Comput. Geom. Theory Appl. 44(4), 191–205 (2011)
[6] Yao, A.C.C.: On constructing minimum spanning trees in k-dimensional spaces and related problems. SIAM J. Comput. 11(4), 721–736 (1982)
[7] Guibas, L.J., Mitchell, J.S.B.: Voronoi Diagrams of Moving Points in the Plane. In: Schmidt, G., Berghammer, R. (eds.) WG 1991. LNCS, vol. 570, pp. 113–125. Springer, Heidelberg (1992)
[8] Fu, J.J., Lee, R.C.T.: Minimum spanning trees of moving points in the plane. IEEE Trans. Comput. 40(1), 113–118 (1991)
[9] Agarwal, P.K., Eppstein, D., Guibas, L.J., Henzinger, M.R.: Parametric and kinetic minimum spanning trees. In: FOCS, pp. 596–605. IEEE Computer Society (1998)
[10] Basch, J.: Kinetic data structures. PhD Thesis, Stanford University (1999)
[11] Basch, J., Guibas, L.J., Zhang, L.: Proximity problems on moving points. In: Proceedings of the Thirteenth Annual Symposium on Computational Geometry, SCG 1997, pp. 344–351. ACM, New York (1997)
[12] Chew, L.P., Dyrsdale III, R.L.S.: Voronoi diagrams based on convex distance functions. In: Proceedings of the First Annual Symposium on Computational Geometry, SCG 1985, pp. 235–244. ACM, New York (1985)
[13] Agarwal, K.P., Sharir, M.: Davenport–schinzel sequences and their geometric applications. Technical report, Durham, NC, USA (1995)
[14] Kruskal, J.B.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proceedings of the American Mathematical Society, 7 (1956)
[15] Prim, R.C.: Shortest connection networks and some generalizations. Bell Systems Technical Journal, 1389–1401 (November 1957)
[16] Katoh, N., Tokuyama, T., Iwano, K.: On minimum and maximum spanning trees of linearly moving points. In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, SFCS 1992, pp. 396–405. IEEE Computer Society, Washington, DC (1992)
[17] Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. Syst. Sci. 26(3), 362–391 (1983)
[18] Chan, T.M.: On levels in arrangements of curves. Discrete and Computational Geometry 29, 375–393 (2003)
[19] Marcus, A., Tardos, G.: Intersection reverse sequences and geometric applications. J. Comb. Theory Ser. A 113(4), 675–691 (2006)