Balanced Covering Problem in Bicolored point Sets

Sujoy Bhore[1], **Supantha Pandit**[2], **and Sasanka Roy**[3]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
`sujoy.bhore@gmail.com`
2 **Stony Brook University, Stony Brook, NY, USA**
`pantha.pandit@gmail.com`
3 **Indian Statistical Institute, Kolkata, India**
`sasanka.ro@gmail.com`

Abstract

We study a variation of the classical set cover problem called the *balanced covering (BC)* problem on a set of red and blue points in the Euclidean plane. Let P be a set of red and blue points in the plane. An object is called a *balanced* object with respect to P, if it contains an equal number of red and blue points from P. In the BC problem, the objective is to cover the points in P with a minimum number of homogeneous geometric objects (i.e., unit squares, intervals) such that each object is *balanced*. For points in the plane, we prove that the BC problem is NP-hard when the covering objects are unit squares. For points on a line, we show that if the ratio of the total numbers of reds and blues is more than 2 then, there exists no solution of the BC problem. Subsequently, we devise a linear time exact algorithm for the BC problem with intervals. Finally, we study the study the problem of computing a balanced object of maximum cardinality. For this, we give polynomial time algorithms with unit squares in the plane and intervals on a line.

1 Introduction

Set cover is a well-studied problem in computer science with numerous application in various fields. In this problem a set P of points and a set O of objects are given and the objective is to cover all the points in P with a minimum number of objects in O. We consider a variation of this problem on a bicolored (red and blue) point sets. Let $P = R \cup B$ be a set of bicolored points in the plane, where R denotes a set of red and B denotes a set of blue points. We say that a geometric object X is *balanced* if it covers an equal number of red and blue points. Here we consider two problems based on the coverage of the points in P.

Balanced Covering (BC) Problem
Given a set $P = R \cup B$ of bicolored points in the plane, the objective is to find a minimum collection set of balanced objects that covers P.

Maximum Balanced Object (MBO) Problem
Given a set $P = R \cup B$ of bicolored points in the plane, the objective is to find a balanced object that covers the maximum number of points in P.

A related problem to the BC problem is the class cover problem: given a set of red and a set of blue points and a set of objects, the goal is to cover all the blue points excluding the red points with minimum number of objects [2, 1]. Another related problem is the red-blue set cover problem [3]. Generalized versions of these problems are studied in [7]. Chan and Hu [4] considered this problem when the covering objects are unit squares, and proved the

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

NP-hardness and gave a PTAS. We would like to mention that, in the discrete setting, the *BC* problem is equivalent to the standard geometric set cover problem and therefore becomes NP-hard [6]. However, in the continuous setting, one would require to decide the particular placement of the objects due to the color constraint.
Our Results: In Section 2, for points in the plane, we prove that the *BC* problem is NP-hard when the covering objects are unit squares. In Section 3, for points on a line, we show that if the ratio of the total numbers of reds and blues is more than 2 then, there does not exist a solution of the *BC* problem. Subsequently, we devise a linear time exact algorithm for the *BC* problem with intervals. Finally, we study the problem of computing a single balanced object of maximum cardinality. For this, we give polynomial time algorithms with unit squares (in section 3) in the plane and intervals on a line.

2 NP-Hardness: *BC* problem with Unit Squares

We prove that the *BC* problem with unit squares is NP-hard by a reduction from the rectilinear planar monotone 3SAT (shortly RPM3SAT) problem that is known to be NP-complete [5]. We define this problem as follows. A clause is said to be a positive (resp. negative) clause if all the literals it contains are positive (resp. negative). We are given a 3SAT instance ϕ with n variables and m clauses either positive or negative. The variables are positioned on a horizontal line. The positive clauses are above this line, and they connect to its corresponding variables with three legs: (i) ┌, (ii) |, and (iii) ┐. The negative clauses are below the line, and they connect to its corresponding variables with three legs: (iv) └, (v) |, and (vi) ┘. Finally, these legs do not intersect each other. The objective is to find a satisfying assignment for ϕ. See Figure 1 for an instance of the RPM3SAT problem.

