Color Spanning Objects: Algorithms and Hardness Results

Sandip Banerjee[1⋆], Neeldhara Misra[2⋆⋆], Subhas C. Nandy[1]

[1] Indian Statistical Institute, Kolkata, India
{*sbanerjee,nandysc*}*@isical.ac.in*,
[2] Indian Institute of Technology, Gandhinagar, India
neeldhara.m@iitgn.ac.in

Abstract. In this paper, we study the Shortest Color Spanning Intervals problem, and related generalizations, namely Smallest Color Spanning t Squares and Smallest Color Spanning t Circles. The generic setting is the following: we are given n points in the plane (or on the line), each colored with one of k colors, and for each color i we also have a demand s_i. Given a budget t, we are required to find at most t objects (for example, intervals, squares, circles, etc.) that cover at least s_i points of color i. Typically, the goal is to minimize the maximum perimeter or area.

We provide exact algorithms for these problems for the cases of intervals, circles and squares, generalizing several known results. In the case of intervals, we provide a comprehensive understanding of the complexity landscape of the problem after taking several natural parameters into account. Given that the problem turns out to be $W[1]$-hard parameterized by the standard parameters, we introduce a new parameter, namely sparsity, and prove new hardness and tractability results in this context. For squares and circles, we use existing algorithms of one smallest color spanning object in order to design algorithms for getting t identical objects of minimum size whose union spans all the colors.

Keywords: Color Spanning Sets, Computational Geometry, Parameterized Complexity, Exact Algorithms.

1 Introduction

We are given a set of n points on a line, each colored with one of the k colors, and a non-negative demand s_i for each color i. The problem of Shortest Color Spanning t-Intervals involves finding at most t intervals each of length at most d such that at least s_i points of every color i are covered by the union of these intervals, where t and d are positive integers. A natural generalization

⋆ The work was done while the author was visiting the Indian Institute of Science, Bangalore, India.
⋆⋆ The author is supported by the DST-INSPIRE fellowship, project DSTO-1209.

involves considering points in the plane, and attempting to meet all the demands using a collection of geometric objects in the plane that minimize some desirable parameter, for instance, the maximum perimeter. In the context of location planning, suppose there are n facilities of k types e.g. schools, banks, hospitals, etc. in a locality, the objective is to choose a suitable residential location with a representative from each facility type in the neighborhood. This situation calls for the computation of a *color spanning object* of smallest perimeter or area. In this work, we study the problems of spanning intervals, squares, and circles from an algorithmic perspective. Apart from studying the problem in the general form described above, our study covers situations involving a fixed number of objects, and also the important special case when all demands are one. In addition to proposing some polynomial time algorithms, we consider the parameterized complexity of those variants that are NP-hard in general.

In parameterized complexity, each problem instance comprises of an instance x in the usual sense, and a parameter k. A problem with parameter k is called *fixed parameter tractable* (FPT) if it is solvable in time $f(k) \cdot g(|x|)$, where f is an arbitrary function of k and g is a polynomial in the input size $|x|$. Just as NP-hardness is used as evidence that a problem probably is not polynomial time solvable, there exists a hierarchy of complexity classes above FPT, and showing that a parameterized problem is hard for one of these classes is considered as an evidence that the problem is unlikely to be fixed-parameter tractable. The main classes in this hierarchy are: $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P] \subseteq XP$. A parameterized problem belongs to the class XP if there exists an algorithm for it with running time bounded by $n^{h(k)}$, for some function h of k. For the details in *parameterized complexity* refer to [6].

We will mostly focus on the parameterized complexity of SCSI-t problem, where we are given budgets d and t, and we are required to find t intervals of length at most d each, such that their union covers at least s_i points of color i. We first show that the FPT algorithm described in [9] for the case when all demands are "1" can be extended to a FPT algorithm when parameterized by k and s^*, where s^* is the maximum demand among all the colors. This FPT result is the best we can hope for, in the following sense: if we do not incorporate the maximum demand as a parameter, we know that the problem is $W[1]$-hard even for the case of *unit intervals* if we parameterize by *both* k and t [9]. Also note that it does not make sense to parameterize by s^* alone, as the problem is NP-hard even when $s_i = 1$ for all the colors i [9]. In [9] it is also shown that for unit intervals the problem is $W[2]$-hard with respect to the parameter t even when all demands are "1". Therefore, for all combinations of natural parameters, namely k, t, d and s^*, the complexity is understood. We have FPT if we incorporate both k and s^* as parameter; but for any other combination of the parameters, we encounter hardness in the parameterized sense.

