Spanning Colored Points with Intervals

Payam Khanteimouri * Ali Mohades* Mohammad Ali Abam † Mohammad Reza Kazemi*

Abstract

We study a variant of the problem of spanning colored objects where the goal is to span colored objects with two similar regions. We dedicate our attention in this paper to the case where objects are points lying on the real line and regions are intervals. Precisely, the goal is to compute two intervals together spanning all colors. As the main ingredient of our algorithm, we first introduce a kinetic data structure to keep track of minimal intervals spanning all colors. Then we present a novel algorithm using the proposed KDS to compute a pair of intervals which together span all the colors with the property that the largest one is as small as possible. The algorithm runs in $O(n^2 \log n)$ using $O(n)$ space where n is the number of points.

1 Introduction

Background. In the view of location planning, suppose there are n facilities with k types like banks, network access points, etc. in the plane. Each type t can be represented by a unique color $c(t)$, i.e., each facility of type t is colored with $c(t)$. In some applications, accessing to one representative of each type suffices which means location planning is defined based on types rather than facilities in these applications. One basic problem arises here is to find a location where there is at least one representative of each type in its nearby. This suggests the problems of computing *the smallest area/perimeter color-spanning objects*. A region is said to be *color-spanning* if it contains at least one point from each color.

The other motivation specially for 1D colored points comes from planning a toolpath in layered manufacturing [4, 11]. In this application, a 3D object is defined by its layers in the plane and each layer consists of contours or polygons. To construct an object, a toolpath like a laser beam cuts the boundary of contours one by one. In practice, the laser beam moves in an straight line from one contour to other and each contour is traced only once. A trace path contains all contours and the line segments connecting them. Since the straight lines indicate the wasted time which is significant, toolpath planning is computing the trace path which minimize the total wasted time. In particular, when each contour is almost a single point, the problem is computing the TSP and clearly is NP-Hard. To obtain a heuristic method, Tang and Pang [11] proposed an algorithm in which they compute the *smallest color-spanning interval* for a set of colored points on the real line. Beside these two motivations, studying on spanning colored points has other applications in imprecise data, statistical clustering, pattern recognition and generalized range searching [6, 7, 9, 10].

Since the main ingredient of algorithms is a KDS for maintaining the minimal color-spanning intervals, we here sketch an overview of Kinetic Data Structures. A *kinetic data structure (KDS)* is a structure for keeping the trajectory of an *attribute* e.g. the sorting, for moving objects. We define a set of *certificates* for a KDS which are some boolean functions. Indeed, the set of certificates is a *proof scheme* for the attribute which means the validity of all certificates leads to correctness of the attribute. Therefore, when an event occurs, i.e., a certificate fails, we should update the KDS. Thus, we use a priority queue like a min heap to store the failure times of the events. A KDS is analysed with the following concepts. We say a KDS is *compact* if it totally uses $O\big(n \text{ polylog}(n)\big)$ space and is *local* if each object participates in $O\big(\text{polylog}(n)\big)$ certificates. In addition, a KDS is *responsive* if it can be updated in $O\big(\text{polylog}(n)\big)$ time when an event occurs. The event which changes the attribute is an *external event* and any KDS should be updated in its failure time. Moreover, there may be an *internal event* which means our KDS should be updated while the attribute remains unchanged. A KDS is said to be *efficient* if the ratio of the number of all events and the number of external events is $O\big(\text{polylog}(n)\big)$.

Related works. In the view of imprecise data, for a set of n points with k colors, the main problem is computing exactly k points with different colors in which a property like diameter, closest pair, and etc. is minimized/maximized. C. Fan et al. [6] showed that the problem of computing *the largest closest pair* is NP-Hard under the L_p metric ($1 \leq p < \infty$) even for 1D points. They [6] also proposed an algorithm with $O(n \log n)$ expected time for computing *the maximum diameter*.