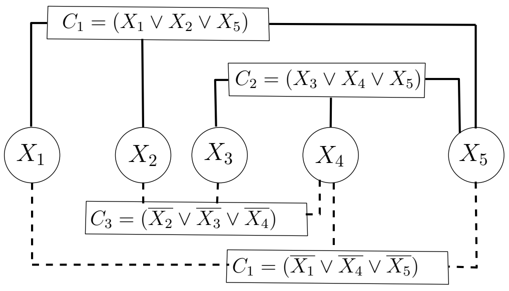
$C_1 = (X_1 \vee X_2 \vee X_5)$
$C_2 = (X_3 \vee X_4 \vee X_5)$
X_1
X_2
X_3
X_4
X_5
$C_3 = (\overline{X_2} \vee \overline{X_3} \vee \overline{X_4})$
$C_1 = (\overline{X_1} \vee \overline{X_4} \vee \overline{X_5})$

Figure 1 An instance of the RPM3SAT problem.

We construct an instance I_ϕ of the *BC* problem from an instance ϕ of the RPM3SAT problem. Let $\{u, v\}$ be a pair of two differently colored points that are ϵ ($0 < \epsilon \ll 1$) distance apart. We call such a pair a *site*. We assume that the given points are a collection of sites.
Variable Gadget: The gadget for the variable x_i has two parts: a cycle and a set of chains that are attached to the cycle. Let d be the maximum number of clauses in which x_i is present, i.e, the number of legs attached to x_i. Then, x_i contains d chains one for each leg. Consider two imaginary axis-parallel horizontal lines ℓ_1 and ℓ_2 that are suitably placed such that any two sites placed on ℓ_1 and ℓ_2 cannot be covered by a unit square (see Figure 2). We place $6m + 2$ sites $s_1, s_2, \ldots, s_{6m+2}$ on ℓ_1 such that the distance between any two consecutive sites is exactly 1. In a similar fashion, we place $6m + 2$ sites $t_1, t_2, \ldots, t_{6m+2}$ sites on ℓ_2 as well (see Figure 2). We use 6 additional sites and place them in the following way; 3 sites each to the left and right of the above arrangement (see Figure 2). Thus as a total of $12m + 10$ sites form a cycle like structure. Notice that, due to this construction,

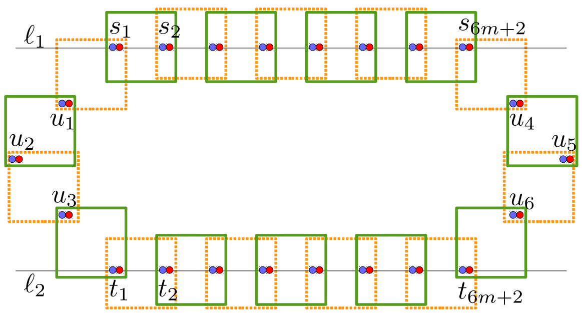

Figure 2 The cycle gadget corresponding to a variable x_i.

in the continuous balanced covering any unit square can cover at most two sites that are consecutive along the cycle. In Figure 2, we demonstrate a set of possible canonical unit squares that cover the sites. Note that in an optimal balanced covering exactly half of the squares are selected: either all solid or all dotted. There are three types of *chains* that are attached to the cycle of a variable gadget. Each chain is a specific geometric embedding of a set of sites. The gadgets of types (i) and (ii) chains are shown in Figure 3(a) and Figure 3(b) respectively. The other types ((iii)-(vi)) of chains are constructed by a simple modification of types (i) and (ii) chains.

