International Journal of Foundations of Computer Science
Vol. 25, No. 5 (2014) 623–639

DOI: 10.1142/S0129054114500257

World Scientific
www.worldscientific.com

AN ALGORITHMIC FRAMEWORK FOR SOLVING GEOMETRIC COVERING PROBLEMS — WITH APPLICATIONS*

TAHA GHASEMI

Software Systems R&D Laboratory, Computer Engineering and IT Department
Amirkabir University of Technology
#424 Hafez Avenue, P. O. Box 15875-4413, Tehran, Iran
Institute for Research in Fundamental Sciences (IPM)
P. O. Box 19395-5746, Tehran, Iran
tghasemi@aut.ac.ir

HOSSEIN GHASEMALIZADEH† and MOHAMMADREZA RAZZAZI‡

Software Systems R&D Laboratory, Computer Engineering and IT Department
Amirkabir University of Technology
#424 Hafez Avenue, P. O. Box 15875-4413, Tehran, Iran
†*ghasemalizadeh@aut.ac.ir*
‡*razzazi@aut.ac.ir*

Received 15 October 2013
Accepted 3 April 2014
Communicated by Juraj Hromkovic

In geometric covering problems, the aim is to cover a set of given points using a minimum number of shapes with prescribed properties. These problems extend in many ways such as covering in the presence of obstacles and partial covering. Since in most cases these problems are NP-hard, researchers have developed a large number of approximation algorithms to solve them.

In this paper, with a more general view on these approaches, we propose an algorithmic framework that gives an approximation algorithm for the covering problem at hand, provided that its prerequisites are satisfied. As the result, we present a general class of geometric covering problems for which one can obtain polynomial-time approximation schemes (PTASs). This class of problems involves covering point sets with bounded shapes in a space with fixed dimension. For a problem to be in this class, it must be possible to find a polynomial number of maximal shapes and it must be possible to cover all the points in a small box with a small number of shapes.

Using this framework, we present two new PTASs for the problems of covering with disks in the presence of polygonal obstacles and covering with sectors (pie-shaped wedges). The first problem has never been addressed before and for the second problem, only a logarithmic approximation algorithm was known in general and a 9-approximation algorithm for wide angles.

Keywords: Geometric covering; covering with disks; covering with obstacles; covering with sectors; set cover problem; algorithmic framework; approximation algorithm; computational geometry.

*This research was in part supported by a grant from IPM (No. CS1391-4-16).

1. Introduction

Covering problems play a central role in the development of algorithms, especially approximation algorithms. In the set cover problem, we have a universe of elements and several subsets of this universe and we want to select a minimum number of these subsets such that every element is in at least one of them. The problem is NP-hard and no sub-logarithmic approximation algorithm exists for it unless P = NP [14]. This barrier motivated the study of a geometric version of the problem, called the geometric set cover problem. In this problem, the elements are points in the Euclidean space and the subsets are implicitly defined by giving a set of geometric shapes in the space. Even simpler, in geometric covering problems, the set of shapes is not given in advance but only their properties are specified. For instance, in the covering-with-disks problem, one is interested in covering a given set of points in the plane with a minimum number of disks with a prescribed radius (the disks can be freely placed everywhere in the plane). Even in this apparently simple case, the problem is strongly NP-hard and thus no fully polynomial-time approximation scheme (FPTAS) exists for it [23].

The set cover problem can be approximated within a logarithmic bound using a simple greedy algorithm [27]. The algorithm can be adapted to solve more general problems such as weighted set cover, submodular set cover, capacitated set cover, and multi-set multi-cover [8, 28]. This greedy algorithm is generalized by Bar-Ilan *et al.* [3] to solve even more general problems.

For the geometric set cover problem there are better results. If the Vapnik–Chervonenkis (VC) dimension of the corresponding set system is bounded (e.g. when the shapes have constant description complexity), there is a $\log(OPT)$-approximation algorithm, where OPT is the size of an optimal solution [5]. The algorithm is an iterative reweighting algorithm, which actually equals solving the linear programming formulation of the problem and then finding an appropriate ε-net over the modified instance of the problem. In addition, if all the shapes are of the same type and there exists a function u such that the union complexity of m of them is $O(m \cdot u(m))$, then there is an $O(\log u(OPT))$-approximation algorithm [2]. For instance, as disks have linear union complexity, we have an $O(1)$-approximation algorithm for the problem of geometric set cover with disks [9]. For the special cases of covering with unit disks and covering with axis-aligned unit squares there are also polynomial-time approximation schemes (PTASs) using local search and dynamic programming [12, 26]. In spite of these results, for many types of shapes there is no hope for a PTAS [6, 25]. There are also efficient algorithms with constant-factor approximation ratios [10, 18].

