A PTAS for the Weighted Unit Disk Cover Problem ⋆

Jian Li Yifei Jin

IIIS, Tsinghua University, China
{lijian83@mail, jin-yf13@mails}.tsinghua.edu.cn

Abstract. We are given a set of weighted unit disks and a set of points in Euclidean plane. The minimum weight unit disk cover (WUDC) problem asks for a subset of disks of minimum total weight that covers all given points. WUDC is one of the geometric set cover problems, which have been studied extensively for the past two decades (for many different geometric range spaces, such as (unit) disks, halfspaces, rectangles, triangles). It is known that the unweighted WUDC problem is NP-hard and admits a polynomial-time approximation scheme (PTAS). For the weighted WUDC problem, several constant approximations have been developed. However, whether the problem admits a PTAS has been an open question. In this paper, we answer this question affirmatively by presenting the first PTAS for WUDC. Our result implies the first PTAS for the minimum weight dominating set problem in unit disk graphs. Combining with existing ideas, our result can also be used to obtain the first PTAS for the maxmimum lifetime coverage problem and an improved constant approximation ratio for the connected dominating set problem in unit disk graphs.

1 Introduction

The set cover (SC) problem is a central problem in theoretical computer science and combinatorial optimziation. In the problem, we are given a ground set U and collection $\mathcal{S}$ of subsets of U. Each set $S \in \mathcal{S}$ has a non-nagative weight w_S. The goal is to find a subcollection $\mathcal{C} \subseteq \mathcal{S}$ of minimum total weight such that $\bigcup \mathcal{C}$ covers all elements of U. The approximibility of the general SC problem is rather well understood: it is well known that the greedy algorithm is an H_n-approximation ($H_n = \sum_{i=1}^{n} 1/i$) and obtaining a $(1-\epsilon)\ln n$-approximation for any constant $\epsilon > 0$ is NP-hard [12, 19]. In the *geometric set cover problem*, U is a set of points in some Euclidean space $\mathbb{R}^d$, and $\mathcal{S}$ consists of geometric objects (e.g., disks, squares, triangles), In such geometric setting, we can hope for better-than-logarithmic approximations due to the special structure of $\mathcal{S}$. Most geometric set cover problems are NP-hard, even for the very simple classes of objects such as unit disks [8, 24] (see [6, 22] for more examples and exceptions). Approximation algorithms for geometric set cover have been studied extensively for the past two decades, not only because of the importance of the problem per se, but also its rich connections to other important notions and problems, such as

⋆ Research supported in part by the National Basic Research Program of China Grant 2015CB358700, 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61202009, 61033001, 61361136003.

VC-dimension [4, 9, 18], ϵ-net, union complexity [7, 32, 33], planar separators [20, 29], even machine scheduling problems [3].

In this work, we study the geometric set cover problem with one of the simplest class of objects, unit disks. The formal definition of our problem is as follows:

Definition 1. *Weighted Unit Disk Cover (WUDC): Given a set $\mathcal{D} = \{\mathsf{D}_1, \ldots, \mathsf{D}_n\}$ of n unit disks and a set $\mathcal{P} = \{P_1, \ldots, P_m\}$ of m points in Euclidean plane $\mathbb{R}^2$. Each disk D_i has a weight $w(\mathsf{D}_i)$. Our goal is to choose a subset of disks to cover all points in $\mathcal{P}$, and the total weight of the chosen disks is minimized.*

We note that WUDC is the general version of minimum weight dominating set problem in unit disk graphs (UDG). In fact, several previous results on WUDC were stated in the context of the dominating set problem.

1.1 Previous Results and Our Contribution

We first recall that a polynomial time approximation scheme (PTAS) for a minimization problem is an algorithm $\mathcal{A}$ that takes an input instance, a constant $\epsilon > 0$, returns a solution SOL such that $\mathsf{SOL} \leq (1+\epsilon)\mathsf{OPT}$, where OPT is the optimal value, and the running time of $\mathcal{A}$ is polynomially in the size of the input for any fixed constant ϵ.

WUDC is NP-hard, even for the unweighted version (i.e., $w(\mathsf{D}_i) = 1$) [8]. For unweighted dominating set in unit disk graphs, Hunt et al. [26] obtained the first PTAS in unit disk graphs. For the more general disk graphs, based on the connection between geometric set cover problem and ϵ-nets, developed in [4, 9, 18], and the existence of ϵ-net of size $O(1/\epsilon)$ for halfspaces in $\mathbb{R}^3$ [30] (see also [21]), it is possible to achieve a constant factor approximation. As estimated in [29], these constants are at best 20 (A recent result [5] shows that the constant is at most 13). Moreover, there exists a PTAS for unweighted disk cover and minimum dominating set via the local search technique [20, 29].

For the general weighted WUDC problem, the story is longer. Ambühl et al. [2] obtained the first approximation for WUDC with a concrete constant 72, without using the ϵ-net machinery. Applying the shifting techique of [23], Huang et al. [25] obtained a $(6+\epsilon)$-approximation algorithm for WUDC. The approximation factor was later improved to $(5+\epsilon)$ [10], and to $(4+\epsilon)$ by several groups [11,16,34]. The current best ratio is 3.63. [1] Besides, the *quasi-uniform sampling method* [7,33] provides another approach to achieve a constant factor approximation for WUDC (even in disk graphs). However, the constant depends on several other constants from rounding LPs and the size of *the union complexity*. Very recently, based on the separator framework of Adamaszek and Wiese [1], Mustafa and Raman [28] obtained a QPTAS (Quasi-polynomial time approximation scheme) for weighted disks in $\mathbb{R}^2$ (in fact, weighted halfspaces in $\mathbb{R}^3$), thus ruling out the APX-hardness of WUDC.