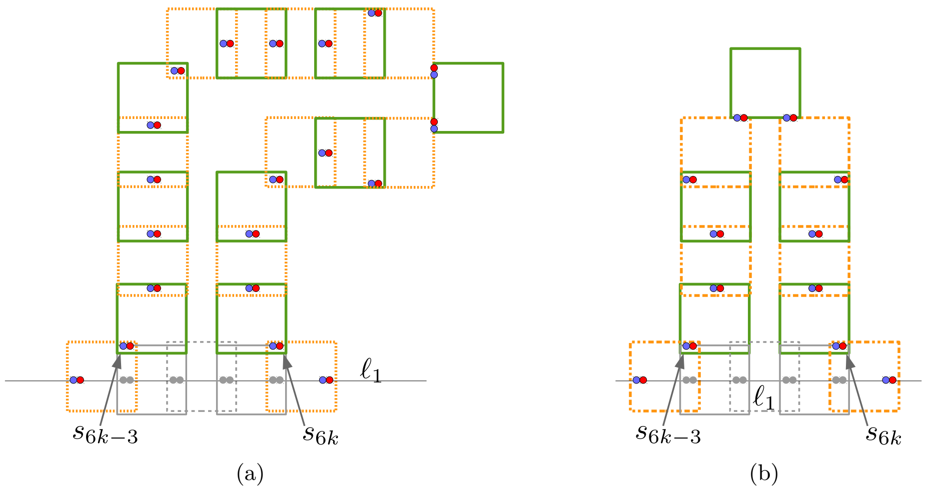

Figure 3 The chains of a variable (a)Type 1 (b)Type 2.

It needs to be mentioned that the number of sites is not fixed for every chain, even for similar chains of different clauses. Now we explain how the chains are attached to the variable cycle. Let C_1, C_2, ... be the order of the positive clauses that connect to a variable. We associate the four site s_{6k-3}, s_{6k-2}, s_{6k-1}, and s_{6k} in the cycle with the k-th clause in this order. We remove the two sites s_{6k-2} and s_{6k-1} from the cycle and perturb the other two sites s_{6k-3} and s_{6k} slight vertically up. See Figure 3 for detailed explanation.

▶ Observation 1. Let S_{x_i} be a set of unit squares associated with x_i. Exactly $\delta_i = |S_{x_i}|/2$ squares (either all solid or all dotted) are required for an optimal balanced-covering of x_i.

Clause: We describe a clause gadget and how it interacts with variable gadgets. Assume, w.l.o.g., that $C_i = (x_i, y_i, z_i)$ is a positive clause. For C_i, we take a special site w_i called the

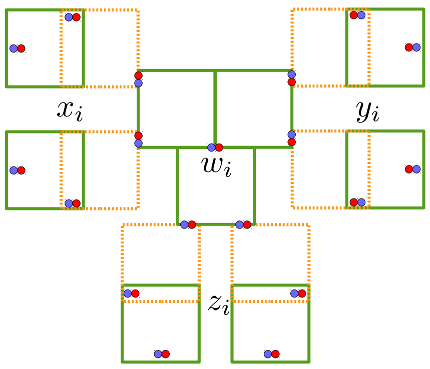

Figure 4 Positive clause interaction with the three variables it contains.

clause-site that connects the chains corresponding to the variables. The placement of w_i with respect to the three chains is shown in Figure 4. Notice that, no two sites from two different chains are covered by a single square. Clearly, I_ϕ can be constructed in polynomial time.

▶ **Lemma 2.1.** *ϕ is satisfiable if and only if I_ϕ has a balance cover with $\delta = \sum_{i=1}^{n} \delta_i$ squares.*

Thereby, we conclude the following theorem.

▶ **Theorem 2.2.** *The BC problem is* NP*-hard.*

3 Points on a Line

Let P be a set of of m red and n blue points on a line. We show that there is no solution for the BC problem when $\frac{m}{n} > 2$. Note, however it is not guaranteed that there is always a solution for BC problem if $\frac{m}{n} \leq 2$.

▶ **Theorem 3.1.** *Given a set P of m red points and n blue points on a real line $\mathcal{L}$, if $\frac{m}{n} > 2$ then, there does not exist a solution for the BC problem.*

Proof. For the sake of contradiction, let us assume that there is an optimal solution, $OPT = \{I_1, \ldots, I_j\}$ of the BC problem that covers P.