Given the overwhelming proportion of hardness results, we are motivated to look for other parameters that reflect possible structure in the point set. To this end, we introduce the following notion: we say that a point set P is (q, d)-*sparse* if any interval of length at most d covers at most q points from P. In this context, we

show that SCSI-t problem remains NP-hard on $(3,1)$-sparse point sets even when P contains at most "2" points of each color. However, we complement this by showing that the problem is FPT when parameterized by t, q and the maximum frequency with which any color repeats itself. We also generalize the $O(n^2)$ time algorithm for SCSI-2 problem [9] to the case of t intervals, with two algorithms of running times $O(n^t k)$ and $O((f(s^*,t))^k n)$ respectively, where $f(s^*,t)$ denotes the number of ordered sets of t non-negative integers that sum to s^*. Note that an $O(n^{t+2})$ time algorithm for this problem is relatively straightforward by brute-force: we can guess the two endpoints of the largest interval — this gives us a candidate upper bound on the length of the remaining intervals. Therefore, it is now enough to guess the left endpoints of all the remaining $(t-1)$ intervals, and checking, in $O(n)$ time, whether all demands are satisfied. Subsequently, we also consider the color spanning problem for two congruent objects in the plane with an objective to minimize the perimeter. Specifically, we focus on axis-parallel squares, and circles. For a given set P of n points with k colors in the plane, we can compute in $O((n^2k + \min(m^2k^{\omega-2}, mn\log^2 n))\log n)$ time two color-spanning axis-parallel squares, where m denotes $\min(2^k, n^2)$ and ω denotes the constant in the power of N in the time complexity of multiplying two $N \times N$ matrices. We also provide an algorithm with running time $O(n^3 + (n^2k\log n + \min(mnk\log n, m^2k^{\omega-2}))\log n)$ time for the case of two circles, where m is as defined earlier. These results naturally generalize the complexity of SCSI-t problem to that of finding t color spanning objects with the goal of minimizing the maximum perimeter.

Related Work. Chen and Misiolek [4] studied the problem of finding a shortest color spanning interval when the points are on a line, and they provide a linear time algorithm for the case when all points are sorted. Khanteimouri et al. [11] gave an $O(n^2\log n)$-time algorithm for the special case of SCSI-2 problem with $s_i = 1$ for all colors i. This was subsequently improved by Jiang and Wang [9] to an $O(n^2)$ time algorithm with arbitrary demands. In [9], SCSI-t problem is also studied at length from the perspective of approximation and fixed-parameter tractable algorithms. First, several hardness results are established. For instance, they show that (i) approximating SCSI-t problem within any ratio is NP-hard when t is a part of the input, (ii) is W[2]-hard when t is a parameter, and (iii) is W[1]-hard with both t and k are parameters. On the other hand, they show that SCSI-t problem with $s_i = 1$ for all i is fixed-parameter tractable with k as the parameter, and admits an exact algorithm running in $O(2^k n\max(k, logn))$ time.

In the context of points in a plane, there have been several studies for different geometric objects, and with varying objectives. In the context of location planning, a natural question is to ask for the smallest color spanning circle. This can be found in $O(kn\log n)$ time by computing the upper envelope of Voronoi surfaces [8]. Abellanas et al. [2] proposed an $O(n(n-k)\log^2 k)$ time and $O(n)$ space algorithm for the problem of compuing a smallest color spanning axis-parallel rectangle. Das et al. [5] improved the running time to $O(n\log^2 n)$. They studied

related problems like color spanning strips and rectangles of arbitrary orientations. In a subsequent work, Abellanas et al. [1] used the farthest colored Voronoi diagram (FCVD) to develop an $O(n^2\alpha(k)\log k)$ time algorithm for the smallest color-spanning circle problem. More recently, Khanteimouri et al. [10] studied the problem of computing the smallest color-spanning axis-parallel square. Their proposed algorithm runs in $O(n\log^2 n)$ time using $O(n)$ space.