Another closely related work is by Erlebach and van Leeuwen [17], who obtained a PTAS for set cover on weighted unit squares, which is the first PTAS for weighted geometric set cover on any planar objects (except those poly-time solvable cases [6, 22]). Although it may seem that their result is quite close to a PTAS for weighted

[1] The algorithm can be found in Du and Wan [14], who attributed the result to a manuscript by Willson et al.

WUDC, as admitted in their paper, their technique is insufficient for handling unit disks and "completely different insight is required".

In light of all the aforementioned results, it seems that we should expect a PTAS for WUDC, but it remains to be an open question (explicitly mentioned as an open problem in a number of previous papers, e.g., [2,14–17,31]). Our main contribution in this paper is to settle this question affirmatively by presenting the first PTAS for WUDC.

Theorem 1. *There is a polynomial time approximation scheme for the WUDC problem. The running time is $n^{O(1/\epsilon^9)}$.*

Due to the equivalence between WUDC and minimum weight dominating set in unit disk graphs, we immediately have the following corollary.

Corollary 1. *There is a polynomial time approximation scheme for the minimum weight dominating set problem in unit disk graphs.*

We note that the running time $n^{\mathrm{poly}(1/\epsilon)}$ is nearly optimal in light of the negative result by Marx [27], who showed that an EPTAS (i.e., Efficient PTAS, with running time $f(1/\epsilon)\mathrm{poly}(n)$) even for the unweighted dominating set in UDG would contradict the exponential time hypothesis.

Finally, we show that our PTAS for WUDC can be used to obtain improved approximation algorithms for two important problems in wireless sensor networks, the connected dominating set problem and the maximum lifetime coverage problem in UDG.

2 Our Approach - A High Level Overview

By the standard shifting technique [13], it suffices to provide a PTAS for WUDC when all disks are located in a square of constant size (we call it a block, and the constant depends on $1/\epsilon$). This idea is formalized in Huang et al. [25], as follows.

Lemma 1 (Huang ta al. [25]). *Suppose there exists a ρ-approximation for WUDC in a fixed $L \times L$ block, with running time $f(L)$. Then there exists a $(\rho + O(1/L))$-approximation with running time $O(L \cdot n \cdot f(L))$ for WUDC. In particular, setting $L = 1/\epsilon$, there exists a $(\rho+\epsilon)$-approximation for WUDC, with running time $O\left(\frac{1}{\epsilon} \cdot L \cdot f(\frac{1}{\epsilon})\right)$.*

In fact, almost all previous constant factor approximation algorithms for WUDC were obtained by developing constant approximations for a single block of a constant size (which is the main difficulty). The main contribution of the paper is to improve on the previous work [2, 10, 16, 25] for a single block, as in the following lemma.

Lemma 2. *There exists a PTAS for WUDC in a fixed block of size $L \times L$ for $L = 1/\epsilon$. The running time of the PTAS is $n^{O(1/\epsilon^9)}$*

From now on, the approximation error guarantee $\epsilon > 0$ is a fixed constant. Whenever we say a quantity is a constant, the constant may depend on ϵ. We use OPT to represent the optimal solution (and the optimal value) in this block. We use capital letters $A, B, C, \ldots$ to denote points, and small letters $a, b, c, \ldots$ to denote arcs. For two

points A and B, we use $|AB|$ to denote the line segment connecting A and B (and its length). We use D_i to denote a disk and D_i to denote its center. For a point A and a real $r > 0$, let $\mathsf{D}(A, r)$ be the disk centered at A with radius r. For a disk D_i, we use $\partial\mathsf{D}_i$ to denote its boundary. We call a segment of $\partial\mathsf{D}_i$ *an arc*.

First, we guess that whether OPT contains more than C disks or not for some constant C. If OPT contains no more than C disks, we enumerate all possible combinations and choose the one which covers all points and has the minimum weight. This takes $O\left(\sum_{i=1}^{C}\binom{n}{i}\right) = O(n^C)$ time, which is polynomial.

The more challenging case is whether OPT contains more than C disks. In this case, we guess (i.e., enumerate all possibilities) the set $\mathcal{G}$ of the C most expensive disks in OPT. There are at most a polynomial number (i.e., $O(n^C)$) possible guesses. Suppose our guess is correct. Then, we delete all disks in $\mathcal{G}$ and all points that are covered by $\mathcal{G}$. Let D_t (with weight w_t) be the cheapest disk in $\mathcal{G}$. We can see that $\mathsf{OPT} \geq Cw_t$. Moreover, we can also safely ignore all disks with weight larger than w_t (assuming that our guess is correct). Now, our task is to cover the remaining points with the remaining disks, each having weight at most w_t. We use $\mathcal{D}' = \mathcal{D} \setminus \mathcal{G}$ and $\mathcal{P}' = \mathcal{P} \setminus \mathcal{P}(\mathcal{G})$ to denote the set of the remaining disks and the set of remaining points respectively, where $\mathcal{P}(\mathcal{G})$ denote the set of points covered by some disk in $\mathcal{G}$.