For geometric covering problems, there are several PTASs. Using a shifting grid, Hochbaum and Maass developed the first PTAS for the covering-with-disks problem [23]. Their approach can solve a more general problem of covering with an arbitrary fixed shape in a fixed dimension $d \geq 1$. Later, Feder and Greene [13] and Gonzalez [20] developed faster PTASs for the covering-with-squares and covering-with-disks

problems. To achieve this, using dynamic programming, they first exactly solved the problem when all the points are in a height-bounded strip, and then applied the shifting strategy of Hochbaum and Maass [23] to obtain a PTAS for the problem in the general case. This shifting strategy is later adapted to solve the partial geometric covering problem where the target is not to cover all the points but a fraction of them [17, 19]. There are also several constant-factor approximation algorithms for the covering-with-disks problem [15, 16].

In this paper, by generalizing the results of Feder and Greene [13] and Gonzalez [20], we propose an algorithmic framework, which can be used to solve more general geometric covering problems. To solve a geometric covering problem using this framework, one must first provide the required sub-procedures according to the problem at hand and then plug them into the framework to obtain a PTAS for the problem. Although it is more general, we also managed to make the descriptions and proofs more compact. As the result of our work, we present a general class of geometric covering problems for which one can obtain PTASs. This class of problems involves covering point sets with bounded shapes in a space with fixed dimension. For a problem to be in this class, it must be possible to find a polynomial number of maximal shapes and it must be possible to cover all the points in a small box with a small number of shapes.

To show the effectiveness of the framework, we tackle several geometric covering problems. The first problem is the covering with disks in the presence of obstacles. This problem is similar to the covering-with-disks problem but there are also several polygonal obstacles in the plane and the centers of the disks should not be placed inside these obstacles. The presence of obstacles or barriers in single-facility location problems was considered by several researchers (we refer to the book of Klamroth [24]) and even in more general cases [4, 21], but, to the best of our knowledge, in the geometric setting no such result exists. The second problem is the problem of covering with sectors. Previously, only an approximation algorithm with logarithmic bound is developed for this problem [29]. In addition, when the sector angle is between $\pi/3$ and π, there is a 9-approximation algorithm [29]. Here, we present the first PTAS for this problem when the sector angle is a fixed arbitrary number.

This paper is structured as follows. In Sec. 2, the framework is introduced and its properties are proved. In Sec. 3, we show how this framework can be used to solve several covering problems that were previously solved. In Sec. 4, we solve the covering-with-disks problem in the presence of obstacles. Finally, in Sec. 5, we use the framework to solve the covering-with-sectors problem.

2. The Framework

In this section, we present an algorithmic framework for solving geometric covering problems. This framework requires several inputs to be specified according to the covering problem at hand and as the result, it gives an approximation algorithm for solving the covering problem. The framework is built on the existing techniques but it provides a unified, more general, and a rather simple view of them.

For a fixed integer $d > 1$, the given problem to the framework is defined by:

(1) A set P of n points in $\Re^d$.
(2) A set U of bounded regions in the space. For convenience, we usually refer to the members of this set as shapes.

For a region $c \subseteq \Re^d$, let $P(c)$ denote those points in P that are contained in c. For a set C of regions we define $P(C) = \bigcup_{c \in C} P(c)$. In the (complete) covering problem, we are looking for a subset $C \subseteq U$ of shapes with minimum cardinality such that $P(C) = P$. We assume that $P(U) = P$ and thus the problem has at least one feasible solution.

Let $S \subseteq U$ be a finite subset of shapes such that for any shape $c \in U$ with $P(c) \neq \emptyset$, we can find at least one shape in S, denoted by $S(c)$, such that $P(c) \subseteq P(S(c))$. The definition of S enables us to confine our search for the feasible solutions in the subsets of S only. The following lemma proves this fact.

Lemma 1. *Let $C \subseteq U$ be a feasible solution for the covering problem. There is another feasible solution $C' \subseteq S$ such that $|C'| \leq |C|$.*

Proof. The definition of S implies that any feasible solution C for the covering problem can be converted to a solution $C' = \bigcup_{c \in C} S(c)$ and hence $C' \subseteq S$, and $|C'| \leq |C|$. □

Since an optimal solution is a feasible solution, we have the following corollary.

Corollary 2. *There is an optimal solution OPT to the covering problem where $OPT \subseteq S$.*

Let $B = B(a_1, a_2, ..., a_d)$ be a d-dimensional axis-parallel hyperrectangle with side lengths of $a_1, a_2, ..., a_d$ along the $x_1-, x_2-, ..., x_d-$axes, respectively. Define $m(B)$ as a function such that $m(B) \geq \min\{|C| : C \subseteq S \wedge P(B) \subseteq P(C)\}$. In words, $m(B)$ specifies how many shapes from S are sufficient to cover all the points in B. To solve the covering problem, the following inputs should be provided to the framework. We assume that the set P of points is given and fixed.

