Geometric Red-Blue Set Cover for Unit Squares and Related Problems

Timothy M. Chan* Nan Hu*

Abstract

We study a geometric version of the Red-Blue Set Cover problem originally proposed by Carr, Doddi, Konjevod, and Marathe (SODA 2000): given a red point set, a blue point set, and a set of objects, we want to use objects to cover all the blue points, while minimizing the number of red points covered. We prove that the problem is NP-hard even when the objects are unit squares in 2D, and we give the first PTAS for this case. The technique we use simplifies and unifies previous PTASes for the weighted geometric set cover problem and the unique maximum coverage problem for 2D unit squares.

1 Introduction

Given a red set R and a blue set B of total size m, and a family $\mathcal{S}$ of n subsets of $R \cup B$, the Red-Blue Set Cover problem is to find a subfamily of $\mathcal{S}$ which covers all the elements in B, but covers the minimum number of elements in R.

The problem was first introduced by Carr, Doddi, Konjevod and Marathe [1], who proved that even in the restricted case where every set in $\mathcal{S}$ contains only one blue and two red elements, the problem cannot be approximated to within $2^{\log^{1-\delta} n}$ factor for $\delta = 1/\log^c \log n$ and for any constant $c < 1/2$, unless P = NP. Carr et al. also gave a $2\sqrt{n}$-approximation algorithm for the case where every set in $\mathcal{S}$ contains only one blue element.

We study a geometric version of Red-Blue Set Cover where the elements of R and B are points, and the sets of $\mathcal{S}$ are geometric objects. Specifically, we focus on the case where the objects are unit squares[1] in 2D; we call the resulting problem Red-Blue Unit-Square Cover. We prove that Red-Blue Unit-Square Cover remains NP-hard, and we present a PTAS (i.e., a polynomial-time $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$) for this problem.

Previous work on PTASes. There have already been a number of PTASes for problems related to geometric set cover and hitting set in the literature. To put our new PTAS into context, we note that most of the known techniques can be classified into a few categories:

1. Hochbaum and Maass' original *shifted grid* technique [8]. This is among the earliest PTAS techniques developed, and is usually applicable only to "continuous" versions of geometric set cover and hitting set problems. For example, in the continuous version of the standard (monochromatic) Unit-Square Cover problem, we want the smallest number of unit squares to cover a given point set, where the allowed unit squares can be located anywhere rather than from a given ("discrete") set. When applicable, the technique is general enough to handle other types of similar-sized fat objects, such as unit disks in 2D, or unit balls in higher fixed dimensions. Extensions of the technique based on *shifted quadtrees* have also been explored for some related problems [2].

2. Mustafa and Ray's *local search* technique [10]. This yields the first PTAS for the general discrete version of the Unit-Square Cover and the analogous Unit-Disk Cover problem in 2D. However, the technique inherently does not work for weighted problems. For example, in the Weighted Unit-Square Cover problem, given a point set and a set of unit squares each with a positive weight, we want a subset of unit squares of the smallest total weight to cover the given point set.

3. Sophisticated *dynamic programming* combined with Hochbaum and Maass' shifting technique. Erlebach and van Leeuwen [5] used this approach to obtain the first PTAS for Weighted Unit-Square Cover. Recently, Ito et al. [9] have also applied a similar approach to obtain a PTAS for the following variant of unit-square cover called Unique Unit-Square Coverage: given a point set and a set of unit squares, we want a subset of unit squares to maximize the number of points that are covered exactly once. (Erlebach and van Leeuwen introduced the general unique coverage problem for sets in 2008 [4].) At the moment, these PTASes are limited to the special case of 2D unit squares and do not seem generalizable to unit disks or to higher dimensions.

Our PTAS belongs to the third category and is similar to Erlebach and van Leeuwen's and Ito et al.'s PTASes. For example, our approach can easily handle a weighted version of Red-Blue Unit-Square Cover, where the

*Cheriton School of Computer Science, University of Waterloo, {tmchan,n3hu}@uwaterloo.ca

[1] All squares in this paper are assumed to be axis-aligned.

red points have weights and we want to minimize the total weight of the red points covered. However, our approach works only for unit squares and not for other types of objects.

