Red Blue Set Cover Problem on Axis-Parallel Hyperplanes and Other Objects

V P Abidha, Pradeesha Ashok
International Institute of Information Technology Bangalore, India

Abstract

Given a universe $\mathcal{U} = R \cup B$ of a finite set of red elements $R$, and a finite set of blue elements $B$ and a family $\mathcal{F}$ of subsets of $\mathcal{U}$, the Red Blue Set Cover problem is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ that covers all blue elements of $B$ and minimum number of red elements from $R$.

We prove that the Red Blue Set Cover problem is NP-hard even when $R$ and $B$ respectively are sets of red and blue points in $\mathbb{R}^2$ and the sets in $\mathcal{F}$ are defined by axis-parallel lines i.e., every set is a maximal set of points with the same $x$ or $y$ coordinate. We then study the parameterized complexity of a generalization of this problem, where $\mathcal{U}$ is a set of points in $\mathbb{R}^d$ and $\mathcal{F}$ is a collection of set of axis-parallel hyperplanes in $\mathbb{R}^d$, under different parameterizations. For every parameter, we show that the problem is fixed-parameter tractable and also show the existence of a polynomial kernel.

We further consider the Red Blue Set Cover problem for some special types of rectangles in $\mathbb{R}^2$.

Keywords:
Red blue set cover, Axis parallel hyperplane, Parameterized complexity, Polynomial kernels, Axis parallel quadrants.

1. Introduction

The Set Cover problem and its variants are well studied problems in Computer Science. Given a universe $\mathcal{U}$ with $n$ elements and a family $\mathcal{F}$ of subsets of $\mathcal{U}$, the Set Cover problem is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ that covers all elements in $\mathcal{U}$ and $|\mathcal{F}'|$ is minimized. Here $u \in \mathcal{U}$ is said to be covered by $\mathcal{F}'$, if $\exists \mathcal{F} \in \mathcal{F}'$ such that $u \in \mathcal{F}$.

Researchers have looked into many variations of this question. We study the Red Blue Set Cover problem, which is a generalization of the Set Cover problem, where $\mathcal{U}$ is bi-chromatic. Given a universe $\mathcal{U} = R \cup B$ of a finite set $R$ of red elements, and a finite set $B$ of blue elements and a family $\mathcal{F}$ of subsets of $\mathcal{U}$, the Red Blue Set Cover problem is to find a subset $\mathcal{F}' \subseteq \mathcal{F}$ that covers all blue elements of $B$ and the minimum number of red elements from $R$. Note that here the number of sets in $\mathcal{F}'$ is not optimized. The Set Cover problem can be considered as a special case of the Red Blue Set Cover problem where every set contains a distinct red element. This implies that the Red Blue Set Cover problem is as hard as the classical Set Cover problem.

The Red Blue Set Cover problem was introduced by Carr et al. [3] in 2000, motivated by applications in data mining. Several algorithmic results for the Red Blue Set Cover problem are given in [14, 5, 8, 12]. A special case of the Red Blue Set Cover problem is the Geometric Red Blue Set Cover problem, where $\mathcal{U} = R \cup B$ and a family $\mathcal{F}$ is defined by a collection of geometric objects, which contains points from $\mathcal{U}$.

In the Geometric Red Blue Set Cover problem, a wide variety of geometrical objects are studied. Chan and Hu [4] proved that the problem is NP-hard even when the objects are unit squares in $\mathbb{R}^2$, and gave the...
first polynomial-time approximation scheme (PTAS) for this problem. Pandit and Mudgal [10] have shown a constant factor approximation algorithm for the Red Blue Set Cover problem on unit disks. A study of different axis-parallel systems has been conducted by RR Madireddy et al. [11]. They showed the APX-hardness on axis-parallel set systems like axis-parallel rectangle, axis-parallel strips and axis-parallel line segments. The geometric Red Blue Set Cover problem is also studied in the context of parameterized complexity. Ashok et al. [1] investigate the following objects. We consider the Red Blue Set Cover problem on general lines in parameterized complexity of the Red Blue Set Cover problem on general lines in \( \mathbb{R}^2 \) under an array of parameters. In this paper, we mainly study the parameterized complexity of the Red Blue Set Cover problem on axis-parallel lines in \( \mathbb{R}^2 \) and axis-parallel hyperplanes in \( \mathbb{R}^d \). This can be seen as a continuation of the results given in [11], where lines and hyperplanes of arbitrary orientation are considered. We show the existence of positive results when the objects are restricted to be axis-parallel.

**Problems studied and results:**

We study the following decision variant of the Red Blue Set Cover problem.

| **Input:** A universe \( \mathcal{U} = R \cup B \), a family \( \mathcal{F} \subseteq 2^{\mathcal{U}} \), an integer \( k \).  |
| **Question:** Is there a subset \( \mathcal{F}' \subseteq \mathcal{F} \) that covers all points in \( B \) and at most \( k \) points from \( R \)? |

We consider the Red Blue Set Cover problem for the following objects.