In the view of location planning, for a given set of

*Laboratory of Algorithms and Computational Geometry, Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), {p.khanteimouri,mohades,mr.kazemi}@aut.ac.ir

†Department of Computer Engineering, Sharif University of Technology, abam@sharif.edu

n colored points with k colors in the plane, computing *the smallest color-spanning axes parallel rectangle* is the most studied problem. Abellanas et al. [1] proposed an algorithm of $O\big((n-k)^2 \log^2 k\big)$ time and $O(n)$ space while they showed there are $\Theta\big((n-k)^2\big)$ minimal color-spanning rectangles in the worst case. Das et al. [5] have recently improved the running time to $O(n^2 \log n)$. They [5] also present an algorithm in $O(n^3 \log k)$ time and $O(n)$ space to solve the arbitrary oriented case. Another studied problem by these papers is computing *the smallest color-spanning strip*. Das et al. [5] propose an algorithm in $O(n^2 \log n)$ time and $O(n)$ space using the dual of the given points to solve the problem. The results are near efficient with respect to testing all minimal objects. Recall that, a *minimal color-spanning object* contains at least one point from each color and any sub-region of it does not contain all colors.

In addition, Abellanas et al. [2] defined *the farthest colored Voronoi diagram (FCVD)*. For a set of n colored sites with k colors in the plane, the FCVD is the subdivision of the plane in which for any region R there is a unique site p such that any color-spanning circle centered at a point in R must contain p. Therefore, to compute the *smallest color-spanning circle* a simple algorithm is to compute the FCVD and test circles centered at the vertices of FCVD. They [2] proposed an algorithm with $O\big(n^2 \alpha(k) \log k\big)$ time to compute the FCVD and the smallest color-spanning circle. The other approach mentioned by Abellanas et al. [1] is to obtain the smallest color-spanning circle and the *smallest color-spanning axes-parallel square* in $O(kn \log n)$ time and $O(n)$ space using the upper envelope of Voronoi surfaces [8].

Computing the smallest color-spanning interval has been widely studied by Chen and Misiolek [4]. For a set of n points with k colors on the real line they [4] showed that minimal color-spanning intervals form a strictly increasing sequence. Due to this property they proposed two algorithms for computing the smallest color-spanning interval. The first algorithm simply compute the smallest color-spanning interval by a left to right sweeping in $O(n)$ time and space apart from sorting. Next, they assumed that points are given one by one in a sorted order and each point must be processed only once which is suitable for an online processing. They [4] proposed an algorithm of $O(n)$ time and $O(k)$ space.

Our results. In this paper, we study on spanning a set of n points with k colors on the real line by two intervals. We first assume points are moving on $\mathbb{R}^1$. We design a kinetic data structure for keeping the track of all minimal color-spanning intervals. We show our KDS is efficient, responsive, local and compact. Next, we use this result to propose an algorithm to compute two intervals which together span all colors and the largest one is as small as possible. The algorithm runs in $O(n^2 \log n)$ time and $O(n)$ space.

This paper is organized as following. In Section 2 we show how to keep track of all minimal color-spanning intervals for a set of moving colored points on $\mathbb{R}^1$. Next, in Section 3 we propose an almost efficient algorithm to compute two intervals together spanning all colors and the largest one is as small as possible. Finally we conclude in Section 4.

2 Minimal Color-Spanning Intervals for Moving Points

We first pay our attention to static points on the real line and then switch to moving points where the trajectory of each colored point is a polynomial function with degree at most s. For moving points, we are interested in maintaining all minimal color-spanning intervals. Since the problem is trivial for $k = 2$ we assume $k > 2$.

Static Points. Let $\mathcal{P} = \{p_1, p_2, \cdots, p_n\}$ be a set of n colored points with k colors on the real line. We assume points are in general position which means no two points coincide. A *minimal color-spanning interval (MCSI)* is an interval containing all colors and any sub-interval of it, does not contain all colors. From the definition, we can immediately deduce that the colors of the start point and the end point of an MCSI are different and unique in the interval. We first present the following lemma.

Lemma 1 *For a set of n points with k colors on the real line, there are at most $n-k+1$ minimal color-spanning intervals.*

Proof. Let p_i be the start point of an MCSI $[p_i, p_j]$. Obviously, p_i cannot be the start point of another MCSI $[p_i, p'_j]$ due to the minimality of both intervals. Therefore, each MCSI can be uniquely charged to its start point. On the other hand, the start point cannot be among the $k-1$ rightmost points as MCSI must contain at least k points. These together show the number of MCSIs is at most $n-k+1$. To give a tight lower bound, consider the sequence $1, 2, \cdots, k$ repeated $\frac{n}{k}$ times. The number of MCSIs in this example obviously is $n-k+1$. □

To compute the MCSIs, it suffices to sweep the points from left to right with two sweep lines. The sweep lines stop at the start and the end points of an interval. To recognize if the sweep lines indicate an MCSI, we use an array for keeping the number of points from each color and a variable for the number of different colors between the two sweep lines. From the fact that both sweep lines go forward in each step, the algorithm runs is $O(n)$ time and space apart from sorting. We avoid the details due to the simplicity and conclude the following lemma.