Arguably the most interesting aspect of this paper lies not so much in the specific result about red-blue set cover, but in our technique, which we feel is conceptually simpler than Erlebach and van Leeuwen's and Ito et al.'s PTASes [5, 9] for Weighted Unit-Square Cover and Unique Unit-Square Coverage. In fact, our technique leads to alternative PTASes for these two problems as well, and can potentially be more easily applied to other variants of set cover problems for unit squares.

The descriptions of the dynamic programming algorithm in both papers [5, 9] are lengthy. For example, the algorithm by Erlebach and van Leeuwen is obtained by simulating a plane sweep that involves multiple sweep lines moving at different speeds. We get around most of the complications by one very simple idea: a "mod-one trick".

In section 2, we give the NP-hardness proof of Red-Blue Unit-Square Cover. We introduce the mod-one trick and give a PTAS in section 3, followed by a brief discussion on how to apply our technique to other problems in section 4.

2 NP-Hardness

Theorem 1 Red-Blue Unit-Square Cover *is NP-hard.*

Proof. We reduce from the vertex cover problem on degree-3 planar graphs, which is well known to be NP-hard [7].

Lemma 2 [3] *Every planar graph $G = (V, E)$ of maximum degree at most 4 has an orthogonal planar drawing on an $O(|V|) \times O(|V|)$ grid (i.e., vertices are placed at grid points and edges are drawn as a rectilinear polygonal chain with corners at grid points, with no crossings).*

Lemma 3 (Folklore) *Given a graph G and an edge e in G, define a new graph G' obtained from G by subdividing e through the addition of two new "dummy" vertices. Then the size of a minimum vertex cover of G' is precisely the size of a minimum vertex cover of G plus 1.*

Given a degree-3 planar graph G with n vertices, we create an orthogonal drawing by Lemma 2. We define a new graph G' by subdividing each edge e through the addition of new dummy vertices at each grid point along e. Each edge in G' is now a horizontal or vertical line segment of length 1 in the drawing. If e contains an odd number of dummy vertices, we insert an extra new dummy vertex at the midpoint of a line segment. Then all edge lengths in G' are 1/2 or 1. By rescaling by a factor slightly less than 2, we can ensure that all edge lengths in G' are strictly between 2/3 and 2. Now, each edge in the original graph G has an even number of dummy vertices, and by repeated applications of Lemma 3, finding the size of the minimum vertex cover of G is equivalent to finding the size of the minimum vertex cover of G'.

To construct an instance of Red-Blue Unit-Square Cover from G', we replace each vertex in G' by a red point r_i. For each edge $r_i r_j$ in G', we create a blue point b_{ij} in the middle of the edge and add a unit square containing precisely b_{ij} and r_i and a unit square containing precisely b_{ij} and r_j. See Figure 1. Such squares exist since the distance between two adjacent blue and red points is strictly between 1/3 and 1.

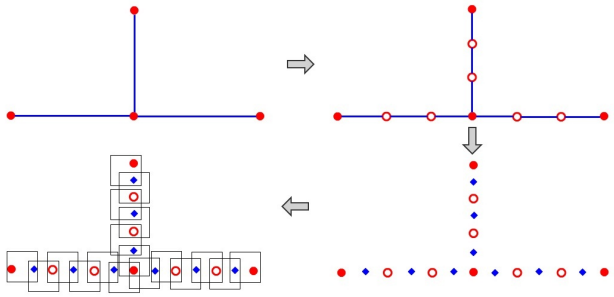

Figure 1: The reduction from vertex cover. (Red points are drawn as dots, and blue points are drawn as diamonds.)

Correctness of the reduction is easy to see: Given a vertex cover of G' of size k, we can select all the squares that cover the corresponding k red points; these squares would cover all blue points. Conversely, given a subset of squares covering all blue points, the red points covered by these squares form a vertex cover of G'. □

3 PTAS

We now present a PTAS for Red-Blue Unit-Square Cover. We begin with a definition:

Definition 4 *Let $S = \{s_1, \ldots, s_t\}$ be a set of unit squares, where $s_1, \ldots, s_t$ are arranged in increasing x-order of their centers. We say that S forms a monotone set, if the centers of $s_1, \ldots, s_t$ are in increasing or decreasing y-order.*

Note that the boundary of the union of the squares in a monotone set S consists of two monotone chains ("staircases"), as as shown in figure 2. We say that these two chains are *complementary.*

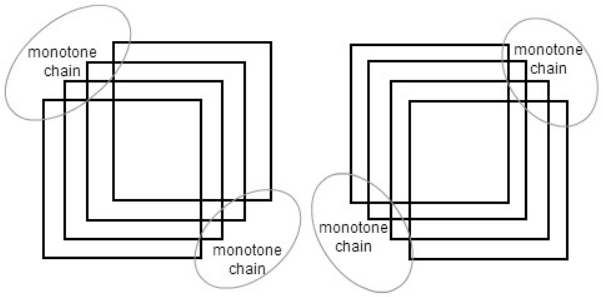

Figure 2: A monotone set may be "increasing" (left) or "decreasing" (right).