1. **AxRBSC-LINES**: We study the Red Blue Set Cover problem when \( \mathcal{U} \) is a set of points in \( \mathbb{R}^2 \) and \( \mathcal{F} \) is defined by a set of axis-parallel lines (refer to as AxRBSC-LINES). We show that AxRBSC-LINES is NP-complete. See Section 3.

2. **AxRBSC-HYPERPLANES**: We study a generalization of AxRBSC-LINES to \( d \) dimensions. We consider the problem where \( \mathcal{U} \) is a set of points in \( \mathbb{R}^d \) and \( \mathcal{F} \) is defined by a set of axis-parallel hyperplanes. This problem is NP-complete since AxRBSC-LINES is NP-complete. We study the parameterized complexity of AxRBSC-HYPERPLANES under different parameterizations. For each parameter, we consider whether the problem is fixed parameter tractable and if yes, we investigate whether the problem admits polynomial kernels. See Section 4.

A comparison of results for arbitrary hyperplanes and axis-parallel hyperplanes is given in Table 1.

We further consider some special types of rectangles as follows.

3. **AxRBSC-SKYLINES**: Bi-directional strips refer to axis-parallel rectangles which are unbounded on two opposite sides. They are either of the form \([a, b] \times (-\infty, \infty)\) or \((-\infty, \infty) \times [a, b] \). Note that axis-parallel lines is a special case of bi-directional strips and hence the Red Blue Set Cover problem on bi-directional strips is NP-complete. We study the Red Blue Set Cover problem on a generalization of strips called skylines (referred to as AxRBSC-SKYLINES) and show that this problem is W-hard and therefore unlikely to admit FPT algorithms. See Section 5.

4. **AxRBSC-QUADRANTS**: We further consider quadrants, a special case of skylines and show that the Red Blue Set Cover problem on quadrants is polynomial time solvable. See Section 6.

| Parameter | Axis-parallel | Arbitrary |
|-----------|--------------|-----------|
| \( k \)  | FPT and poly.kernel | W[2]-hard |
| \( b \)  | FPT and poly.kernel | W[1]-hard |
| \( r \)  | FPT and poly.kernel | FPT and no poly.kernel |

Table 1: Comparison of results for arbitrary hyperplanes (taken from [1]) and axis-parallel hyperplanes.

**2. Preliminaries**

In this section, we give definitions and results that will be used in subsequent sections.

The formal definitions of objects that we consider are as follows:

**Axis-parallel hyperplane**: A hyperplane \( H \) is a subspace whose dimension is one less than that of its ambient space. Here we consider hyperplanes parallel to any one of the axis in \( \mathbb{R}^d \).
**Axis-parallel Skylines**: Skylines are axis-parallel rectangles that are unbounded in one direction. Every skyline is bounded by three axis-parallel edges. We define a horizontal skyline as a rectangle of the form \((-\infty, x] \times [y_1, y_2]\) and a vertical skyline as a rectangle of the form \([x_1, x_2] \times (-\infty, y]\) in \(\mathbb{R}^2\).

**Axis-parallel Quadrants**: Quadrants are axis-parallel rectangles that are bounded by two adjacent axis-parallel edges. We define quadrants of the first type as rectangles of the form \([x, +\infty) \times [y, +\infty)\), quadrants of the second type as rectangles of the form \((-\infty, x] \times [y, +\infty)\), quadrants of the third type as rectangles of the form \((-\infty, x] \times [y, -\infty)\) and quadrants of the fourth type as rectangles of the form \([x, +\infty) \times [y, -\infty)\). In this paper, we consider quadrants of the first type in \(\mathbb{R}^2\). Note that the results on quadrants of the first type can be extended to quadrants of other types too.

In this paper, we study the parameterized complexity of the problems using different parameters.

**Parameterized complexity** [6]: Parameterized complexity was introduced as a means to tackle NP-hardness. A parameterized problem instance is a pair \((\Pi, k)\) where \(\Pi\) is the actual problem instance and \(k\) is called a parameter. \((\Pi, k)\) is said to be fixed parameter tractable (FPT) if there exists an algorithm that solves \(\Pi\) in \(O(f(k)|\Pi|^{O(1)})\) time, where \(f(k)\) is a computable function independent of \(|\Pi|\).

Parameterized complexity is interesting when a problem is known to be NP hard. Here we aim to restrict the combinatorial explosion to a parameter, where the parameter is much smaller than the size of input.

**W-Hierarchy**: Theory of intractability of parameterized problems orders the problems into a hierarchy called the W-hierarchy based on its complexity. It is organized as \(\text{FPT} \subseteq W[1] \subseteq W[2] \cdots\). Under standard complexity theoretical assumptions, a problem which is \(W[i]\)-hard does not admit FPT algorithms, for \(i > 0\). To show intractability there exists a concept of parameterized reduction.