Next, we carefully choose to include in our solution a set $\mathcal{H} \subseteq \mathcal{D}'$ of at most ϵC disks. The purpose of $\mathcal{H}$ is to break the whole instance into many (still a constant) small pieces (substructures), such that each substructure can be solved optimally, via dynamic programming. [2] One difficulty is that the substructures are not independent and may interact with each other (i.e., a disk may appear in more than one substructure). In order to apply the dynamic programming technique to all substructures simultaneously, we have to ensure the orders of the disks in different substructures are consistent with each other. Choosing $\mathcal{H}$ to ensure a globally consistent order of disks is in fact the main technical challenge of the paper.

Suppose we have a set $\mathcal{H}$ which suits our need (i.e., the remaining instance $(\mathcal{D}' \setminus \mathcal{H}, \mathcal{P}' \setminus \mathcal{P}(\mathcal{H}))$ can be solved optimally in polynomial time by dynamic programming). Let $\mathcal{S}$ be the optimal solution of the remaining instance. Our final solution is $\mathsf{SOL} = \mathcal{G} \cup \mathcal{H} \cup \mathcal{S}$. First, we can see that $w(\mathcal{S}) \leq w(\mathsf{OPT} - \mathcal{G} - \mathcal{H}) \leq \mathsf{OPT} - w(\mathcal{G})$, since $\mathsf{OPT} - \mathcal{G} - \mathcal{H}$ is a feasible solution for the instance $(\mathcal{D}' \setminus \mathcal{H}, \mathcal{P}' \setminus \mathcal{P}(\mathcal{H}))$. Hence, we have that $\mathsf{SOL} = w(\mathcal{G}) + w(\mathcal{H}) + w(\mathcal{S}) \leq \mathsf{OPT} + \epsilon C w_t \leq (1+\epsilon)\mathsf{OPT}$, where the 2nd to last inequality holds because $|\mathcal{H}| \leq \epsilon C$, and the last inequality uses the fact that $\mathsf{OPT} \geq w(\mathcal{G}) \geq Cw_t$.

Constructing $\mathcal{H}$: Now, we provide a high level sketch for how to construct $\mathcal{H} \subseteq \mathcal{D}'$. First, we partition the block into *small squares* of side length $\mu = O(\epsilon)$ such that any disk centered in a square can cover the whole square and the disks in the same square are close enough. Let the set of small squares be $\Xi = \{\Gamma_{ij}\}_{1 \leq i,j \leq K}$ where $K = L/\mu$. For a small square Γ, let $\mathsf{D}_{s_\Gamma} \in \Gamma$ and $\mathsf{D}_{t_\Gamma} \in \Gamma$ be the furthest pair of disks (i.e., $|D_{s_\Gamma} D_{t_\Gamma}|$ is maximized). We include the pair D_{s_Γ} and D_{t_Γ} in $\mathcal{H}$, for every small square $\Gamma \in \Xi$, and call the pair *the square gadget* for Γ. We only need to focus on covering the remaining points in the *uncovered region* $\mathbb{U}(\mathcal{H})$.

[2] An individual substructure can be solved using a dynamic program similar to [2, 22].

We consider all disks in a small square Γ. The uncovered portion of those disks defined two disjoint connected regions We call such a region, together with all relevant arcs, a *substructure* (formal definition in Section 4). In fact, we can solve the disk covering problem for a single substructure optimally using dynamic programming (which is similar to the dynamic program in [2, 22]). It appears that we are almost done, since ("intuitively") all square gadgets have already covered much area of the entire block, and we should be able to use similar dynamic program to handle all such substructures as well. However, the situation is more complicated (than we initially expected) since the arcs are dependent. See Figure 1 for a "not-so-complicated" example. Firstly, there may exist two arcs (*sibling arcs*) which belong to the same disk when the disk is centered in the *core-center area*). The dynamic program has to make decisions for two sibling arcs, which belong to two different substructures (called R-correlated substructures), together. Second, in order to carry out dynamic program, we need a suitable order of all arcs. To ensure such an order exists, we need all substructures interact with each other "nicely".

In particular, besides all square gadgets, we need to add into $\mathcal{H}$ a constant number of extra disks. This is done by a series of "cut" operations. A cut can either break a cycle, or break one substructure into two substructures. To capture how substructures interact, we define an auxiliary graph, call substructure relation graph $\mathfrak{S}$, in which each substructure is a node. The aforementioned R-correlations define a set of blue edges, and geometrically overlapping relation define a set of red edges. Though the cut operations, we can make blue edges form a matching, and red edges also form a matching, and $\mathfrak{S}$ acyclic (we call $\mathfrak{S}$ an acyclic 2-matching). The special structure of $\mathfrak{S}$ allows us to define an ordering of all arcs easily. Together with some other simple properties, we can generalize the dynamic program for one substructure to all substructures.

3 Square Gadgets

We discuss the structure of a square gadget $\mathsf{Gg}(\Gamma)$ associated with the small square Γ. Recall that the square gadget $\mathsf{Gg}(\Gamma) = \mathsf{D}_s \cup \mathsf{D}_t$, where D_s and D_t are the furthest pair of disks in Γ. We can see that for any disk D_i in Γ, there are either one or two arcs of $\partial\mathsf{D}_i$ which are not covered by $\mathsf{Gg}(\Gamma)$. Without loss of generality, assume that D_sD_t is horizontal. The line D_sD_t divides the whole plane into two half-planes which are denoted by H^+ (the upper half-plane) and H^- (the lower half-plane). $\partial\mathsf{D}_s$ and $\partial\mathsf{D}_t$ intersect at two points P and Q. We need a few definitions which are useful throughout the paper.