2 Shortest Color Spanning Intervals

In this section, we study the Shortest Color Spanning Intervals (SCSI-t) problem. An instance of the problem is given by $(P, k, t, d, \{s_1, \ldots, s_k\})$, where P is a set of points on the line and $[k]$ is a set of colors, t and d are non-negative integers. The question is if there exists a collection of at most t intervals of length at most d each, such that they together cover at least s_i points of every color i. We will use s^* to denote $\max(s_1, \ldots, s_k)$. We use Shortest Color Spanning Unit Intervals to refer to the special case when d is fixed to be "1", i.e., we would like to meet all demands using at most t *unit* intervals. We first consider the natural parameters that arise from the problem, namely k, t, s^* and d. Subsequently, we introduce the parameters *max frequency* and *sparsity*, and analyze the problem in the presence of these additional parameters as well. We also use OPT-SCSI-t to refer to the version of the problem where d is not given as input, and the goal is to minimize d when t is fixed.

2.1 Standard Parameterizations

By Theorem 2 in [9], we know that SCSI-t problem is $W[2]$-hard when parameterized by t, even for constant values of d and s^*, and $W[1]$-hard when parameterized by both t and k, for constant d. They also show (in Theorem 3) that the problem is FPT when parameterized by k, when all demands are "1". We note that the reduction from Clique that establishes the $W[1]$-hardness, when parameterized by t and k, has some colors with unbounded demands. Hence, it is natural to ask if the problem is FPT when parameterized by both k and the maximum demand s^*. We answer this question in the affirmative by adopting a natural generalization of the dynamic programming (DP) approach proposed in [9]. We begin by recalling the following lemma.

Lemma 1 ([9]). *There must exist an optimal solution for the problem SCSI-t problem such that a longest interval has both left and right endpoints in P.*

Let w denote a word of length k over the alphabet $\{1, 2 \ldots s^*\}$ and let w^* represent the initial demand for each color, that is, $w^*[i] = s_i$ for all $i \in [k]$. For $w_1, w_2 \in \{1, 2 \ldots s^*\}^k$, we define $w_d = w_1 - w_2$ as follows:

$$w_d[j] = \begin{cases} w_1[j] - w_2[j] \text{ if } w_1[j] \geq w_2[j]; \\ 0 \text{ otherwise,} \end{cases}$$

for all $j \in [k]$.

Let $N[w, i]$ be the minimum number of intervals required to cover color j $w[j]$ times among the points $p_1, p_2 \dots p_i$. Our final answer from the DP table will be $N[w^*, n]$. To describe the recurrence in the DP we need following:

- An index $g(i)$, which is the smallest index j, $1 \leq j \leq i$, such that the points $p_j, p_{j+1} \dots p_i$ are covered by an interval I_i^r of length d with right end-point at p_i.
- An indicator vector $\hat{C}_i$, where $\hat{C}_i[j]$ denotes the number of times color j appears in the interval I_i^r.

The index $g(i)$ can be computed by scanning all the points from right to left in $O(n)$ time. To compute $\hat{C}_i[j]$, we sweep the points from right to left by using an interval I of length d, and maintaining for color j the number of points covered by the current interval I. Assume that we have already computed $\hat{C}_i[j]$ and the right end point of I is at p_i. To compute $\hat{C_{i-1}}[j]$ we simply add the number of points of color j that are covered by I_{i-1}^r but not I_i^r, and subtract "1" if the point p_i had color j. We now propose the following recurrence:

$$N[w, i] = \min(N[w - \hat{C}_i, i - g(i)] + 1, N[w, i - 1]).$$

The two considered cases in this recurrence are whether p_i is the last point covered by an interval $I = [p_{g(i)}, p]$ of length d with the color set $\hat{C}_i$, or not.

The running time of this procedure is $O((s^*)^k kn)$ since in the DP table $N[w, i]$ has $(s^*)^k n$ entries, and each entry takes $O(k)$ time to compute because $g(i)$ and $\hat{C}_i[j]$ have been computed initially and the set union operation takes $O(k)$ time. We have thus shown the following.