This technique can be used to transforms an instance of a parameterized problem \(\Pi_A\) into an instance of a parameterized problem \(\Pi_B\), mapping yes-instances for \(\Pi_A\) to yes-instances for \(\Pi_B\) and vice versa. If this transformation can be done in fixed-parameter tractable time, this implies that if \(\Pi_B\) is fixed-parameter tractable, then so is \(\Pi_A\); equivalently, if \(\Pi_A\) is not fixed-parameter tractable, then neither is \(\Pi_B\). This reduction is similar to that of polynomial reduction in classical complexity. Parameterized reductions are a powerful technique for relating problems to each other.

Another important concept related to parameterized complexity is kernelization.

**Kernelization**: A kernelization algorithm \((A)\) is usually a set of reduction rules that, given an instance \((I, k)\) of a parameterized problem \(\Pi\), runs in polynomial time and returns an equivalent instance \((I', k')\) of \(\Pi\). Moreover, we require that \(|I'| + |k'| \leq g(k)\) for some computable function \(g\) depending on \(k\).

The output instance \(I'\) is called the kernel, and the function \(g\) is referred to as the size of the kernel. If \(g(k) = k^{O(1)}\), then we say that \(\Pi\) admits a polynomial kernel. The set of reduction rules used in kernelization is said to be safe, if it converts a problem instance to an equivalent instance.

One important result in parameterized complexity is that a parameterized problem \(\Pi\) is FPT if and only if it admits a kernel. Thus for a problem which is in FPT, an interesting question is to know whether there exists a polynomial kernel or not.

Another technique used in designing of parameterized algorithms is Bounded Search Trees.

**Bounded Search Trees**: In this technique, we try to build a feasible solution to the parameterized problem by making a sequence of decisions. Nodes of a search tree correspond to problem instances, and children of a node correspond to a number of decisions that can be taken during the search. While designing an FPT algorithm, if we unravel the recursion tree, the depth of this tree and branching factor will be bounded by a function of \(k\). This will bound the total size of the tree as a function of \(k\) and the resulting algorithm is a fixed parameter tractable algorithm. Now we give some more definitions and results.

**Orthogonal Convex Hull**: A simple polygon \(C\) is said to be orthogonal convex if every vertical or horizontal line intersects the boundary of \(C\) at zero, one or two points. Orthogonal convex hull of a point set \(P\) is defined as the smallest area orthogonal convex polygon that contains \(P\).

An algorithm that finds the orthogonal convex hull of a set of \(n\) points in \(O(n \log n)\) time is presented in [2]. We define the Red Blue Set Cover problem studied in weighted settings as follows.
WEIGHTED RED BLUE SET COVER

Input: A bi-chromatic point set \( \mathcal{U} \) of blue and red elements and a family \( \mathcal{F} \) of subsets of \( \mathcal{U} \), a weight function \( w : R \to \mathbb{N} \), an integer \( k \).

Question: Is there a subfamily of \( \mathcal{F} \) that covers all blue points and \( k \) red points such that \( S \in \mathcal{F} \) covers all blue points and at most \( k \) red points. This shows that Reduction rule 3 is safe. Similarly, if \( S \) contains only blue points and \( \mathcal{F} \) is a solution that covers all blue points and \( k \) red points then \( \mathcal{F} \) is also a solution that covers all blue points and \( k \) red points. Thus, Reduction rule 3 is safe. The safety of Reduction rule 3 follows from the fact that the budget is at most \( k \) and if a set contains more than \( k \) red points, then it can not be a part of the solution.

\[ \square \]

3. NP-completeness

Theorem 3. AxBSC-lines is NP-complete.

Proof. We give a reduction from the Vertex Cover problem. Let \( G = (V, E) \) be an instance of the Vertex Cover problem, where \( |V| = n \) and \( |E| = m \). We give an instance of AxBSC-lines, \((R \cup B, \mathcal{L})\), corresponding to \( G \). For every \( v_i \in V \), we add a red point \( r_i \) to \( R \) with coordinate \((i, i) \in \mathbb{R}^2 \). Add horizontal and vertical lines that pass through \( r_i \), for \( 1 \leq i \leq n \), to \( \mathcal{L} \). For every edge \( e(v_i, v_j) \in E \), we add a blue point \( b_{ij} \) to \( B \) at the intersection of the horizontal line that passes through \( r_i \) and the vertical line that passes through \( r_j \).

We now claim that \( G = (V, E) \) admits a vertex cover of size \( k \) if and only if \((R \cup B, \mathcal{L})\) admits a solution of size \( k \). Let \( V' \subset V \) be a vertex cover of \( G \) of size \( k \). Without loss of generality, let \( V' = \{v_1, v_2, \ldots, v_k\} \). Now the set of lines in \( \mathcal{L} \) that pass through the points \( r_i \), \( 1 \leq i \leq k \), covers all the blue points and \( k \) red points.