1. (Center Area and Core-center Area) Define the *center area* of $\mathsf{Gg}(\Gamma)$ as the intersection of the two disks $\mathsf{D}(D_s, r_{st})$ and $\mathsf{D}(D_t, r_{st})$ in the square Γ, where $r_{st} = |D_sD_t|$. We use $\mathfrak{C}$ to denote it. Since D_s and D_t are the furthest pair, we can see that every other disk in Γ is centered in the center area $\mathfrak{C}$.
   We define the *core-center area* of $\mathsf{Gg}(\Gamma)$ is the intersection of two unit disks centered at P, Q respectively. Essentially, any unit disk centered in the core-center area has four intersections with the boundary of gadget. Let us denote the area by $\mathfrak{C}_o$.
2. (Active Region) Consider the regions $\left(\bigcup_{D_i\in\mathfrak{C}_o}\mathsf{D}_i - (\mathsf{D}_s\cup\mathsf{D}_t)\right)\cap H^+$ and $\left(\bigcup_{D_i\in\mathfrak{C}_o}\mathsf{D}_i - (\mathsf{D}_s\cup\mathsf{D}_t)\right)\cap H^-$. We call each of them an *active region* associated

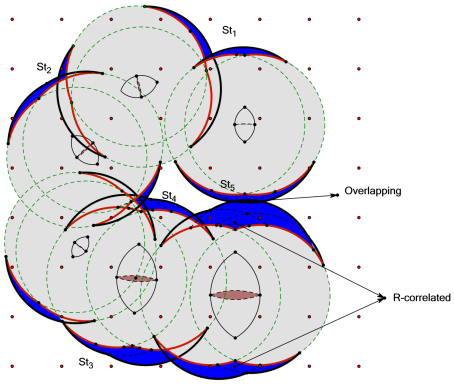

Fig. 1. The general picture of the substructures in a block. The red points are the grid points of squares. Dash green disks are what we have selected in $\mathcal{H}$. There are five substructures in the block.

with square Γ. An active region can be covered by disks in the core-center area. We use Ar to denote an active region.

4 Substructures

Initially, $\mathcal{H}$ includes all square gadgets. In Section 7, we will include in $\mathcal{H}$ a constant number of extra disks. For a set S of disk, we use $\mathbb{R}(S)$ to denote the region covered by disks in S (i.e., $\cup_{\mathsf{D}_i \in S} \mathsf{D}_i$). Assuming a fixed $\mathcal{H}$, we now describe the basic structure of the uncovered region $\mathbb{R}(\mathcal{D}') - \mathbb{R}(\mathcal{H})$. [3] For ease of notation, we use $\mathbb{U}(\mathcal{H})$ to denote the uncovered region $\mathbb{R}(\mathcal{D}') - \mathbb{R}(\mathcal{H})$. Figure 1 shows an example. Intuitively, the region consists of several "strips" along the boundary of $\mathcal{H}$. Now, we define some notions to describe the structure of those strips.

1. (Baseline) We use $\partial\mathcal{H}$ to denote to be the boundary of $\mathcal{H}$. Consider an arc a whose endpoints P_1, P_2 are on $\partial\mathcal{H}$. We say the arc a cover a point $P \in \partial\mathcal{H}$, if P lies in the segment between P_1 and P_2 along $\partial\mathcal{H}$. We say a point $P \in \partial\mathcal{H}$ *can be covered* if some arc in $\mathcal{D}'$ covers P. A baseline is a consecutive maximal segment of $\partial\mathcal{H}$ that can be covered. We usually use b to denote a baseline.
2. (Substructure) A substructure $\mathsf{St}(\mathsf{b}, \mathcal{A})$ consists of a baseline b and the collection $\mathcal{A}$ of arcs which can cover some point in b. The two endpoints of each arc $a \in \mathcal{A}$ are on b and $\angle(a)$ is less than π. Note that every point of b is covered by some arc in $\mathcal{A}$. Figure 2 illustrates the components of an substructure.

Arc Order: Now we switch our attention to the order of the arcs in a substructure $\mathsf{St}(\mathsf{b}, \mathcal{A})$. Suppose the baseline b starts at point Q_s and ends up at point Q_t. Consider any two points P_1 and P_2 on the baseline b. If P_1 is more close to Q_s than P_2 along the

[3] Recall that $\mathcal{D}' = \mathcal{D} \setminus \mathcal{G}$ where $\mathcal{G}$ is the C most expensive disks in OPT.

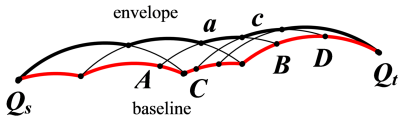

Fig. 2. A substructure. The baseline b consists of the red arcs which are the part of consecutive boundary of $\partial\mathcal{H}$. Q_s, Q_t are the endpoints of b. The black arcs are in the uncovered region. The arc $a \prec c$ since $A \prec C$ and $B \prec D$. The bold black arcs form the envelope.

baseline b, we say that P_1 *appears earlier* than P_2 (denoted as $P_1 \prec P_2$). Consider any two arcs a and c in $\mathcal{A}$. The endpoints of arc a are A and B and the endpoints of arc c are C and D. All of points A, B, C, D are on the baseline b. Without loss any generality, we assume that $A \prec B$, $C \prec D$ and $A \prec C$. If $B \prec D$, We say arc a *appears earlier* than arc c (denoted as $a \prec c$). Otherwise, we say a and c are incomparable. See Figure 2 for an example. It is easy to see that $\prec$ defines a partial order.