Lemma 2 *For a set of n points with k colors on the real line, all minimal color spanning intervals can be computed in $O(n)$ time and space apart from sorting.*

Moving Points. We now concentrate on maintaining MCSIs while points are moving continiously on the real line. Let $p(t)$ be the position of point p at time t. We define the following ordered sets:

- $\mathcal{P}(t) = \{p_{i_1}, \cdots, p_{i_n}\}\ ;\ p_{i_1}(t) < \cdots < p_{i_n}(t)$,
- $M(t) = \big\{[p_i, p_j] \mid [p_i, p_j] \text{ is an MCSI at time } t\big\}$.
- $\mathcal{S}(t) = \big\{p_i \mid [p_i, p_j] \in M(t)\big\}$,
- $\mathcal{E}(t) = \big\{p_j \mid [p_i, p_j] \in M(t)\big\}$.

Indeed, $\mathcal{P}(t)$ is the increasingly ordered list of the moving points according to their positions at time t. Moreover, $M(t)$ is the set of all MCSIs $[p_i, p_j]$ at time t. Since for any two MCSIs $[p_i, p_j]$ and $[p_s, p_t]$, we have either $p_i < p_s$ and $p_j < p_t$ or $p_s < p_i$ and $p_t < p_j$ due to the minimality, $M(t)$ can be recognized as an ordered list based on MCSI's start points. We have also stored the start and respectively the end points of MCSIs at time t in distinct sets $\mathcal{S}(t)$ and $\mathcal{E}(t)$. In addition, let $c(p)$ denotes the color of point p, $pred(p)$ and $suc(p)$ be the previous and respectively the next point of p with color $c(p)$.

Now, we show how $M(t)$ changes while the points are moving. It is obvious while the sorted list of points, $\mathcal{P}(t)$, does not change which means there is no swap between two consecutive points in $\mathcal{P}(t)$, $M(t)$ remains unchanged. Therefore, we maintain a kinetic sorting for moving points in $\mathcal{P}$ and explain how to handle an event in the kinetic sorting where two consecutive points p and q swap.

To handle events, we start with a useful lemma. Suppose there are two MCSIs $I_i = [p_i, p]$ and $I_j = [p_j, q]$ in $M(t)$ such that p and q are consecutive points in $\mathcal{P}(t)$ $(p < q)$ —see Figure 1. Therefore, I_i and I_j should also be consecutive in $M(t)$. Since the intersection of I_i and I_j contains exactly $k-1$ colors $\big($all colors except $c(q)\big)$, the color of point p_i should be the same as the color of q, i.e, $c(p_i) = c(q)$. In addition, $p_i = pred(q)$ which means p_i is the previous point of q with color $c(q)$. Put together we obtain the following result.

Lemma 3 *Suppose there are two intervals $I_i = [p_i, p]$ and $I_j = [p_j, q]$ in $M(t)$ such that p and q are consecutive in $\mathcal{P}$, then $c(p_i) = c(q)$ and $p_i = pred(q)$.*

Now, we concentrate on cases in which two consecutive points p and q swap their positions in $\mathcal{P}(t)$. In all cases, when $M(t)$ changes we update the sets $\mathcal{S}(t)$ and $\mathcal{E}(t)$. The arising cases are as following.

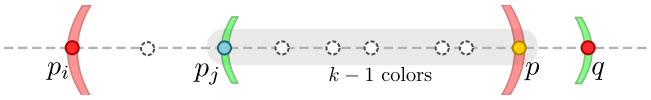

Figure 1: MCSIs $I_i = [p_i, p]$ and $I_j = [p_j, q]$ for two consecutive points p and q.

1. $p \notin \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \notin \mathcal{S}(t) \cup \mathcal{E}(t)$.
   In this case, $M(t)$ does not change. If $c(p) = c(q)$, we update *pred* and *suc* of p, q and the adjacent points.