Conversely, suppose \((R \cup B, \mathcal{L})\) has a solution that covers at most \( k \) red points, say, \( R' = \{r_1, r_2, \ldots, r_{\ell}\} \). Then the vertices of \( V \) corresponding to red points in \( R' \) is a vertex cover of \( G \).

AxBSC-lines is in NP since for a given solution of AxBSC-lines, it easy to verify whether the solution covers all blue points and covers at most \( k \) red points in polynomial time.

Now the result follows from the fact that Vertex Cover is NP-complete [9].

Corollary 4. AxBSC-Hyperplanes is NP complete.

Corollary 5. AxBSC-Bi directional strips is NP complete.
4. RBSC on Axis-parallel hyperplanes

From Corollary 2, we know that AxRBSC-Hyperplanes is NP-complete. In this section, we show fixed parameter tractability and the existence of polynomial kernels for the parameterized AxRBSC-Hyperplanes problem, under different parameterizations.

4.1. Parameterizing by \(k\)

In this section, we consider the AxRBSC-Hyperplanes problem parameterized by \(k\), where \(k\) is the number of red points in the solution. Let \((R \cup B, \mathcal{H}, k)\) be an instance of the AxRBSC-Hyperplanes problem parameterized by the number of red points \(k\), where \(R \cup B\) and \(\mathcal{H}\) are set of points and set of hyperplanes in the instance respectively.

We show the AxRBSC-Hyperplanes problem is FPT parameterized by \(k\), using the bounded search tree technique. We apply reduction rules \([1][2][3]\) and \([4]\) on \((R \cup B, \mathcal{H}, k)\) exhaustively. Now every hyperplane contains at least one red point and at least one blue point. Every blue point is covered by at most \(d\) hyperplanes. Thus, for a blue point, we check with the set of hyperplanes that passes through it in a brute-force manner and we can recursively solve the remaining problem. Since every hyperplane contains at least one red point, the associated budget \(k\) is reduced by at least one in the subproblem. The bounded search tree algorithm returns YES, when \(B = \emptyset\) else if \(k = 0\) and \(B \neq \emptyset\) it returns NO.

Thus giving us an algorithm that runs in \(O^*(d^k)\) time. The next result improves this running time.

**Theorem 6.** The problem AxRBSC-Hyperplanes can be solved in time \(O(c_d k n^{O(1)})\) where \(c_d = \frac{d - 1 + \sqrt{(d - 1)^2 + 4}}{2}\).

**Proof.** Our bounded search tree algorithm considers a blue point \(b\) such that at least one hyperplane that contains \(b\) has two red points and recursively solve the remaining problem on the hyperplane that covers \(b\) in the solution. Thus, \(k\) drops by at least 2 in at least one branch and drops by at least one in all other branches. This gives us a branching algorithm that runs in \(O(c_d^k n^{O(1)})\) time where \(c_d = \frac{d - 1 + \sqrt{(d - 1)^2 + 4}}{2}\). If the instance does not contain such a blue point (i.e., every hyperplane contains exactly one red point), then the instance can be reduced to an instance of the \(d\)-Hitting Set problem parameterized by solution size. Let \((R \cup B, \mathcal{H}, k)\) be an instance of AxRBSC-Hyperplanes such that every \(H \in \mathcal{H}\) contains exactly one red point from \(R\). We construct an instance \((U, \mathcal{F}, k)\) of \(d\)-Hitting Set with \(U = R, k = k\), and \(\mathcal{F}\) contains sets corresponding to every blue point, where a set corresponding to a blue point \(b\) contains all the red points contained in the at most \(d\) hyperplanes that contain \(b\). It is easy to see that \((R \cup B, \mathcal{H}, k)\) is a YES instance if and only if \((U, \mathcal{F}, k)\) is a YES instance. By Proposition \(\Box\), \((U, \mathcal{F}, k)\) can be solved using a branching algorithm that runs in \(O(c_d^k n^{O(1)})\) time. \(\Box\)

**Kernelization** We apply the following set of reduction rules for all integers \(\delta = 1\) to \(d - 1\) exhaustively. For \(2 \leq \delta \leq d - 1\), we apply the reduction rule corresponding to \(\delta\) only when the reduction rule corresponding to \((\delta - 1)\) is not applicable. For a point \(p \in \mathbb{R}^d\), \(x_i(p)\) denotes the value of the \(i\)th coordinate of \(p\), for \(1 \leq i \leq d\).

**Reduction rule 5** \((\delta)\). Repeat for all \(D' \subset [d]\) such that \(|D'| = d - \delta\).

Let \(B' \subseteq B\) be such that for all \(p, q \in B'\), \(x_i(p) = x_i(q)\) for all \(i \in D'\). If \(|B'| > \delta!(k)^{\delta} + 1\) then delete all but \(\delta!k^\delta + 1\) points of \(B'\).