Lemma 5 *Let* OPT *be an optimal solution for an instance where the blue point set B is inside a $k \times k$ square. Then* OPT *can be decomposed into $O(k^2)$ monotone sets.*

Proof. We may assume that all squares in OPT intersect the $k \times k$ square. Draw a grid with unit side length over the $k \times k$ square. Consider a grid point p. Let $S(p)$ be the set of squares in OPT containing p; every square in OPT belongs to one of the $S(p)$'s. Let $U(p)$ denote the boundary of the union of the squares of $S(p)$. We may assume that each square in $S(p)$ appears on $U(p)$, for otherwise we could remove the square from OPT and the resulting solution is no worse than OPT.

Divide the plane into 4 quadrants at p. For each $i \in \{1, 2, 3, 4\}$, let $S_i(p)$ be the subset of squares in OPT containing p that contributes to the portion of $U(p)$ inside the i-th quadrant. Then each $S_i(p)$ is a monotone set. Thus, we have decomposed OPT into $4(k+1)^2$ monotone sets. (These sets may not be disjoint, but can be made disjoint by deleting elements from sets, since a subset of a monotone set is still monotone.) □

The heart of our PTAS is an exact dynamic programming solution for the special case of the problem where all points are inside a $k \times k$ grid for a constant k. The idea is to use a sweep-line algorithm to guess the $O(k^2)$ monotone sets at the same time. We can "remember" a constant number ($O(k^2)$) of intersections of the monotone chains with a vertical sweep line as we sweep from left to right. However, each monotone set defines two complementary monotone chains, and the guess of one chain should be consistent with the guess of its complementary chain; but by the time the sweep line gets to the second chain, we would have forgotten information about the first chain. This is why Erlebach and van Leeuwen [5] needed a more complicated approach involving multiple sweep lines moving at different speeds.

To avoid this difficulty, we overlay all the monotone sets into one grid cell by introducing a "mod-one" transformation:

Definition 6 *We define the* mod-one *mapping* $(x, y) \mapsto (x \bmod 1, y \bmod 1)$, *where* $z \bmod 1$ *denotes the fractional part of a real number* z.

With this transformation, a unit square is rearranged into four pieces covering the unit grid cell, as shown in figure 3.

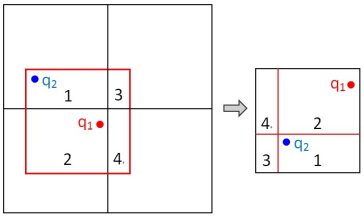

Figure 3: Applying the mod-one transformation to a unit square.

Furthermore, the union of the squares in a monotone set is rearranged as shown in figure 4. Notice that the two complementary monotone chains are mapped to two monotone chains that are connected at the corner points. This is the key property we need about the mod-one transformation. By redesigning the sweep-line algorithm to sweep over the unit grid cell in the transformed space, we can guess the two complementary monotone chains of each monotone set at the same time. The remaining pieces of the union consists of two rectangles defined by the start and end square of the monotone set; we can guess these two squares in advance.

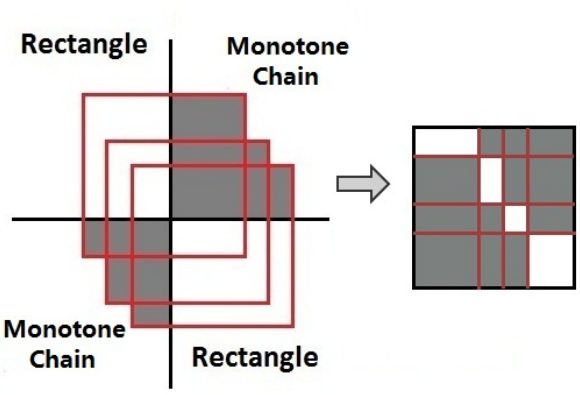

Figure 4: Applying the mod-one transformation to a monotone set.