2. $p \notin \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \in \mathcal{S}(t) \cup \mathcal{E}(t)$.
   Let I be the MCSI whose one of endpoints is q. We can distinguish 4 sub-cases. In sub-cases (a) and (b) p is inside I before the event and in sub-cases (c) and (d) p is outside I before the event.

   (a) p is inside $I = [p_i, q]$ and is the only point with color $c(p)$ in I. Since p is not the end point of an MCSI, according to Lemma 3 there is no point v in the left of p_i such that $c(v) = c(q)$. In this case, if there is a point u with color $c(p)$ in the left of p, precisely $u = pred(p)$, we first update $[p_i, q]$ to $[p_i, p]$. Then, we insert MCSI $[u, q]$ in $M(t)$ —see Figure 2(a). If u does not exist no new MCSI is inserted.

   (b) p is inside I and there is a point u with color $c(u) = c(p)$ in I. In this case $M(t)$ does not change —see Figure 2(b).

   (c) p is outside $I = [p_i, q]$ and $c(p) \neq c(q)$. since I is an MCSI, there should be point u inside I with color $c(u) = c(p)$ —see Figure 2(c). Recall that $u \neq p_i$ according to Lemma 3. Therefore, $M(t)$ remains unchanged.

   (d) $c(p) = c(q)$ —see Figure 2(d). In this case, we first update *pred* and *suc* of p, q and adjacent points. Then, we change the interval $[p_i, q]$ to $[p_i, p]$ in $M(t)$.

3. $p \in \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \notin \mathcal{S}(t) \cup \mathcal{E}(t)$.
   This case can be handled in the same manner in case 2.

4. $p \in \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \in \mathcal{S}(t) \cup \mathcal{E}(t)$.
   We can distinguish two cases based on whether $p, q \in \mathcal{E}(t)$ or $p \in \mathcal{S}(t)$ and $q \in \mathcal{E}(t)$.

   (a) $p, q \in \mathcal{E}(t)$. As Figure 3(a) illustrates, since $[p_i, p]$ is not an MCSI after the swap, we first remove the interval $[p_i, p]$ from $M(t)$ and then we update the interval $[p_j, q]$ to $[p_j, p]$.

   (b) $p \in \mathcal{S}(t), q \in \mathcal{E}(t)$. To handle this case, p affects the interval $[p_i, q]$ as an ordinary point.

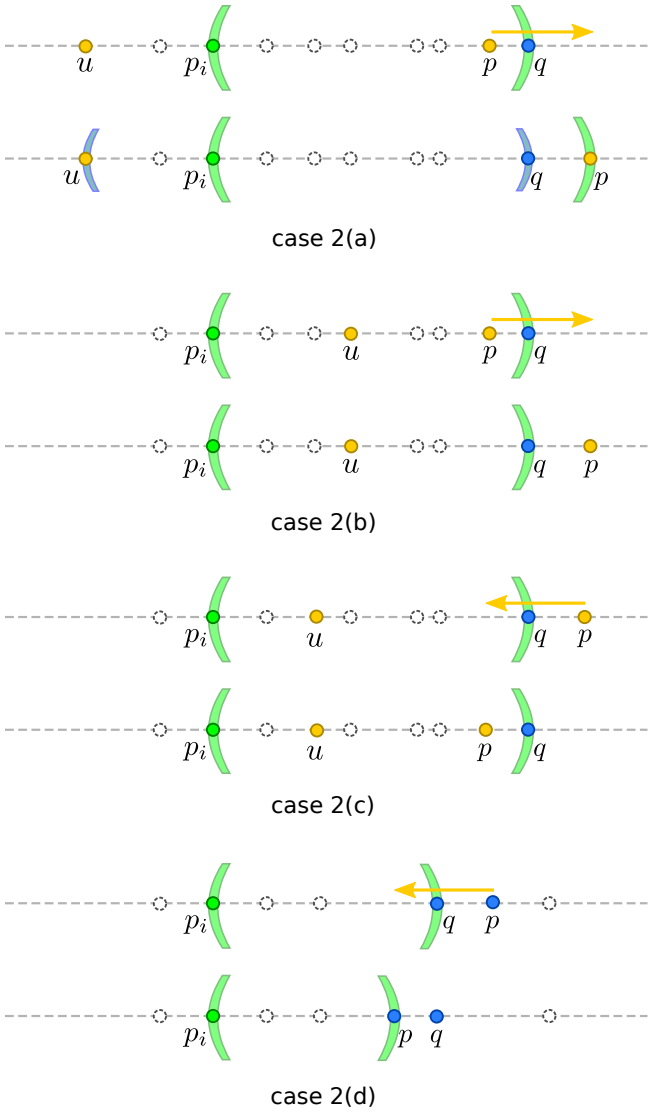

Figure 2: $p \notin \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \in \mathcal{S}(t) \cup \mathcal{E}(t)$.