**Lemma 7.** Reduction Rule \([5]\) is safe.

**Proof.** The proof follows from the fact that in any set of hyperplanes \(\mathcal{H}'\) that covers at most \(k\) red points, all the points of \(B'\) are covered by the same hyperplane. Assume, for contradiction, that no hyperplane in \(\mathcal{H}'\) contains all points in \(B'\). Thus \(B'\) is covered by a set of hyperplanes, say \(\mathcal{H} \subseteq \mathcal{H}'\) such that no hyperplane in \(\mathcal{H}\) is parallel to the \(i\)th co-ordinate axis, for all \(i \in D'\). Without loss of generality, assume that \(D' = \{1, 2, \ldots, d - \delta\}\). We know that every hyperplane contains at least one red point (by Reduction Rule \([2]\) and parallel hyperplanes cannot share a red point. Therefore, \(\mathcal{H}\) can have at most \(k\) hyperplanes that are parallel to the \(j\)th co-ordinate axis, for all \(j \in D'\). Thus \(\mathcal{H}\) contains at most \(k\delta\) hyperplanes.

Let \(H\) be a hyperplane in \(\mathcal{H}\) that is parallel to the \(j\)th co-ordinate axis for \(j \notin D'\). Then for any two points \(p, q \in H \cap B'\), \(x_j(p) = x_j(q)\) and \(x_i(p) = x_i(q)\) for \(1 \leq i \leq d - \delta\). Therefore, hyperplane in \(\mathcal{H}\) can contain at most \((\delta - 1)!(k)^{\delta - 1}\) points from \(B'\). Otherwise, Reduction Rule \([5]\) \((\delta - 1)\) can be applied, which is a contradiction.

Thus \(\mathcal{H}\) covers at most \(\delta!k^\delta\) points. This contradicts our assumption that \(\mathcal{H}\) covers all points of \(B'\). \(\Box\)
Theorem 8. A\textsc{Xrbsc-hyperplanes} admits a kernelization algorithm that returns \((R \cup B, \mathcal{H})\) with \(|B| \leq d!k_r^d\), \(|R| \leq d.d!k_r^{d+1}\), \(|\mathcal{H}| \leq d.d!k_r^d\).

Proof. Let \((R \cup B, \mathcal{H}, k_r)\) be an instance of A\textsc{Xrbsc-hyperplanes} such that none of the reduction rules are applicable. Then, if it is a YES instance, all the points in \(B\) can be covered by at most \(k_r.d\) hyperplanes. Any hyperplane can contain at most \(d(d-1)(d-2)\ldots(d-(d-r-1))\) blue points, otherwise Reduction Rule 3 \((d - 1)\) can be applied. Thus a YES instance can have at most \(d!k_r^d\) blue points.

Every hyperplane in \(\mathcal{H}\) contains at least one blue point and any blue point is contained in at most \(d\) hyperplanes. Therefore, \(|\mathcal{H}| \leq d.d!k_r^d\). By Reduction Rule 4, a hyperplane contains at most \(k_r\) red points. Thus \(|R| \leq d.d!k_r^{d+1}\). ∎

4.2. Parameterizing by \(b\)

Since every blue point is contained in at most \(d\) hyperplanes, the next result is easy to see by the idea of bounded search trees.

Lemma 9. The A\textsc{Xrbsc-hyperplanes} problem parameterized by \(b\) can be solved in time \(O^*(d^n.n^{O(1)})\).

Now we consider the existence of polynomial kernels for this problem.

Theorem 10. The A\textsc{Xrbsc-hyperplanes} problem parameterized by \(b\) admits a polynomial kernel.

Proof. Assume, we cannot apply reduction rules anymore on \((R \cup B, \mathcal{H}, k_r)\), every hyperplane contains at least one blue point. One blue point is covered by at most \(d\) hyperplanes. Then the number of hyperplanes \(\mathcal{H}\) is at most \(db\). To bound the number of red points we reduce an instance of the A\textsc{Xrbsc-hyperplanes} problem to an instance of the Weighted A\textsc{Xrbsc-hyperplanes} problem as follows:

The family of hyperplanes and the set of blue points remain the same as in the original instance.

Let \(R_d \subseteq R\) be the set of red points that lie in the intersection of \(d\) hyperplanes in \(\mathcal{H}\). For all \(r \in R_d\), assign \(w(r) = 1\). Since any set of \(d\) hyperplanes in \(\mathbb{R}^d\) intersect at a point, \(|R_d| \leq h^d\), where \(h\) is the number of hyperplanes.

For \(i = 1\) to \(d − 1\), we perform a reduction as follows. For all subsets of exactly \(i\) hyperplanes in \(\mathcal{H}\), say, \(H_i = \{h_1, h_2, \ldots, h_i\}\), consider the set of red points that is contained in all hyperplanes of \(H_i\) and no other hyperplane in \(\mathcal{H}\). In the reduced instance, delete all but one of these red points and assign a weight equal to the number of deleted points +1.