Theorem 7 *For any instance of* Red-Blue Unit-Square Cover *where B is inside a $k \times k$ square for a constant k, we can find the optimal solution in $O(mn^{O(k^2)})$ time.*

Proof. We find it best to describe our dynamic programming algorithm in terms of a state-transition diagram. We define a *state* to consist of

- a vertical sweep line ℓ that passes through a corner of an input square, after taking mod 1;
- $O(k^2)$ 4-tuples of the form $(s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}})$, subject to the conditions that $s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}}$ are in increasing x-order and form a monotone set, and that ℓ lies between the corners of s_{prev} and s_{curr}, mod 1.

Intuitively, a state represents current information about a decomposition of a solution into monotone sets at the sweep line (the monotone sets are not required to be disjoint). Specifically, each 4-tuple corresponds to a monotone set S; s_{start} and s_{end} represent the start and end square of S; and s_{prev} and s_{curr} represent the squares that define intersections of the sweep line with the two complementary monotone chains of S, after taking mod 1. These two squares s_{prev} and s_{curr} are adjacent in the monotone set S.

Given this state, we create a *transition* into a new state as follows: We pick the 4-tuple $(s_{\text{start}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{end}})$ such that the corner point of s_{curr} has the smallest x-coordinate, mod 1. The new sweep line ℓ' will be at the corner of s_{curr}. This 4-tuple is replaced by a new 4-tuple $(s_{\text{start}}, s_{\text{curr}}, s', s_{\text{end}})$ satisfying the stated conditions for some square s'. All other 4-tuples are unchanged. Let j_r (resp. j_b) be the number of red (resp. blue) points that lie between ℓ and ℓ', after taking mod 1, and are covered (resp. not covered) by the squares in the $O(k^2)$ 4-tuples (before taking mod 1). If $j_b > 0$, we remove this transition. Otherwise, we set the cost of this transition to j_r.

The problem is thus reduced to finding the shortest path in this state-transition diagram (a directed acyclic graph), after adding suitable transitions involving start and end states. There are at most $O(mn^{O(k^2)})$ states, and each state has at most $O(n)$ outgoing transitions (since there are $O(n)$ choices for s'). Thus, we can construct the graph and find the shortest path by dynamic programming in $O(mn^{O(k^2)})$ time. □

We can now apply Hochbaum and Maass' grid shifting technique [8] to obtain our final result:

Theorem 8 *There is a PTAS for* Red-Blue Unit-Square Cover.

Proof. For each shift $a, b \in \{0, \ldots, k-1\}$, let $S^{a,b}$ be the union of the solutions found by Theorem 7 for the blue points inside every $k \times k$ square $[ik+a, (i+1)k+a] \times [jk+b, (j+1)k+b]$, with $i, j \in \mathbb{Z}$. We return the $S^{(a,b)}$ with the smallest $c(S^{(a,b)})$, where $c(S)$ denotes the number of red points covered by S.

To analyze the approximation factor, let OPT be the optimal solution. Let $\text{OPT}^{a,*}$ (resp. $\text{OPT}^{*,b}$) be the subset of squares in OPT intersecting the lines $x = ik+a$ with $i \in \mathbb{Z}$ (resp. the lines $y = jk+b$ with $j \in \mathbb{Z}$). Since the algorithm in Theorem 2 covers the minimum number of red points for the subproblem for each $k \times k$ square, we have

$$c(S^{a,b}) \;\le\; c(\text{OPT}) + 2\,c(\text{OPT}^{a,*}) + 2\,c(\text{OPT}^{*,b}).$$

Since $\sum_{0 \le a < k} c(\text{OPT}^{a,*})$ and $\sum_{0 \le b < k} c(\text{OPT}^{*,b})$ are both at most $2\,c(\text{OPT})$,

$$\sum_{0 \le a,b < k} c(S^{a,b}) \;\le\; (k^2 + 8k)\,c(\text{OPT}),$$

implying that

$$\min_{0 \le a,b < k} c(S^{a,b}) \;\le\; (1 + 8/k)\,c(\text{OPT}).$$

Setting $k = \lceil 8/\varepsilon \rceil$ gives a $(1+\varepsilon)$-approximation algorithm. □

4 Related Problems

Weighted Unit-Square Cover. Erlebach and van Leeuwen [5] studied the following related problem: Given a set P of points and a set $\mathcal{S}$ of unit squares in 2D where each square has a positive weight, we want to find a smallest-weight subset of $\mathcal{S}$ to cover all the points in P.