Theorem 1. *SCSI-t problem admits an algorithm with running time $O((s^*)^k kn)$, and therefore is FPT with respect to the parameters s^* and k.*

We also remark that the kernel lower bound stated in Proposition 1 is a direct consequence of Theorem 2 in [9], as their reduction can be viewed as a polynomial parameter transformation from Colorful Red-Blue Dominating Set to SCSI-t problem. The parameterization of the Colorful Red-Blue Dominating Set problem by the size of the solution and number of blue vertices seems unlikely to admit a polynomial kernel [12].

Proposition 1. *SCSI-t problem does not have a polynomial kernel when parameterized by k and t, even for constant values of d and s^*, unless $CoNP \subseteq NP/Poly$.*

2.2 Frequency and Sparsity

We now consider two auxiliary parameters: max frequency, and sparsity. We use f_i to denote the number of points in P that have color i. Clearly, $\sum_{i \in [k]} f_i = n$. We use f^* to denote $\max(f_1, \dots, f_k)$. We also introduce the following definition:

Definition 1. *A point set P is said to be* (q, d)-sparse*, or has sparsity q with respect to d, if any interval of length d contains at most q points from P.*

We first show that SCSI-t problem is NP-hard even when the given point set is $(3, 1)$-sparse, every color appears at most twice, and the demand of each color is one. In other words, the problem is para-NP complete by the combined parameters f^*, s^*, q and d. Subsequently, we observe that when parameterized by q, f^* and t, the problem is FPT.

Theorem 2. *SCSI-t problem for* unit intervals *is NP-hard on a $(3, 1)$ sparse point set even when there are at most two points of every color and $t = k$.*

Proof. Our reduction is from vertex cover restricted to cubic graphs [7]. Let $(G = (V, E), k)$ denote an instance of the vertex cover problem, where G is a cubic graph on n vertices and m edges, and the problem is to find a vertex cover of size at most k. We will construct n clusters of points, namely $\{\mathcal{C}_1, \mathcal{C}_2 \ldots \mathcal{C}_n\}$, on a real line corresponding to the n vertices of V. Each cluster $\mathcal{C}_\alpha$ consists of three points in an unit interval. The distance between any pair of points in two different clusters is greater than "1" (see Figure 1). Now, we map each edge $(\alpha, \beta) \in E$ to a pair of points in $\mathcal{C}_\alpha$ and $\mathcal{C}_\beta$ respectively. Also ensure that no point is being mapped for more than one edge. Now, we assign colors to the points placed on the line. We first assign distinct colors to the edges of the graph G. If an edge is of color i, then its two adjacent points are also assigned color i. Thus, we have a $(3, 1)$-sparse point set where each color appear twice. This completes the description of the construction. We now turn to a proof of equivalence.

In the forward direction, suppose G admits a vertex cover S of size at most k $(= t)$. For every vertex $v \in S$, we can choose the corresponding cluster $\mathcal{C}_v$. Since S is a vertex cover of size k, the edges are incident to the members in S span all the m colors. Thus, the corresponding clusters $\{\mathcal{C}_v, v \in S\}$ also span all the m colors at least once. Since each cluster spans unit interval, we have color spanning k intervals corresponding to the clusters $\{\mathcal{C}_v, v \in S\}$.

In the reverse direction, suppose there is a solution with at most t unit intervals of the SCSI-t problem. We choose the vertices of the graph corresponding to the clusters that are (fully or partially) spanned by these t intervals. Note that, none of these intervals spans more than one cluster. Now, suppose the chosen vertices do not form a vertex cover. This implies that our solution of t intervals of the SCSI-t problem didn't cover the representative point of those colors whose corresponding edges have been missed out in G. This contradicts the correctness of the solution of the SCSI-t problem. □

We now turn to our FPT algorithm. Let us fix one particular point p in the point set. We first argue that the number of d-length intervals that can cover p is bounded by a function of q and f^* on (q, d)-sparse point sets. We then show that SCSI-t problem, when restricted to (q, d)-sparse point sets, reduces to r-Hitting Set, where all sets have at most r elements. The r-Hitting Set