Adjacency: Consider two arcs a (with endpoints $A \prec B$) and c (with endpoints $C \prec D$). If $a \prec c$ and $C \prec B$, we say that a and b are *adjacent* (we can see they must intersect exactly once), and c is the *adjacent successor* of a. Similarly, we can define the adjacent successor of subarc $a[P_1, P_2]$. If c is the adjacent successor of a, meanwhile c intersects with subarc $a[P_1, P_2]$, we say that c is the *adjacent successor* of subarc $a[P_1, P_2]$. Among all adjacent successors of $a[P_1, P_2]$, we call the one whose intersection with $a[P_1, P_2]$ is closest to P_1 the *first adjacent successor* of $a[P_1, P_2]$.

5 Simplifying the Problem

The substructures may overlap in a variety of ways. As we mentioned in Section 2, we need to include in $\mathcal{H}$ more disks in order to make the substructures amenable to the dynamic programming technique. However, this step is somewhat involved and we decide to postpone it to the end of the paper (Section 7). Instead, we present in this section what is the organization of the substructures *after* including more disks in $\mathcal{H}$ and what properties we need for the final dynamic program.

Self-Intersections: In a substructure St, suppose there are two arcs a and c in $\mathcal{A}$ with endpoints A, B and C, D respectively. If $A \prec B \prec C \prec D$ and a and c cover at least one and the same point in $\mathcal{P}$, we say the substructure is *self-intersecting*. So we will eliminate all self-intersections in Section 7. In the rest of the section, we assume all substructures are *non-self-intersecting* and discuss their properties.

Order Consistency: There are two types of relations between substructures which affect how the orientations should be done. One is the overlapping relation and the other is *Remote-Correlation.* See Figure 1 for some examples.

Definition 2 (Remotely correlation). *Consider two substructures* St_u *and* St_l *which are not overlapping. They contain different related active regions of the same gadget. We say that they are* remotely correlated *or* R-correlated.

There are two possible baseline orientations for each substructure (clockwise or anticlockwise around the center of the arc), which gives rise to four possible ways to orient both St_u and St_l. However, there are only two (out of four) of them are consistent (thus we can do dynamic programming on them). More formally, we need the following definition:

As different substructures may interact with each other, we need a dynamic program which can run over all substructures simultaneously. Hence, we need to define a globally consistent ordering of all arcs.

Definition 3 (Global Order Consistency). *We have global order consistency if there is a way to orient the baseline of each substructure, such that the partial orders of the disks for all substructures are consistent in the following sense: It can not happen that $a_i \prec b_i$ in substructure $\mathsf{St}_i(\mathsf{b}_i, \mathcal{A}_i)$ but $a_j \prec b_j$ in $\mathsf{St}_j(\mathsf{b}_j, \mathcal{A}_j)$, where $a_i, a_j \in \partial\mathsf{D}_a, b_i, b_j \in \partial\mathsf{D}_b$ and $a_i, b_i \in \mathcal{A}_i, a_j, b_j \in \mathcal{A}_j$.*

Substructure Relation Graph $\mathfrak{S}$: we construct an auxiliary graph $\mathfrak{S}$, called the *substructure relation graph*, to capture all R-correlations and Overlapping relations. Each node in $\mathfrak{S}$ represents a substructure. If two substructures are R-correlated, we add a blue edge between the two substructures. If two substructure overlap, we add a red edge.

Definition 4 (Acyclic 2-Matching). *We say the substructure relation graph $\mathfrak{S}$ is an acyclic 2-matching, if $\mathfrak{S}$ is acyclic and is composed by a blue matching and a red matching. In other words, $\mathfrak{S}$ contains only paths,and the red edges and blue edges appear alternately in each path.*

Definition 5 (Point Order Consistency). *Suppose a set $\mathcal{P}_{\text{co}}$ of points is covered by both of two overlapping substructures $\mathsf{St}_1(\mathsf{b}_1, \mathcal{A}_1)$ and $\mathsf{St}_2(\mathsf{b}_2, \mathcal{A}_2)$. Consider any two points $P_1, P_2 \in \mathcal{P}_{\text{co}}$ and four arcs $a_1, a_2 \in \mathcal{A}_1, b_1, b_2 \in \mathcal{A}_2$. Suppose $P_1 \in \mathbb{R}(a_1) \cap \mathbb{R}(b_1)$ and $P_2 \in \mathbb{R}(a_2) \cap \mathbb{R}(b_2)$. But $P_1 \notin \mathbb{R}(a_2) \cup \mathbb{R}(b_2)$ and $P_2 \notin \mathbb{R}(a_1) \cup \mathbb{R}(b_1)$. We say P_1 and P_2 are point-order consistent if $a_1 \prec a_2$ in St_1 and $b_1 \prec b_2$ in St_2. We say the points in $\mathcal{P}_{\text{co}}$ satisfy point order consistency if all pair of points in $\mathcal{P}_{\text{co}}$ are point-order consistent.*

All introducing all relevant concepts, we can finally state the set of properties we need for the dynamic program.