Our algorithm can easily be modified to solve this problem. Specifically, in the proof of Theorem 7, if there are any points that lie between ℓ and ℓ', after taking mod 1, and are not covered by any of the squares in the $O(k^2)$ 4-tuples, then we remove the transition. Otherwise, we set the cost of the transition to the weight of the square s'.

Budgeted Maximum Coverage for Unit Squares. Erlebach and van Leeuwen [5] also considered the following problem: Given a set P of points where each point has a positive profit value, and given a set $\mathcal{S}$ of unit squares where each square has a positive cost, and given a budget B, we want to find a subset of $\mathcal{S}$ with total cost at most B, maximizing the total profit of all points in P that are covered by the subset.

Erlebach and van Leeuwen [5] described how a modification of their dynamic programming algorithm combined with additional ideas can yield a PTAS for this problem. Our approach can be used to simplify the dynamic programming part of their PTAS.

Partial Unit-Square Cover. Gandhi et al. [6] studied the *partial set cover* problem. A geometric version can be stated as follows: Given a set P of points and a set $\mathcal{S}$

of unit squares in 2D, and given an integer K, we want to find a smallest subset of squares in $\mathcal{S}$ to cover at least K points in P.

Gandhi et al. gave a PTAS for a continuous version of the problem based on Hochbaum and Maass' shifted grid technique [8]. For the discrete version, we can obtain a PTAS by using an appropriate modification of our dynamic programming algorithm, in conjunction with shifted grids as in Gandhi et al.'s paper.

Unique Unit-Square Coverage. Ito et al. [9] studied the following problem: Given a set P of points and a set $\mathcal{S}$ of unit squares in 2D, find a subset of $\mathcal{S}$ to maximize the number of points in P that are covered exactly once by the subset.

Again our algorithm can be modified to solve this problem. In the proof of Theorem 7, we use 6-tuples $(s_{\text{start}}, s_{\text{prev2}}, s_{\text{prev}}, s_{\text{curr}}, s_{\text{curr2}}, s_{\text{end}})$ instead of 4-tuples, where intuitively s_{prev2} represents the predecessor of s_{prev} and s_{curr2} represents the successor of s_{curr} in a monotone set. We set the cost of the transition to be the number of points that lie between ℓ and ℓ', after taking mod 1, and are uniquely covered by the squares in the $O(k^2)$ 6-tuples. This works because squares that are not part of these 6-tuples are irrelevant as to whether a point is uniquely covered.

5 Conclusion

We have shown that Red-Blue Unit-Square Cover is NP-hard, and have given a PTAS using a "mod-one" transformation. The main advantage of our PTAS is that it is simpler to describe than previous PTASes by Erlebach and van Leeuwen and Ito et al. for related problems [5, 9]. To be fair, we should mention that our $n^{O(1/\varepsilon^2)}$ running time is slower than the $n^{O(1/\varepsilon)}$ running time of the previous PTASes.

References

[1] R. D. Carr, S. Doddi, G. Konjevod, and M. Marathe. On the red-blue set cover problem. In *Proc. SODA*, pages 345–353, 2000.

[2] T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46:178–189, 2003.

[3] B. N. Clark, C. J. Colbourn and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86:165–177, 1990.

[4] T. Erlebach and E. J. van Leeuwen. Approximating geometric coverage problems. In *Proc. SODA*, pages 1267–1276, 2008.

[5] T. Erlebach and E. J. van Leeuwen. PTAS for weighted set cover on unit squares. In *Proc. APPROX and RANDOM*, pages 166–177, 2010.

[6] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53:55-84, 2004.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[8] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32:130–136, 1985.

[9] T. Ito, S.-I. Nakano, Y. Okamoto, Y. Otachi, R. Uehara, T. Uno, and Y. Uno. A polynomial-time approximation scheme for the geometric unique coverage problem on unit squares. In *Proc. SWAT*, pages 24–35, 2012.

[10] N. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete Comput. Geom.*, 44:883–895, 2010.