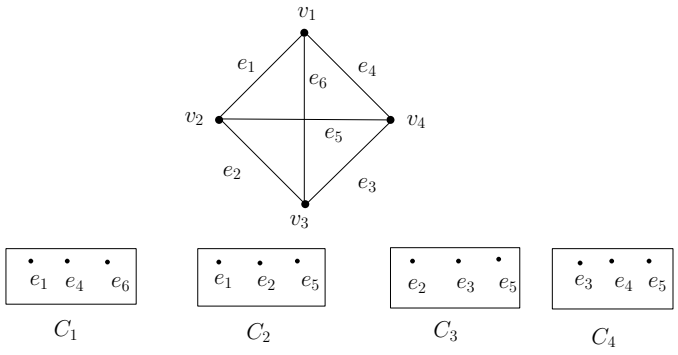

Fig. 1. Proof of Theorem 2
Here G represents the given cubic graph. 4 clusters C_i, $i = \{1, 2, 3, 4\}$ are constructed from the 4 vertices where each cluster has 3 vertices representing 3 edges incident on each vertex.

problem is FPT when parameterized by both r and the size of the solution [3]. In our reduction, r will be a function of f^* and q, while the size of the solution will be t. Therefore, this will establish that SCSI-t problem is FPT when parameterized by f^*, q and t when the point set is (q, d)-sparse and all demands are one. This complements our NP-hardness result above, which showed that the problem is para-NP-hard when parameterized by both f^* and q. We also remark that the problem is trivially FPT if parameterized by f^* and k, since $n = kf^*$. We remark that if we branch exhaustively on the sets in the family, then it is also easy to keep track of arbitrary demands. The following theorem says the conclusion of this discussion. Here we omit the proof details due to lack of space.

Theorem 3. *SCSI-t problem is FPT with respect to the parameters q, f^* and t, where q is the sparsity of the point set with respect to d, and f^* is the maximum number of points of any given color. The running time is bounded by the running time of algorithms for the (qf^*)-*Hitting Set *problem.*

2.3 Polynomial time cases and XP algorithms

We now turn to the optimization variants of the SCSI-t problem, where the length of the interval is not given as input. First we will discuss an $O(n^t k)$ time algorithm for the decision version of SCSI-t problem when d is a part of the input. For every point p in P, we maintain a 2 dimensional table T_p of size kn, where $T_p[i][q]$ stores the number of points with color i between the points p and q. But if the distance of p and q is larger than d, then $T_p[i][q]$ stores only the number of points with color i between the points $p_{g(p)}$ and p, where $g(p)$ is the smallest index j $(1 \leq j \leq p)$ such that the points $p_j, p_{j+1}, \ldots p$ are covered by an interval of length d with right end point at p.

After this preprocessing, we can choose t different points in P for placing the right endpoints of t intervals. The number of such choices is $O(n^t)$. For each choice

we check whether the demand of all colors are satisfied using the corresponding array entries. The algorithm returns **true** if there exists a set of t intervals among $O(n^t)$ such sets for which the demand is met.

.

When d is not given as input, the algorithm needs to determine the length of the longest interval that satisfies the demand of all the colors. This requires a little more computation as stated in the following result.

Theorem 4. *The optimization version of the SCSI-3 admits an* $O(\min((f(s^*))^k n, n^3 k))$ *time algorithm, where* $f(x)$ *is the number of ordered triplets of non-negative integers that sum to* x. .

Proof. For the first algorithm, we proceed as follows. For each color $j \in [k]$, let the array $\mathcal{A}_j$ store the locations of the points that are colored j in sorted order. Further, we allocate an array $\mathcal{B}$ of size $n \times k$ whose $[i, j]$-th entry contains the distances required to have s_j points of color j nearest to p_i, for $j = 1, 2, \ldots, k$, and $i = 1, 2, \ldots, n$. Each of these elements can be computed in time $O(\log n)$ using a binary search over the arrays $\mathcal{A}_j$. In each row of the array $\mathcal{B}$, the color is attached with its each element. Next, we sort each row of the array $\mathcal{B}$ with respect to their distance values. This completes our preprocessing step, and it takes $O(nk \log n)$ time.