Lemma 3. *After choosing $\mathcal{H}$, we can ensure the following properties holds:*

P1. (Active Region Uniqueness) Each substructure contains at most one active region.
P2. (Non-self-intersection) Every substructure is non-self-intersecting.
P3. (Acyclic 2-Matching) The substructure relation graph $\mathfrak{S}$ is an acyclic 2-matching, i.e., $\mathfrak{S}$ consists of only paths. In each path, red edges and blue edges appear alternately.
P4. (Point Order Consistency) Any point is covered by at most two substructures. The points satisfy the point order consistency.

How to ensure all these properties will be discussed in details in Section 7. Now, everything is in place to describe the dynamic program.

6 Dynamic Programming

Suppose we have already constructed the set $\mathcal{H}$ such that Lemma 3 holds (along with an orientation for each substructure). Without loss any generality, we can assume that the remaining disks can cover all remaining points (otherwise, either the original instance is infeasible or our guess is wrong). In fact, our dynamic program is inspired, and somewhat similar to those in [2, 16, 22].

We can see that we only need to handle each path in $\mathfrak{S}$ separately (since different paths have no interaction at all). Hence, from now on, we simply assume that $\mathfrak{S}$ is a path. Suppose the substructures are $\{\mathsf{St}_k(\mathsf{b}_k, \mathcal{A}_k)\}_{k\in[m]}$. We use A_k and B_k to denote two endpoints of b_k. Generalizing the previous section, a state for the general DP is $\Phi = \{P_k\}_{k\in[m]}$, where P_k is an intersection point in substructure St_k. Let b_{Pk} and t_{Pk} be the two arcs intersecting at P_k. Suppose $b_{Pk} \prec t_{Pk}$. We call arc b_{Pk} *base-arc* and t_{Pk} *top-arc* for point $\P_k$. Denote the endpoints of b_{Pk} by C_k, C'_k and the endpoints of t_{Pk} by D_k, D'_k. Suppose $b_{Pk}(P, C'_k]$ intersects its first successor at P^b (called *base-adjacent point*) and $t_P(P, D'_k]$ intersects its first successor at P^t (called *top-adjacent point*). For each $k \in [m]$, we define $\mathsf{St}_k^{[P_k]}(\mathsf{b}_k[P_k], \mathcal{A}_k[P_k])$ as follows.

- $\mathsf{b}_k[P_k]$ is the concatenation of subarc $b_P[P, C'_k]$ and the original baseline segment $\mathsf{b}_1[C'_k, P_t]$. All arcs in $\mathsf{b}_1^{[P]}$ have cost zero.
- $\mathcal{A}_k[P_k]$ consists of all arcs $a' \in \mathcal{A}_k$ such that $b_{P_k} \prec a'$ (of course, with the portion covered by $\mathsf{b}_k[P]$ subtracted). The cost each such arc is the same as its original cost.

We use $\mathcal{P}(a)$ (or $\mathcal{P}(\mathcal{A})$) to denote the points can be covered by arc a (or arc set $\mathcal{A}$). Let $\mathcal{P}\left[\{P_k\}_{k\in[m]}\right]$ be the point set we need to cover in the subproblem: $\mathcal{P}\left[\{P_k\}_{k\in[m]}\right] = \bigcup_{k\in[m]} \mathcal{P}(\mathcal{A}_k[P_k]) - \bigcup_{k\in[m]} \mathcal{P}(b_{Pk})$. The subproblem $\mathsf{OPT}(\{P_k\}_{k\in[m]})$ is to find, for each substructure St_k, a valid path from P_k to B_k, such that all points in $\mathcal{P}[\{P_k\}_{k\in[m]}]$ can be covered and the total cost is minimized.

The additional challenge for the general case is caused by R-correlations. If two arcs (in two different substructures) belong to the same disk, we say that they are *siblings* of each other. If we processed each substructure independently, some disks would be counted twice. In order to avoid double-counting, we should consider both siblings together, i.e., select them together and pay the disk only once in the DP.

In order to implement the above idea, we need a few more notations. We construct an auxiliary bipartite graph $\mathfrak{B}$. The nodes on one side are all disks in $\mathcal{D}' \setminus \mathcal{H}$, and the node on the other side are substructures. If disk D_i has an arc in the substructure St_j, we add an edge between D_i and St_j. Besides, for each arc of baselines, we add a node to represent it and add an edge between the node and the substructure which contains the arc. Because the weight of any arc of baselines is zero, it shall not induce contradiction that regard them as independent arcs. In fact, there is a 1-1 mapping between the edges in $\mathfrak{B}$ and all arcs.

Fix a state $\Phi = \{P_k\}_{k\in[m]}$. For any arc a in St_k (with intersection point P_k and base-arc b_{Pk}), a has three possible positions: (1) $a \prec b_{Pk}$: we label its corresponding edge with "unprocessed"; (2) $a = b_{Pk}$: we label its corresponding edge with "processing"; (3) Others: we label its corresponding edge with "done". As mentioned before, we

need to avoid the situation where one arc becomes the base-arc first (i.e., being added in solution and paid once), and its sibling becomes the base-arc in a later step (hence being paid twice). With the above labeling, we can see that all we need to do is to avoid the states in which one arc is "processing" and its sibling is "unprocessed". If disk D is incident on at least one "processing" edge and not incident on any "unprocessed" edge, we say the D is *ready*. Let $\mathcal{R}$ be the set of ready disks. For each ready disk D, we use $N_{\mathrm{p}}(\mathsf{D})$ to denote the set of neighbors (i.e., substructures) of D connected by "processing" edges. We should consider all substructures in $N_{\mathrm{p}}(\mathsf{D})$ together.