It easy to see that the A\textsc{Xrbsc-hyperplanes} problem instance has a solution of size \(k\) if and only if the Weighted A\textsc{Xrbsc-hyperplanes} problem instance has a solution of total weight \(k\).

In the reduced instance, the number of red points that lie in the intersection of exactly \(i\) hyperplanes is \(O(h^i)\). Therefore the total number of red points in the reduced instance is \(O(h^d + h^{d-1} + h^{d-2} + \cdots + h) = O(h^d) = O((db)^{d+1})\).

Now we bound the weight of each red point in the reduced instance. By Reduction rule 4, all hyperplanes in \(\mathcal{H}\) contain at most \(k_r\) red points. Therefore, the weight of any red point is at most \(k_r\). We see that we need at most \(h\) bits to encode the weight of a red point. If not, \(k_r > 2^h\).

Note that there exists a brute force algorithm that solves the problem in \(O^*(2^h)\) time. If \(2^h < k_r\), this is a polynomial time algorithm. Hence it follows that the weight can be encoded using \(h \leq db\) bits. Hence the size of the reduced instance is \(O((db)^{d+1})\).

There exists a polynomial-time many-one reduction from Weighted A\textsc{Xrbsc-hyperplanes} problem to A\textsc{Xrbsc-hyperplanes} problem. Thus, we obtain a polynomial-size kernel for A\textsc{Xrbsc-hyperplanes} parameterized by \(b\). ∎

4.3. Parameterizing by \(h\)

We design a parameterized algorithm and a kernel for the A\textsc{Xrbsc-hyperplanes} problem when parameterized by the number of hyperplanes, \(h\). We enumerate all possible subsets of \(\mathcal{H}\) and for each subset, we check in polynomial time whether it covers all blue points and at most \(k_r\) red points. The algorithm runs in time \(O^*(2^h(|U| + |F|))\).

Now we present a polynomial size kernel to bound the number of blue points we use the following reduction rules

To bound the number of blue points, we use the following reduction rule.

We apply the following set of reduction rules for all \(\delta = 1\) to \(d − 1\) exhaustively. For \(2 \leq \delta \leq d − 1\), we apply the reduction rule corresponding to \(\delta\) only when the
Reduction rule 6 ($\delta$). Repeat for all $D' \subseteq [d]$ such that $|D'| = d - \delta$.

Let $B' \subseteq B$ be the set of blue points such that for all $p, q \in B'$, $x_i(p) = x_i(q)$ for all $i \in D'$. If $|B'| > A_\delta$ then delete all but $A_\delta + 1$ points of $B'$.

Lemma 11. Reduction Rule 6 ($\delta$) is safe.

Proof. The proof follows from the fact that in any solution $\mathcal{H}'$, all the points of $B'$ are covered by the same hyperplane. Assume, for contradiction, that no hyperplane in $\mathcal{H}'$ contains all points in $B'$. Thus $B'$ is covered by a set of at most $h$ hyperplanes, say $\mathcal{H}_1 \subseteq \mathcal{H}'$ such that no hyperplane in $\mathcal{H}_1$ is parallel to the $i$th co-ordinate axis, for all $i \in D'$. Without loss of generality, assume that $D' = [1, 2, \ldots, d - \delta]$.

Let $H$ be a hyperplane in $\mathcal{H}_1$ that is parallel to the $j$th co-ordinate axis for $j \notin D'$. Then for any two points $p, q \in H \cap B'$, $x_j(p) = x_j(q)$ and $x_i(p) = x_i(q)$ for $1 \leq i \leq d - \delta$. Therefore any hyperplane in $\mathcal{H}_1$ can contain at most $A_{\delta-1} + 1$ points from $B'$. Otherwise, Reduction Rule 6 ($\delta - 1$) can be applied, which is a contradiction.

Thus $\mathcal{H}_1$ covers at most $A_\delta$ points. This contradicts our assumption that $\mathcal{H}_1$ covers all points of $B'$.

Theorem 12. The AxRBSC-hyperplanes problem parameterized by $h$ admits a polynomial kernel.

Proof. Let $(R \cup B, \mathcal{H}, k_i)$ be an instance of AxRBSC-hyperplanes such that none of the reduction rules are applicable. Then, if it is a YES instance, all the points in $B$ can be covered by at most $h$ hyperplanes. Any hyperplane can contain at most $A_{\delta-1} = O(h^{d-1})$ blue points, otherwise Reduction Rule 6 ($\delta - 1$) can be applied. Thus a YES instance can have at most $h^d$ blue points.

Now that we have bounded $b$ as a polynomial function of $h$, the result follows from Theorem 10.

5. Red Blue Set Cover on Skylines

Given $P = R \cup B$ in the plane and $\mathcal{K}$, a set of $m$ bidirectional skylines, we prove that the AxRBSC-skylines problem parameterized by $k$, is W[1]-hard by giving a parameterized reduction from the Square Stabbing problem.