Guess the left endpoints of the three intervals in the solution, at points say $p_i, p_j, p_\ell \in P$, in $O(n^3)$ ways. For each triple (i, j, ℓ), we execute a linear scan of the i-th, j-th and ℓ-th row of $\mathcal{B}$ to find the minimum length required for the three intervals with left end-point at p_i, p_j, p_k respectively, to satisfy the demands of all the colors. Thus, the entire algorithm needs $O(n^3 k)$ time.

For the second algorithm, we consider every demand s_i, and guess how the three intervals of an optimal solution will meet these demands. Thus, for each s_i, we create all possible ordered triplets of non-negative integers (s_i^x, s_i^y, s_i^z) such that $s_i^x + s_i^y + s_i^z = s_i$. The number of such triplets is $f(s_i) = (s_i + 1) + \binom{s_i+1}{2}$. Taking all the colors into account, we have generated $\prod_{i \in k} f(s_i) \leq f(s^*)^k$ many possibilities of creating three intervals to meet the demands of all the colors.

For each possibility $[(s_i^x, s_i^y, s_i^z), i = 1, 2, \ldots, k]$, we now have three subproblems (i) $[s_i^x, i = 1, 2, \ldots, k]$, (ii) $[s_i^y, i = 1, 2, \ldots, k]$ and (iii) $[s_i^z, i = 1, 2, \ldots, k]$ of satisfying demands with one interval. Each of these subproblems can be solved in linear time using the algorithm of [4]. So, the entire process needs $O((f(s^*))^k n)$ time.

□

The arguments above generalizes easily to give us an XP algorithms.

Corollary 1. *SCSI-t admits an algorithm with running time* $O(\min((f(s^*, t))^k n, n^t k))$, *where* $f(s^*, t)$ *denotes the number of ordered sets of* t *non-negative integers that sum to* s^*.

3 Smallest Color Spanning Squares and Circles

In this section, we will discuss the problem of finding optimal color spanning sets with respect to axis-parallel squares and circles as stated below.

SHORTEST COLOR SPANNING SQUARES (CIRCLES) PROBLEM
Input: A set of points $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^2$, and a set of colors $[k]$, a mapping $\ell : P \to [k]$, a collection of demands $\{s_1, \ldots, s_k\}$, and a non-negative integer t.
Question: Does there exist a collection $\mathcal{I}$ of at most t axis parallel squares (resp. circles), such that color i is covered at least s_i times by the union of objects in $\mathcal{I}$, and the maximum side-length (circumference) is minimized?

We use SCSS-t and SCSC-t to refer to the SMALLEST COLOR SPANNING t SQUARES and SMALLEST COLOR SPANNING t CIRCLES problems, respectively. Our focus here is on designing the exact algorithms. In the following sections, we will focus on the special case for $t = 2$. Subsequently we generalize these algorithms to XP algorithms with the parameter t. It is easy to infer the hardness of these problems based on the known hardness results for the case of intervals. Here we describe our results for squares and circles. In the context of SCSS-t problem, the following observation can be easily inferred from known results, see, for instance [10].

Observation 1 ([10]) *For the desired squares S_1 and S_2, its three edges are supported by three points of P of different colors. Thus, the size of S_1 and S_2 is determined by the vertical distance of two horizontal lines defined by a pair of bicolored points in P or the horizontal distance of two vertical lines defined by a pair of bicolored points in P.*

We first consider each pair of bicolored points and compute their horizontal and vertical distances and store them in an array D. We sort the elements of D in increasing order. This step needs $O(n^2 \log n)$ time. Consider an element $d \in D$ and a point $p \in P$. Consider a horizontal interval $[a, b]$ of length $2d$ with p at its middle most point. Now, consider the projection of the points of colors different from p that are in the rectangle of height d drawn on the base $[a, b]$. Observe that the squares with p at its bottom boundary will have their bottom-left corner at the projection points on the horizontal line segment $[a, p]$. We can compute the color content of each of these squares by sweeping the bottom-left corner of the square along $[a, p]$ in $O(n)$ time. Thus, for all such points in P, we can compute the possible squares in $O(n^2)$ time. The color content of each of these squares can be represented by a bit-vector of size k. Now, we have a set X of $O(\min(n^2, 2^k))$ possible bit vectors such that the corresponding colors of each of them can be covered by a square of size d. We need to check whether there