We need in our DP indicator variables to tell us whether a certain transition is feasible: Formally, if $\mathcal{P}[\{P_k\}_{k\in[m]}] = \mathcal{P}[[\mathcal{P}_k][P_i^b]_{\{i\}}]$, let $I_i = 0$. Otherwise, let $I_i = 1$. For ease of notation, for a set $\{e_k\}_{k\in[m]}$ and $S \subseteq [m]$, we write $[e_k][e'_i]_S = \{e_k\}_{k\in[m]\setminus S} \cup \{e'_i\}_{i\in S}$. Hence, $[P_k][P_i^b]_{\{i\}} = \{P_k\}_{k\in[m]\setminus i} \cup P_i^b$ and $[P_k][P_i^t]_{N_{\mathrm{p}}(\mathsf{D})} = \{P_k\}_{k\in[m]\setminus N_{\mathrm{p}}(\mathsf{D})} \cup \{P_i^t\}_{i\in N_{\mathrm{p}}(\mathsf{D})}$. Then we have the dynamic program as follows:

$$\mathsf{OPT}\left(\{P_k\}_{k\in[m]}\right) = \min\begin{cases} \min_{i\in[m]}\left\{\mathsf{OPT}\left([P_k][P_i^b]_{\{i\}}\right) + I_i\cdot\infty\right\}, & \text{add no disk} \\ \min_{\mathsf{D}\in\mathcal{R}}\left\{\mathsf{OPT}\left([P_k][P_i^t]_{N_{\mathrm{p}}(\mathsf{D})}\right) + w_{\mathsf{D}}\right\}, & \text{add disk } \mathsf{D} \end{cases}$$

Note that in the second line, the arc(s) in $N_{\mathrm{p}}(\mathsf{D})$ are base-arcs (w.r.t. state $\Phi(\{P_k\}_{k\in[m]})$.

7 Constructing $\mathcal{H}$

In this section, we describe how to construct the set $\mathcal{H}$ in details. We first include in $\mathcal{H}$ all square gadgets. The boundary of $\mathcal{H}$ consists of several closed curves, as shown in Figure 1. $\mathcal{H}$ and all arcs in the uncovered region $\mathbb{U}(\mathcal{H})$ define a set of substructures.

First, we note that there may exist a closed curve that all points on the curve are covered by some arcs (or informally, we have a cyclic substructure, with the baseline being a cycle). We need to break all such baseline cycles by including a constant number of extra arcs into $\mathcal{H}$. This is easy after we introduce the label-cut operation, and we will spell out all details then. Note that we cannot choose some arbitrary envelope cycle since it may ruin some good properties we want to maintain.

From now on, we assume that all baselines are simple paths. Now, each closed curve contains one or more baselines. So, we have an initial set of well defined substructures. The main purpose of this section is to cut these initial substructures such that Lemma 3 holds.

We will execute a series of operations for constructing $\mathcal{H}$. We first provide below an high level sketch of our algorithm, and outline how the substructures and the substructure relation graph $\mathfrak{S}$ evolve along with the operations.

- First, we deal with active regions. Sometimes, two active region may overlap significantly and become inseparable (formally defined later), they essentially need to be dealt as a single active region. In this case, we merge the two active regions together (we do not need to do anything, but just to pretend that there is only one active region). We can also show that one active region can be merged with at most one other active region. For the rest of cases, two overlapping active region are separable, and we can cut them into at most two non-overlapping active regions, by

adding a small number of extra disks in $\mathcal{H}$. After the merging and cutting operations, each substructure contains at most one active region. Hence, the substructures satisfy the property (P1) in Lemma 3. Moreover, we show that if any substructure contains an active region, the substructure is limited in a small region.
- We ensure that each substructure is non-self-intersecting, using a simple greedy algorithm. After this step, (P2) is satisfied.
- In this step, we ensure that substructure relation graph $\mathfrak{S}$ is an acyclic 2-matching (P3). The step has three stages. First, we prove that the set of blue edges forms a matching. Second, we give an algorithm for cutting the substructures which overlap with two or more other substructures. After the cut, each substructure overlaps with no more than one other substructure. So after the first two stages, we can see that $\mathfrak{S}$ is composed of a blue matching and a red matching. At last, we prove that the blue edges and red edges cannot form a cycle, establishing $\mathfrak{S}$ is acyclic.
- The goal of this step is to ensure the point-order consistency (P4). We first show there does not exist a point covered by more than two substructures, when $\mathfrak{S}$ is an acyclic 2-matching. Hence, we only need to handle the case of two overlapping substructures. We show it is enough to break all cycles in a certain planar directed graph. Again, we can add a few more disks to cut all such cycles.
- Lastly, we show that the number of disks added in $\mathcal{H}$ in the above four steps is $O(K^2)$.