Rectangle Stabbing: Given a set $S$ of $n$ rectangles and a set of axis-parallel lines $L$, the Rectangle Stabbing problem is to decide whether there exists a subset $L' \subseteq L$ with $|L'| \leq k$ such that every rectangle from $S$ is intersected (stabbed) by at least one line in $L'$. If all the rectangles are unit squares, it is the Square Stabbing problem.

Square Stabbing problem parameterized by $k$ is known to be W[1]-hard [7].

Let $(S, L)$ be an instance of the Square Stabbing problem. Let $[l_1, l_2, \ldots, l_n]$ be the set of vertical lines in $L$ which are in non-decreasing order of their $x$-coordinates and let $[h_1, h_2, \ldots, h_m]$ be the set of horizontal lines in $L$ which are in non-increasing order of their $y$-coordinates. For every square $S \in S$, add a blue point $b_S$ to $P$ at the top-left corner of $S$. For every vertical line $l_i$, we add a unit width vertical skyline $S_{l_i}$ with its right edge coinciding with the given line $l_i$. The top edge of the leftmost vertical skyline $S_{l_1}$ is unit distance above the topmost point in $P$. For $i > 1$, the top edge of the skyline $S_{l_i}$ is unit vertical distance away from the top edge of $S_{l_{i-1}}$. Similarly for all horizontal lines, we add a unit width horizontal skyline $S_{h_i}$ whose bottom edge coincides with the line $h_i$. Right edge of the topmost horizontal skyline $S_{h_m}$ is a unit distance away from the rightmost point in $P$. Right edge of $S_{h_i}$ is unit horizontal distance away from the right edge of $S_{h_{i-1}}$, for $i > 1$. For every horizontal skyline, add a red point $r_{h_i}$ to $P$ in the top-right corner of the skyline and for every vertical skyline, add a red point $r_{l_i}$ to $P$ in the top-left corner of the skyline. Thus every skyline has a unique red point. Refer Figure 1.

Figure 1: (a) Instance of Square Stabbing problem, (b) Reduction to an instance of AxRBSC-skylines, (c) Instance of AxRBSC-skylines
Lemma 13. \((S, L)\) can be stabbed using \(k\) lines if and only if \((P, \mathcal{K})\) has a solution of size \(k\).

Proof. If a line \(l \in L\) stabs a square \(S \in \mathcal{S}\), the corresponding skyline \(S_l\) covers the blue point \(b_S\) and the red point \(r_S\). If a line \(l \in L\) stabs a square \(S \in \mathcal{S}\), the corresponding skyline \(S_l\) covers the blue point \(b_S\) and red point \(r_S\). Now \(S\) can be stabbed using \(k\) lines if and only if all blue points can be covered using \(k\) skylines. Since every skyline contains a unique red point, \(S\) can be stabbed using \(k\) lines if and only if \((P, \mathcal{K})\) has a solution of size \(k\). Thus we have proved the following theorem.

Theorem 14. The Red Blue Set Cover problem on bidirectional skylines is \(W[1]\)-hard parameterized by the solution size.

Since a family of bidirectional skylines is a subset of the family of axis parallel rectangles, we have the following result.

Corollary 15. The Red Blue Set Cover problem on axis parallel rectangles is \(W[1]\)-hard parameterized by the solution size \(k\).

6. Quadrants

In this section, we give a polynomial time algorithm to optimally solve the AxRBSC-quadrants problem. Given \(P = R \cup B\) and a family \(Q_i\) of quadrants of the first type with \(|Q_i| = m\), find a subfamily \(Q'_i\) that covers all points from \(B\) and minimum possible number of red points from \(R\).

Observation 16. Let \(q_i\) and \(q_j\) be two quadrants such that \((q_i \cap P) \subseteq (q_j \cap P)\) i.e., \((q_i \cap B) \subseteq (q_j \cap B)\) and \((q_i \cap R) \subseteq (q_j \cap R)\). Then there exist an optimal solution of AxRBSC-quadrants, that does not contain both \(q_i\) and \(q_j\).

Theorem 17. For a given bichromatic point set with \(n\) points, AxRBSC-quadrants can be computed in \(O(mn^3)\) time.

Proof. Let \(O\) be the orthogonal convex hull of \(B\). Consider the left-bottom chain, \(P_L\), of \(O\) i.e., the chain of \(O\) that extends from the blue point with the minimum \(x\)-coordinate to the blue point with the minimum \(y\)-coordinate, in the anti-clockwise direction. Let \(B' = \{b_1, b_2, \ldots, b_l\}\) be the set of blue points on \(P_L\) ordered by increasing \(x\)-coordinate. Note that this is also a non increasing order of \(y\)-coordinate. Let \(R' \subseteq R\) and \(B'' \subseteq B \setminus B'\) respectively be the set of red and blue points that lie on the right side of \(P_L\) (See figure 2). Any set of quadrants that cover all points in \(B''\) will cover all points in \(B''\) and \(R'\). Therefore it is enough to consider the AxRBSC-quadrants problem for \((R = (R' \cup (B = B''), Q_i)\).