Thus, we consider p as a point inside the interval $[p_i, q]$ and handle the event based on case 2. This happens similarly for q which is inside the interval $[p, p_j]$ —see Figure 3(b). Therefore, we handle two events in the type of case 2 instead of this case. The case $p \in \mathcal{E}(t), q \in \mathcal{S}(t)$ can be handled similarly.

Note that since a point can simultaneously be start and end point of MCSIs, more than one of the above cases may be handled when one event happens. As each point can appear once as the start or the end point of MCSIs, there are constant arising cases in an event and all of them can be independently handled as described in the above cases without priority.

To test whether $p \in \mathcal{S}(t)$ in $O(\log n)$ time, we maintain a dynamic search tree (like a red-black tree) over

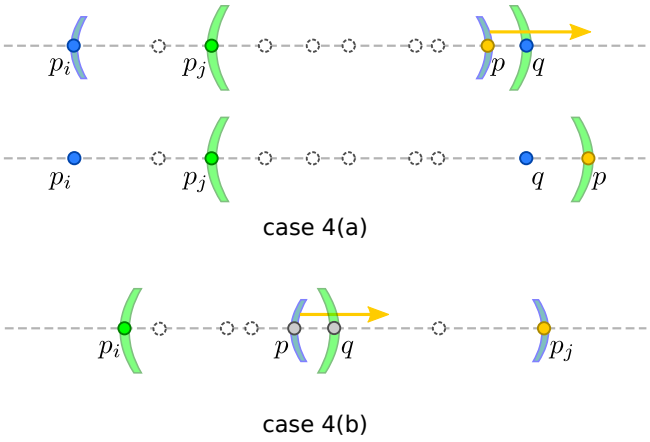

Figure 3: $p \in \mathcal{S}(t) \cup \mathcal{E}(t)$ and $q \in \mathcal{S}(t) \cup \mathcal{E}(t)$.

$\mathcal{S}(t)$ supporting deletion and insertion in $O(\log n)$ time. In each event-handling we update this tree by performing a constant number of deletions and insertions. We also need a similar search tree over $\mathcal{E}(t)$ to test whether $p \in \mathcal{E}(t)$.

Since we just use a kinetic sorting over $\mathcal{P}$, our KDS is obviously compact and local. Moreover, as described above, each event can be handled in $O(\log n)$ time. In the worst cast, we handle $O(n^2)$ events under the assumption that the trajectory of each point is a bounded degree polynomial. Putting all together we conclude the following theorem.

Theorem 4 *For a set of n moving colored points with k colors on the real line, there is an efficient, responsive, local and compact KDS which keeps the track of all minimal color-spanning intervals.*

Proof. In order to show our KDS is efficient, we give a configuration of moving points where MCSIs change $\Omega(n^2)$ times when trajectories are bounded degree polynomials. Consider $\frac{n}{2}$ static points with color 1 and $\frac{n}{2}$ moving points with color 2 passing through all static points. When two points of different colors swap, definitely an MCSI changes. Therefore, the total number of external events in this example is $\Omega(n^2)$ as any point with color 1 swaps with any point with color 2. This shows our KDS is efficient as it handles $O(n^2)$ events in the worst case.

□

3 Computing the Smallest Color-Spanning Two Intervals

We now present our novel algorithm to compute *the smallest color-spanning two intervals (SCS2I)* which is two intervals together spanning all colors with the largest one as small as possible.

We first give a naive algorithm to compute SCS2I. For each interval $[p_i, q_j]$ we can compute the smallest interval $[p_s, p_t]$ which spans the colors that are not appeared in $[p_i, p_j]$ in $O(n)$ time using Lemma 2. As there are $O(n^2)$ different intervals, this algorithm runs in $O(n^3)$ time which is far from being efficient.