References

1. ADAMASZEK, A., AND WIESE, A. Approximation schemes for maximum weight independent set of rectangles. In *FOCS* (2013), IEEE, pp. 400–409.
2. AMBÜHL, C., ERLEBACH, T., MIHALÁK, M., AND NUNKESSER, M. Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs. In *APPROX-RANDOM*. Springer Berlin Heidelberg, 2006.
3. BANSAL, N., AND PRUHS, K. The geometry of scheduling. *SICOMP 43*, 5 (2014), 1684–1698.
4. BRÖNNIMANN, H., AND GOODRICH, M. Almost optimal set covers in finite vc-dimension. *DCG 14*, 1 (1995), 463–479.
5. BUS, N., GARG, S., MUSTAFA, N. H., AND RAY, S. Tighter estimates for epsilon-nets for disks. *arXiv preprint arXiv:1501.03246* (2015).
6. CHAN, T. M., AND GRANT, E. Exact algorithms and apx-hardness results for geometric packing and covering problems. *CGTA 47*, 2, Part A (2014), 112 – 124.
7. CHAN, T. M., GRANT, E., KÖNEMANN, J., AND SHARPE, M. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *SODA* (2012), SIAM, pp. 1576–1585.
8. CLARK, B. N., COLBOURN, C. J., AND JOHNSON, D. S. Unit disk graphs. *Discrete Math. 86*, 1-3 (Jan. 1991), 165–177.
9. CLARKSON, K. L., AND VARADARAJAN, K. Improved approximation algorithms for geometric set cover. *DCG 37*, 1 (2007), 43–58.
10. DAI, D., AND YU, C. A $5+\epsilon$-approximation algorithm for minimum weighted dominating set in unit disk graph. *TCS 410*, 8 (2009), 756–765.
11. DING, L., WU, W., WILLSON, J., WU, L., LU, Z., AND LEE, W. Constant-approximation for target coverage problem in wireless sensor networks. In *INFOCOM* (2012), IEEE, pp. 1584–1592.

12. DINUR, I., AND STEURER, D. Analytical approach to parallel repetition. In *STOC* (2014), ACM, pp. 624–633.
13. DU, D.-Z., L KO, K., AND HU, X. *Design and Analysis of Approximation Algorithms*. Springer, 2011.
14. DU, D.-Z., AND WAN, P.-J. *Connected Dominating Set: Theory and Applications*, vol. 77. Springer Science & Business Media, 2012.
15. ERLEBACH, T., GRANT, T., AND KAMMER, F. Maximising lifetime for fault-tolerant target coverage in sensor networks. In *SPAA* (2011), ACM, pp. 187–196.
16. ERLEBACH, T., AND MIHALÁK, M. A (4+ϵ)-approximation for the minimum-weight dominating set problem in unit disk graphs. In *WAOA*, E. Bampis and K. Jansen, Eds., vol. 5893. Springer Berlin Heidelberg, 2010, pp. 135–146.
17. ERLEBACH, T., AND VAN LEEUWEN, E. Ptas for weighted set cover on unit squares. In *APPROX*, M. Serna, R. Shaltiel, K. Jansen, and J. Rolim, Eds., vol. 6302 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 166–177.
18. EVEN, G., RAWITZ, D., AND SHAHAR, S. M. Hitting sets when the vc-dimension is small. *Information Processing Letters 95*, 2 (2005), 358–362.
19. FEIGE, U. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM) 45*, 4 (1998), 634–652.
20. GIBSON, M., AND PIRWANI, I. A. Algorithms for dominating set in disk graphs: breaking the logn barrier. In *ESA*. Springer, 2010, pp. 243–254.
21. HAR-PELED, S., KAPLAN, H., SHARIR, M., AND SMORODINSKY, S. Epsilon-nets for halfspaces revisited. *arXiv preprint arXiv:1410.3154* (2014).
22. HAR-PELED, S., AND LEE, M. Weighted geometric set cover problems revisited. *JoCG 3*, 1 (2012), 65–85.
23. HOCHBAUM, D. S., AND MAASS, W. Approximation schemes for covering and packing problems in image processing and vlsi. *JACM 32*, 1 (1985), 130–136.
24. HOCHBAUM, D. S., AND MAASS, W. Fast approximation algorithms for a nonconvex covering problem. *Journal of Algorithms 8*, 3 (1987), 305 – 323.
25. HUANG, Y., GAO, X., ZHANG, Z., AND WU, W. A better constant-factor approximation for weighted dominating set in unit disk graph. *JCO 18*, 2 (2009), 179–194.
26. HUNT, H. B., III, MARATHE, M. V., RADHAKRISHNAN, V., RAVI, S. S., ROSENKRANTZ, D. J., AND STEARNS, R. E. NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs, 1997.
27. MARX, D. On the optimality of planar and geometric approximation schemes. In *FOCS* (2007), IEEE, pp. 338–348.
28. MUSTAFA, N. H., RAMAN, R., AND RAY, S. Qptas for geometric set-cover problems via optimal separators. *arXiv preprint arXiv:1403.0835* (2014).
29. MUSTAFA, N. H., AND RAY, S. Ptas for geometric hitting set problems via local search. In *SOCG* (2009), ACM, pp. 17–22.
30. PYRGA, E., AND RAY, S. New existence proofs ε-nets. In *SOCG* (2008), ACM, pp. 199–207.
31. VAN LEEUWEN, E. J. *Optimization and approximation on systems of geometric objects*. Phd thesis.
32. VARADARAJAN, K. Epsilon nets and union complexity. In *SOCG* (New York, NY, USA, 2009), SCG '09, ACM, pp. 11–16.
33. VARADARAJAN, K. Weighted geometric set cover via quasi-uniform sampling. In *SOCG* (2010), ACM, pp. 641–648.
34. ZOU, F., WANG, Y., XU, X.-H., LI, X., DU, H., WAN, P., AND WU, W. New approximations for minimum-weighted dominating sets and minimum-weighted connected dominating sets on unit disk graphs. *TCS 412*, 3 (2011), 198–208.