For this reduced instance \((R \cup B, Q_i)\), we introduce the following notations. Let \(\pi : R \cup B \to [n]\) be the bijection corresponding to the ordering of the points in \(R \cup B\) by increasing \(x\) coordinates and \(\sigma : R \cup B \to [n]\) be the bijection corresponding to the ordering of the points in \(R \cup B\) by increasing \(y\) coordinates. For a quadrant \(Q \in Q_i\), let \(p \in Q \cap (R \cup B)\) be the point with smallest \(x\) coordinate in \(Q\) and let \(q \in Q \cap (R \cup B)\) be the point with the smallest \(y\) coordinate in \(Q\). Define \(left(Q) = \pi[p]\) and \(bottom(Q) = \sigma(q)\). Let \(B_i = \{b \in B | \sigma(b) > i\}\), \(R_i = \{r \in R | \sigma(r) > i\}\) and \(Q_i = \{Q \in Q_i | left(Q) > j\}\).

Now we give a dynamic programming algorithm to solve the AxRBSC-quadrants problem for this instance.

Let \(dp[i, j]\) return the minimum number of red points from \(R_i\), that is covered by a subset of \(Q_i\) that covers all points in \(B_i\). \(dp[0, 0]\) return the value 0.

![Figure 2: Green line denotes quadrants of the first type and highlighted black orthogonal path indicates left-bottom orthogonal path](image-url)
Here $b'$ is the blue point with the smallest value of $\sigma(b')$ in $B_i$.

To see the correctness of the above, observe that the recurrence considers all $Q \in Q_j$ to cover $b'$. For the correct guess, the rest of the solution only covers uncovered red points from $R_{\text{bottom}}(Q)$ using $Q \in Q_{\text{left}}(Q)$. Otherwise, if some $Q \notin Q_{\text{left}}(Q)$ is part of the solution then it contradicts Observation [16]. Similarly an uncovered red point outside $R_{\text{bottom}}(Q)$ is not part of an optimal solution.

It is clear that computing one entry in the DP table takes $O(m)$ time. The number of $dp[*,*]$ values to be computed is clearly $O(n^2)$. We can construct and verify the orthogonal path in $O(n \log n)$ time.

Therefore, the algorithm runs in $O(mn^2)$ time. $\square$

References

[1] Pradeeasha Ashok, Sudeshna Kolay, and Saket Saurabh. Multivariate complexity analysis of geometric red blue set cover. *Algorithmica*, 79(3):667–697, 2017.

[2] Arindam Biswas, Partha Bhowmick, Moumita Sarkar, and Bhargab B Bhattacharya. A linear-time combinatorial algorithm to find the orthogonal hull of an object on the digital plane. *Information Sciences*, 216:176–195, 2012.

[3] Robert D Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the red-blue set cover problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 345–353, 2000.

[4] Timothy M Chan and Nan Hu. Geometric red–blue set cover for unit squares and related problems. *Computational Geometry*, 48(5):380–385, 2015.

[5] Maw-Shang Chang, Hsiao-Han Chung, and Chuang-Chieh Lin. An improved algorithm for the red–blue hitting set problem with the consecutive ones property. *Information Processing Letters*, 110(20):845–848, 2010.

[6] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.

[7] Michael Dom, Michael R Fellows, and Frances A Rosamond. Parameterized complexity of stabbing rectangles and squares in the plane. In *International Workshop on Algorithms and Computation*, pages 298–309. Springer, 2009.

[8] Michael Dom, Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Red-blue covering problems and the consecutive ones property. *Journal of Discrete Algorithms*, 6(3):393–407, 2008.

[9] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[10] Raghunath Reddy Madireddy and Apurva Mudgal. A constant-factor approximation algorithm for red-blue set cover with unit disks. In *International Workshop on Approximation and Online Algorithms*, pages 204–219. Springer, 2020.

[11] Raghunath Reddy Madireddy, Subhas C Nandy, and Supantha Pandit. On the geometric red-blue set cover problem. In *International Workshop on Algorithms and Computation*, pages 129–141. Springer, 2021.

[12] Pauli Miettinen. On the positive–negative partial set cover problem. *Information Processing Letters*, 108(4):219–221, 2008.

[13] Rolf Niedermeier and Peter Rossmanith. An efficient fixed-parameter algorithm for 3-hitting set. *Journal of Discrete Algorithms*, 1(1):89–102, 2003.

[14] David Peleg. Approximation algorithms for the label-covermax and red-blue set cover problems. *Journal of Discrete Algorithms*, 5(1):55–64, 2007.