We next present our main algorithm that uses the KDS described in the previous section. Suppose $\mathcal{P}$ is a set of n points on the real line, each associated with one of the given k colors. Let $\mathcal{P}'$ be obtained by translating $\mathcal{P}$ by a vector $\vec{d}$ to the right such that all points of $\mathcal{P}$ are left to all points of $\mathcal{P}'$ —see Figure 4. Now, consider the kinetic maintenance of MCSIs of the set $\mathcal{P} \cup \mathcal{P}'$ where points in $\mathcal{P}$ are static and points in $\mathcal{P}'$ move all with the same speed to the left.

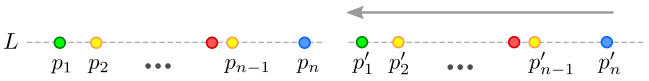

Figure 4: The copied points, $\mathcal{P}'$, move through the static original points.

Let $I(t)$ be the smallest color-spanning interval of $\mathcal{P}(t) \cup \mathcal{P}'(t)$ at time t and set I to be the smallest $I(t)$ for all t. We first show the length of I, the difference of its two endpoints denoted by $|I|$, is the solution to SCS2I and then we explain how to compute I during the kinetic maintenance of MCSIs of $\mathcal{P} \cup \mathcal{P}'$.

Suppose intervals $I_1 = [p_i, p_j]$ and $I_2 = [p_s, p_t]$ are the solution to the problem of SCS2I such that $|I_1| > |I_2|$ —see Figure 5(a). We first give the following lemma.

Lemma 5 *The length of the smallest color-spanning interval over all time, I, is equal to the length of I_1.*

Proof. Let $|I|$ be the length of the smallest color-spanning interval I during the movement of points. Since $\mathcal{P}'(t)$ is the same as $\mathcal{P}(t)$, the intervals $I'_1 = [p'_i, p'_j]$ and $I'_2 = [p'_s, p'_t]$ in $\mathcal{P}'(t)$ are also the solution to SCS2I —see Figure 5(b). Now, consider the time in which p_j and p'_t are swapped. Since the color of the endpoints of I_1 and I'_2 are unique in both intervals and together span all colors, the interval $I_1 = [p_i, p_j]$ becomes an MCSI after the swap —see Figure 5(c) for more illustration. Therefore, we conclude $|I| \leq |I_1|$.

To prove $|I| \geq |I_1|$, for the sake of contradiction assume $|I| < |I_1|$. Now, consider the time t when the length of the smallest color-spanning interval is $|I|$. Since I consists of points in $\mathcal{P}(t) \cup \mathcal{P}'(t)$ we can define two sub-intervals over the points in $\mathcal{P}(t)$ and respectively $\mathcal{P}'(t)$ which together span all colors and the length of the largest one is I which is smaller than $|I_1|$. This leads us to a better solution which is a contradiction. Therefore, we have $|I| = |I_1|$. □

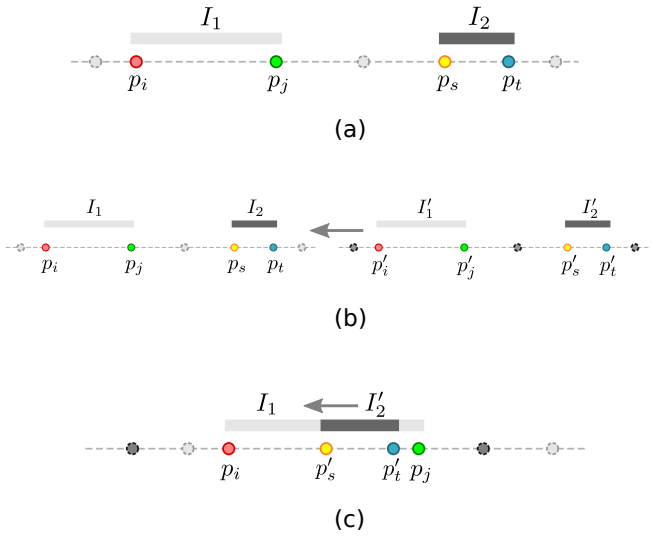

Figure 5: Intervals I_1 and I'_2 becomes an MCSI after swapping the points p_j and p'_t.