▶ Claim 1. Every blue point $b_i \in P$ is contained in at most two intervals in OPT.

Proof. For the sake of contradiction, let b_i be contained in at least three intervals (say I_i, I_j, I_k) in OPT. We select the two intervals from I_i, I_j, I_k; one whose left end point is left most and the other whose right end point is right most. Clearly, removing I_i, I_j, I_k from OPT and adding these two intervals in OPT still covers all the points, a contradiction. ◀

For each interval $I_i \in OPT$, we define a red-blue pairing in the following manner. Let $S(I_i) \subseteq P$ be the subset of points contained in I_i. Let $\{r_1, \ldots, r_p\}$ and $\{b_1, \ldots, b_p\}$ be the red and blue points in sorted order (x-coordinate wise) in I_i. Now, we consider the red-blue pairs $\{\{r_1, b_1\}, \ldots \{r_p, b_p\}\}$. For an interval I_k, consider a pair $\{r_i, b_i\}$ (for some i), we say b_i *balances* r_i. Now any blue point b_i that is contained in an interval I_k can *balance* exactly one red point in I_k. Using Claim 1, we conclude that each blue point b_i can *balance* at most two different red points. Hence, the ratio between reds and blues is at most 2. ◀

3.1 Exact Algorithms for Intervals

We give exact algorithms for *BC* and *MBO* problems while the covering objects are intervals. We denote them as *BCI* and *MBI* problem, respectively. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of red and blue points on a line given in sorted order. The idea is to obtains a set of *candidate* intervals, and then choose intervals from this set with some modification. Observe that, in any optimal solution, a chosen balanced interval is not contained in any other chosen interval.
Finding candidate intervals: We maintain a counter c (initially it is empty) and an array $A[n]$. For each point $p_i \in P$ in the order, if p_i is red we increase the value of c by 1, otherwise decrease the value by 1, and set $A[i] = c$. Note that the values in A are integers and in the range $[-n, n]$. Let l and h be the minimum and maximum values of the counter c, respectively. Note that l and h can be negative. We construct a table T of $(|l| + |h| + 1)$ rows and 3 columns as follows. The first column stores the values in the range $[l, h]$ in order. The values of the second and third columns are initially empty. We update some of the entries during the following procedure. We go through each entry of A one by one. For the ith entry i.e., $A[i]$, if $T[A[i]][2]$ is empty then $T[A[i]][2] = i$. Else $T[A[i]][3] = i$. Now for each row i we generate an interval if $T[i][3]$ is non-empty. Therefore, we generates at most $n/2$ intervals based on the indices of A between the l and h.

▶ Observation 2. For each candidate interval the difference in reds and blues is at most 1.

Now we make each candidate interval (say I_c) that is balanced in the following way. Let (i, j) be the index of I_c. We consider the 9 intervals $\{i-1, i, i+1\} \times \{j-1, j, j+1\}$ and choose the maximum balance one. We make it for all candidate intervals. This process gives us a set of candidate balance intervals. Now to find the maximum balance interval just return the candidate balance interval that contains maximum number of points. For covering the whole point set P, we run the standard greedy algorithm on the candidate balance intervals and return the minimum cardinality subset of intervals that covers P.

▶ **Theorem 3.2.** *Given a set P of red and blue points in sorted order on the line,*

1. *finding the largest balanced interval requires $O(n)$ time and*
2. *finding minimum number of balanced intervals that cover P also requires $O(n)$ time.*

4 Exact Algorithm for Unit Squares

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of red and blue points on a line given in sorted order. We study the *MBO* problem with unit square (shortly, *MBS*). Consider a unit square s, it is called a 2-anchored square if there is at least two distinct points on any of its two consecutive sides. Notice that, the output of the *MBS* problem is a 2-anchored square, otherwise we can always translate it to make such one without loosing any point (see Figure 5(a)). The naive algorithm for the *MBS* problem is following. First, we build a range tree T of P. Now, for each 2-anchored square we check in T, the points that is contained in that square, and report the maximum balanced square. This process takes $O(n^2 \log n)$ time. We give a simple $O(n^2)$ time algorithm for the *MBS* problem.