exists a pair of bit-vectors in X that cover all the colors. The computation of the array X needs $O(n^2k)$ time. While processing an element $d \in D$ (considering a square of size d), for each bit vector in X, we have to perform a binary search in the array X maintained as a balanced search tree of depth k to see the presence of it's complement bit vector in X. We show that the above task can be completed in two different ways with time complexities $O(n^2k + mn\log^2 n)$ and $O(n^2k + m^2k^{\omega-2})$ respectively, where $m = \min(2^k, n^2)$. In order to find the smallest feasible d, we perform a binary search on the array D, and execute the aforesaid task for the chosen values of d. Thus, we have the following theorem.

Theorem 5. *For a given set P of n points with k colors in the plane, we can compute, in $O((n^2k + \min(m^2k^{\omega-2}, mn\log^2 n))\log n)$ time, two squares S_1 and S_2 minimizing $\max(|S_1|, |S_2|)$ such that their union contains at least one point of each color from P, where $|S|$ is the perimeter of the square S, $m = \min(2^k, n^2)$, and ω denotes the (constant) power of N in the time complexity of the multiplication of two $N \times N$ matrices.*

Proof. Computation of the array D needs $O(n^2)$ time. We need to consider $O(\log n)$ values from D for testing the feasibility. Thus, to prove the theorem, we need to justify the time complexity of testing the feasibility of an element $d \in D$. We describe the following two methods for this purpose. It needs to mention that in both the methods, the computation of the array X needs $O(n^2k)$ time. The size of the array X is $m = \min(n^2, 2^k)$.

Method-1 Consider a bit-vector $x \in X$, compute $y = x'$ (the complement of x), and choose points of only the colors corresponding to the "1" entries in y. Now, execute the $O(n\log^2 n)$ algorithm [11] to compute the smallest color spanning square among the points with colors in y. Let it be of size d'. If $d' \leq d$, then d is feasible. Execute this process for all the members of X to check whether there exists at least one $x \in X$ for which d is feasible. Thus, the overall time complexity is $O(mn\log^2 n)$.

Method-2 Here, we need to identify a pair of elements $x, y \in X$ such that the bitwise **OR** of x and y gives $(1, 1, 1, \ldots, 1)$. In other words, bitwise **AND** of x' and y' produces $(0, 0, 0, \ldots, 0)$. We form a matrix $B = X'$ (complement of each vector of X) of size $m \times k$, where $m = \min(2^k, n^2)$. Now, compute $B \times B^T$ (B^T stands for the transpose of B). Thus we need to perform $(m/k)^2$ multiplications of $k \times k$ matrices, where each unit operation consists of k bitwise **AND** operations of two bit-vectors. If there exists at least one zero in the product matrix, then d is feasible. Thus, the time required to test the feasibility of an element $d \in D$ is $(m/k)^2 \times O(k^\omega) = O(m^2k^{\omega-2})$, where ω is as mentioned in the statement of the theorem.

The result follows from the fact that if $n\log^2 n \leq m$, we adopt Method 1, otherwise we adopt Method 2. □

Now we can extend our algorithm for t squares, where t ($t \geq 1$) is a parameter. Split the k colors into t color classes. The number of such splitting is t^k. For each possible split, we execute the following:

For each color-class, compute the smallest color-spanning square with colors in that class. This needs $T = O(n \log^2 n)$ time [10]. Let Δ be the largest size square required considering all the color-classes in that split.

Finally, report the minimum Δ values among all t^k possible splits. Thus, for the t squares we have the following result.

Corollary 2. *For a given set P of n points with k colors in the plane, t squares $S_1, S_2, \ldots S_t$ can be computed in $O(t^{k+1} \times T)$ time, such that their union $\cup_{j=1}^{t} S_j$ contains at least one point of each color, and $\max(|S_1|, |S_2|, \ldots, |S_t|)$ is minimized.*

We now turn to the SCSC-2 problem. We begin with the following observation.