Note that the smallest color-spanning interval for all time, I, is appeared at least two times; precisely at times $I'_2 \subset I_1$ and $I_2 \subset I'_1$.

We now explain how to compute I, minimum over all $I(t)$. In general, we can maintain I using a kinetic tournament over all MCSIs at current time. Note that even there is no swap in the kinetic sorting, interval I can combinatorially change. The kinetic maintenance of $I(t)$ can be simply done by putting a kinetic tournament over intervals in $M(t)$. It is straightforward to show the kinetic sorting together with the kinetic tournament handle $O(\lambda_{s+2}(n^2)\log^2 n)$ events where s is the maximum degree of the polynomials describing the motions [3]. As we just need I, the minimum interval over all times, we can do faster as follows. Fortunately, in our setting where points of $\mathcal{P}$ are static and points of $\mathcal{P}'$ move with the same speed to the left, an MCSI reaches its minimum length when it appears or disappears from $M(t)$. Appearance or disappearance of MCSIs happens at event times where two points swap. Therefore, it suffices to take the minimum over all MCSIs' lengths at their appearance or disappearance times. This obviously can be done in $O(n^2)$ time.

Theorem 6 *For a given set of n colored points with k colors on the real line, the smallest color-spanning two intervals can be computed in $O(n^2 \log n)$ time and $O(n)$ space.*

In addition, We can show if the color of given points is a sequence of $1, 2, \cdots, k$ repeated $\frac{n}{k}$ times, there are $\Omega(n^2)$ minimal color-spanning two intervals. So, our algorithm is near efficient with respect to testing all minimal color-spanning two intervals.

4 Conclusion

For a set of n colored points with k colors on the real line, first the problem of keeping the track of minimal color-spanning intervals is studied in this paper. We present a kinetic data structure which it is efficient, responsive, local and compact. Then, we use this result to compute two intervals which they span all colors together and the length of the largest one is minimum. This is a novel idea which solves a static problem from the kinetic interpretation of it. We propose an algorithm which compute the smallest color-spanning two intervals in $O(n^2 \log n)$ time and $O(n)$ space.

References

[1] M. Abellanas and F. Hurtado and C. Icking and R. Klein and E. Langetepe and L. Ma and B. Palop and V. Sacristán. Smallest Color-Spanning Objects. *ESA, Springer-Verlag*, 278–289, 2001.

[2] M. Abellanas and F. Hurtado and C. Icking and R. Klein and E. Langetepe and L. Ma and B. Palop and V. Sacristán. The Farthest Color Voronoi Diagram and Related Problems. *tech. report. University of Bonn.*, 2006.

[3] J. Basch. *Kinetic Data Structures.* PhD thesis, 1999.

[4] D. Z. Chen and E. Misiolek. Algorithms for interval structures with applications. *Proceedings of the 5th joint international frontiers in algorithmics, FAW-AAIM'11, Springer-Verlag*, 196–207, 2011.

[5] S. Das and P. P. Goswami and S. C. Nandy. Smallest Color-Spanning Object Revisited. *Int. J. Comput. Geometry Appl.*, 19:457–478, 2009.

[6] C. Fan and W. Ju and J. Luo and B. Zhu. On some geometric problems of color-spanning sets. *Proceedings of the 5th joint international frontiers in algorithmics, and 7th international conference on Algorithmic aspects in information and management. FAW-AAIM'11. Springer-Verlag*, 113–124, 2011.

[7] P. Gupta and R. Janardan and M. Smid. Further Results on Generalized Intersection Searching Problems: Counting, Reporting, and Dynamization. *J. Algorithms.*, 19(2):282–317, 1995.

[8] D. P. Huttenlocher and K. Kedem and M. Sharir. The Upper Envelope of voronoi Surfaces and Its Applications. *Discrete Computational Geometry*, 9:267–291, 1993.

[9] J. Matoušek. On enclosing k points by a circle. *Information Processing Letters*, 53(4):217–221, 1995.

[10] M. Smid. Finding k points with a smallest enclosing square. *MPI-I-92-152, Max-Planck-Institut Inform., Saarbr cken, Germany*, 1992.

[11] K. Tang and A. Pang. Optimal connection of loops in laminated object manufacturing. *Computer-Aided Design*, 35(11):1011-1022, 2003.