For each point $p_i \in P$, let $S(p_i)$ be the square centered at p_i. If p_i is a blue point (resp. red point) then, $S(p_i)$ is a blue square (resp. red square). Let $\mathcal{S}$ be the set of red and blue squares based on the points of P. Let E be the set that contains the left and right endpoints of the squares, and $|E| = 2n$. We consider the points in E in sorted order from left-to-right based on their x-coordinates.

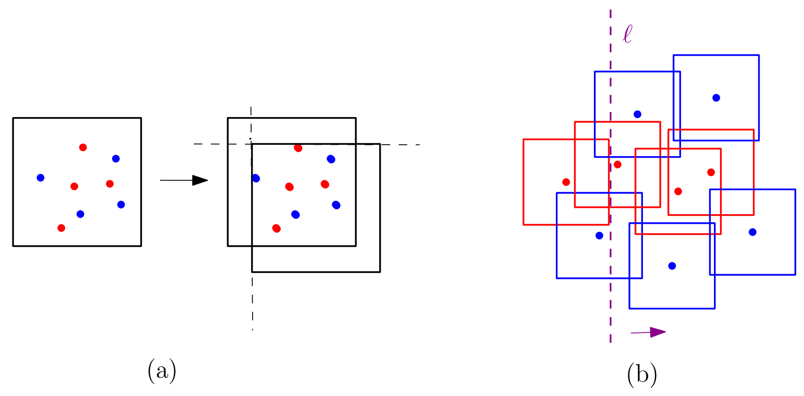

Figure 5 (a) Translation into a 2-anchored square. (b) A vertical line intersecting a set of bicolored squares.

▶ Observation 3. Consider any two points $p_i, p_j \in P$ such that $S(p_i)$ and $S(p_j)$ intersect. For an arbitrary point p_k that is in the intersecting region of $S(p_i)$ and $S(p_j)$, the square $S(p_k)$ contains both p_i and p_j.

We use the following procedure to find a maximum balanced square. Let ℓ be a vertical sweep line that considers the squares from left-to-right. Whenever, ℓ reaches to a point in E, we call it an event point and indeed we have $2n$ event points. For each $i \in [2n]$, the line ℓ passes through a point $p_i \in E$, and let $\mathcal{S}' \subseteq \mathcal{S}$ be a subset of squares intersecting ℓ (see Figure 5(b)). Beside that we maintain two counters (r, b) for each square. Whenever a square s enters or leaves ℓ we update in the range tree the (r, b) values of all the nodes whose corresponding squares intersecting s. Note that, at each event point, we basically have a set of red and blue unit intervals on ℓ. In linear time, we can compute the largest balanced subset clique. We know that, their corresponding squares have a common intersecting region. It is possible to place p_k in that region such that $S(p_k)$ is a balanced square (from observation 3). This process takes $O(n^2)$ time.

▶ **Theorem 4.1.** *Let P be a set of red and blue points on a line given in sorted order, there is an algorithm that computes maximum balanced unit square in $O(n^2)$ time.*

References

1 Sergey Bereg, Sergio Cabello, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, and Inmaculada Ventura. The class cover problem with boxes. *Computational Geometry*, 45(7):294 – 304, 2012.

2 Adam Cannon and Lenore Cowen. Approximation algorithms for the class cover problem. *Ann. Math. Artif. Intell.*, 40(3-4):215–224, 2004.

3 Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav Marathe. On the red-blue set cover problem. In *SODA*, 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 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

6 Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.

7 Apurva Mudgal and Supantha Pandit. Generalized class cover problem with axis-parallel strips. In *Workshop on Algorithms and Computation, WALCOM*, pages 8–21, 2014.