Observation 2 *In the optimum solution of the SCSC-2 problem, the smallest among the two circles must pass either through "3" given points of different colors, or with "2" points of different colors where these two points define its diameter.*

The observation directly implies that the solution of the SCSC-2 problem will be an element of the set D of radii of (i) the disks with all possible pair of bicolored points defining their diameters, and (ii) the disks with all possible triple of points of different color on its circumference. Thus the number of elements in D is $O(n^3)$. For an element $d \in D$, we can check the feasibility of the SCSC-2 problem with radius d by executing the following steps. This needs Voronoi diagram VD_i with all the points of color i, for all $i = 1, 2, \ldots, k$. Our objective is to compute the smallest feasible d in the array D.

Step 1: [Compute the array X, (X can be defined as in the SCSS-2 problem).]
- 1.1: Consider each pair of bichromatic points. If circular arcs of radius d intersect (at a point, say c), then there exists a disk of radius d with these two points on its boundary. In that case, execute the following steps.
- 1.2: Perform Voronoi query with the point c in each VD_i, $i = 1, 2, \ldots, k$. If q_i is the point of color i nearest to c, and $\delta(q_i, c) \leq d$, then color i is covered by C. Here $\delta(.,.)$ is the Euclidean distance between two points. Thus, the colors covered by C is determined in $O(k \log n)$ time.
- 1.3: Store a bit-vector of length k whose 1-entries represent the color present.

Step 2: [Test the feasibility of d] This is similar to the method of solving the SCSS-2 problem where the computation of the smallest color spanning disks C' with all the points of colors that are not covered by C is done using the algorithm of [8].

We perform a binary search to compute the smallest feasible $d \in D$. Thus, we have the following result.

Theorem 6. *For a given set P of n points with k colors in the plane, we can compute in $T = O(n^3 + (n^2 k \log n + \min(mnk \log n, m^2 k^{\omega-2})) \log n)$ time, two congruent circles C_1 and C_2 such that at least one point $p_i \in P$ representing one of the colors k is enclosed by one of the circles C_1 or C_2 with minimizing the $\max(|C_1|, |C_2|)$, where $|C|$ is the circumference of the circle C.*

Similar with the arguments for t squares, we can extend for t circles also. We conclude this section with the following result.

Corollary 3. *For a given set P of n points with k colors in the plane, we can compute in $O(t^{k+1} \times T)$ time t circles, $C_1, C_2, \ldots, C_t$ such that at least one point of each color is enclosed in one of the circles and $\max(|C_1|, |C_2|, \ldots, |C_t|)$ is minimized. Here $|C|$ is the circumference of circle C.*

References

1. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., L. Ma, B.P., Sacristán, V.: The farthest color voronoi diagram and related problems. Tech. rep. (2006)
2. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Palop, B., Sacristán, V.: Smallest color-spanning objects. In: Proceedings of the 9th Annual European Symposium (ESA). pp. 278–289 (2001)
3. Abu-Khzam, F.N.: A kernelization algorithm for d-hitting set. J. Comput. Syst. Sci 76(7), 524–531 (2010)
4. Chen, D., Misiolek, E.: Algorithms for interval structures with applications. Theoretical Computer Sci. 508, 41–53 (2013)
5. Das, S., P.Goswami, P., Nandy, S.C.: Smallest color spanning objects revisited. Internat. J. Comput. Geom. Appl. 19, 457–478 (2009)
6. R. G. Downey and M. R. Fellows. *Parameterized complexity.* Springer-Verlag, New York, 1999.
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
8. Huttenlocher, D.P., Kedem, K., Sharir, M.: The upper envelope of voronoi surfaces and its applications. Discrete & Computational Geometry 9, 267–291 (1993)
9. Jiang, M., Wang, H.: Shortest color spanning intervals. Theoretical Computer Sci.(in Press) (2015)
10. Khanteimouri, P., Mohades, A., Abam, M., Kazemi, M.: Computing the smallest color spanning axis parallel square. In: Proceedings of the 24th International Symposium on Algorithms and Computation ISAAC. pp. 634–643 (2013)
11. Khanteimouri, P., Mohades, A., Abam, M., Kazemi, M.: Spanning colored points with intervals. In: Proceedings of the 25th Canadian Conference on Computational Geometry (CCCG). pp. 265–270 (2013)
12. Dom, M., Lokshtanov, D., Saurabh., S.: Incompressibility through colors and ids. In: Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP). pp. 378–389 (2009)