Improved Algorithms for Finding the Smallest Color-Spanning Two Squares

Chaeyoon Chung[*] Jaegun Lee[*] Hee-Kap Ahn[†]

1 Introduction

Given a set P of n points in the plane, each assigned one of k colors, a **color-spanning object** is a set of geometric shapes whose union contains at least one point of each color. In this paper, we investigate the problem of finding the smallest color-spanning pair of axis-parallel squares—that is, two axis-parallel squares whose union covers all colors while minimizing the side length of the larger one (see Figure 1). We refer to this problem as **SCSS2** throughout the paper.

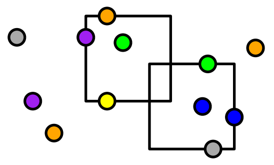
Figure 1: Smallest color-spanning two squares

We present two algorithms for SCSS2 that offer a trade-off between time and space complexity. The first algorithm runs in $O(kn^2 \log k \log n)$ time using $O(kn)$ space, while the second runs in $O(kn^2 \log^2 n)$ time usin $O(n)$ space. Both algorithms improve upon the previously best-known algorithm, which requires $O(kn^2 \log^3 n)$ time and $O(n)$ space, by reducing the time complexity.

2 Characterization

Observation 1. *There always exists an optimal pair of squares for SCSS2 such that each square has at least one input point on its bottom side.*

Both algorithms in Section 3 solve the **decision version** of the problem. Given a positive value d, the goal is to determine whether there exist two squares of side length d that span all colors.

To approach this, we first refine the problem's objective. Let p_1 and p_2 be two input points. We aim to find two squares, one with p_1 on its bottom side and the other with p_2 on its bottom side, that together maximize the number of covered colors. We then check whether all colors are covered.

By Observation 1, such color-spanning squares of side length d exist if and only if there is a pair of points that satisfies this condition. We assume that d is given as the side length of the squares for the decision problem.

Event pair Let p be an arbitrary point in the plane. Among the squares of side length d with p on its bottom side, consider the leftmost one. Now, imagine sliding this square to the right while tracking the set of colors it covers. The set of covered colors changes at specific moments, which we call **events**. For each color, there can be at most two such events.

In this paper, we focus on the case where exactly two events occur for each color, as other cases can be handled in a similar manner. We refer to the two points where the events for color α occur as the **event pair** of p for color α (See Figure 2).

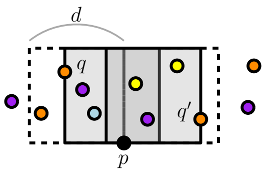

Figure 2: The event pair of p for orange color is (q, q').

We parameterize the sliding range of the square as an interval within $[0, d]$. Let p_1 and p_2 be the event pair of an arbitrary point p for color α, with x-coordinates x_1 and x_2, respectively. Then, the range in which the square contains a point of color α is given by $[0, x_1] \cup [x_2, d]$.

3 Algorithms

We present two algorithms that solve the decision version of SCSS2 for a given value $d > 0$. In both algorithms, we first fix two points $p_1, p_2 \in P$ and determine whether two squares of side length d, one with p_1 on its bottom side and the other with p_2 on its bottom side, can span all colors. We then repeat this process for every pair of points in P.

The first algorithm For each point in P, we precompute all event pairs of p for all k colors. Consider the leftmost square of side length d with p on its bottom side, and let $\mathcal{C}$ be the set of colors not covered by this square. The second square must cover $\mathcal{C}$ while having point p_2 on its bottom side.

To efficiently track coverage, we maintain a segment tree $\mathcal{T}$ storing the intervals induced by $\mathcal{C}$ and p_2. As the first square slides to the right, colors enter or leave $\mathcal{C}$ at event points, prompting insertions or deletions of corresponding intervals in $\mathcal{T}$. By checking whether the maximum depth of intervals in $\mathcal{T}$ is $|\mathcal{C}|$, we can determine whether a valid pair of squares exists.

Before introducing our second algorithm, we first present the following key observation. Let P_α denote the set of points in P of color α.

Lemma 2. *For a color α, the number of distinct event pairs in P of color α generated by points in the plane is $O(|P_\alpha|)$.*

The second algorithm To reduce the time complexity, we avoid precomputing all event pairs for each point. Using Lemma 2, we compute the subdivision of the plane for each color, where points within the same cell share the same event pair. Then, for each pair p_1 and p_2, we find the events for every color from these subdivisions using point-location queries. This approach incurs an additional $O(\log n)$ time per pair.

Theorem 3. *We can find the smallest color-spanning two squares in $O(kn^2 \log k \log n)$ time and $O(kn)$ space, or in $O(kn^2 \log^2 n)$ time and $O(n)$ space.*

[*]Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. `{chaeyoon17, jagunlee}@postech.ac.kr`
[†]Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea. `heekap@postech.ac.kr`