(1) An algorithm to compute S. We assume that this algorithm runs in time $T_s(n)$.
(2) A function $m(a_1, a_2, ..., a_d) = \max m(B)$, where the maximum is taken over all possible placements of $B = B(a_1, a_2, ..., a_d)$. We assume that this function is computable in constant time. Let $\hat{m}(a, b) = m(a, b, b, .., b)$.

For a shape c, let $prj_i(c)$ be the length of the projection of c on the x_i-axis. Consider a function $l(i)$ such that $l(i) \geq \max\{prj_i(c) : c \in S\}$. This means that no shape in S has length more than $l(i)$ in the i^{th} dimension. Since all the shapes are bounded, without loss of generality, we assume that the problem instance, i.e. its points and its shapes, is scaled such that $l(i) = 1$, $1 \leq i \leq d$. By this assumption

and Corollary 2, each shape in OPT can be bounded in a unit hypercube. In the rest of this section, we prove the following theorem:

Theorem 3. *For a given $0 < \varepsilon \leq 1$, there exists an algorithm that solves the covering problem with an approximation factor of $(1+\varepsilon)^{d-1}$ in $O(n \log n + T_s(n) + \varepsilon^{1-d} n |S|^{\hat{m}(2,\varepsilon^{-1})+1})$ time.*

The framework uses the shifting technique of Hochbaum and Maass [23] to reduce the problem to the case in which the points are in a height-bounded strip and then applies dynamic programming to solve the problem in this special case. The dynamic programming is based on the algorithms of Gonzalez [20] and Feder and Greene [13], but to make it more compact and comprehensible we state it using backward dynamic programming recursions instead of describing it in the forward manner.

For $h \geq 0$, a d-dimensional hyperstrip $H = H(a_2, ..., a_d)$ with height h (along the x_1-axis) is defined as $H = [-\infty, +\infty] \times [a_2, a_2 + h) \times ... \times [a_d, a_d + h)$ where $a_i \in \Re$ for $1 < i \leq d$. Given a $(d-1)$-dimensional vector $\vec{a} = (a_2, a_3, ..., a_d) \in \Re^{d-1}$, we also use the notation $H_h(\vec{a})$ to denote $H(a_2, ..., a_d)$ with height h. For $k \geq 1$, let $[0..k)$ denote the set of integers from 0 to $k-1$. With these definitions, the outline of the framework is as follows:

Algorithm COVER

Input.

A set P of points in $\Re^d$.
A set U of bounded regions in the space.
A positive integer k, which indicates the solution quality, the higher value the better solution.

Output.

A set R of shapes such that $P(R) = P$.

Sort the points in P in increasing order of their x_1-coordinates.
Compute S from U.
Let $\vec{a} \in [0..k)^{d-1}$ be a $(d-1)$-dimensional vector of integers. Define the following functions:

$$P(\vec{a}, \vec{b}) \leftarrow P(H_k(\vec{a} + k\vec{b})), \text{ where } \vec{b} \in \mathbb{Z}^{d-1}$$
$$Part(\vec{a}) \leftarrow \{P(\vec{a}, \vec{b}) : \vec{b} \in \mathbb{Z}^{d-1} \wedge P(\vec{a}, \vec{b}) \neq \emptyset\}$$

$R(\vec{a}) \leftarrow \bigcup_{\hat{P} \in Part(\vec{a})} StripCover(\hat{P}, S, k)$
$R \leftarrow R(\vec{a})$ where $\vec{a} = \arg\min\{|R(\vec{a})| : \vec{a} \in [0..k)^{d-1}\}$

Algorithm STRIPCOVER

Input.

A sequence $\hat{P} = p_1, ..., p_{\hat{n}}$ of $\hat{n}$ points, sorted in increasing order of their x_1-coordinates.
A set S of shapes.
A positive integer k. All the points in $\hat{P}$ must lie in a hyperstrip of height k.

Output.

A subset $\hat{S}$ of S such that $\{p_1, ..., p_{\hat{n}}\} \subseteq P(\hat{S})$.

For a subset C of S, let $C|_i$ denote those shapes in C that intersect with the $(d-1)$-dimensional hyperplane that passes through p_i and is normal to the x_1-axis.
Use dynamic programming with the following recursion:

$$N_i(C) = \begin{cases} 0 & i = 0 \\ \infty & i > 0, |C| > \hat{m}(2,k) \\ N_{i-1}(C|_{i-1}) & i > 0, |C| \leq \hat{m}(2,k), p_i \in C \\ \min_{c \in A_i} N_{i-1}((C \cup \{c\})|_{i-1}) + 1 & i > 0, |C| \leq \hat{m}(2,k), p_i \notin C \end{cases}$$

where $C \subseteq S|_i$ and $A_i = \{c \in S : p_i \in P(c)\}$.
Compute $N_{\hat{n}}(\emptyset)$ and derive the corresponding solution $\hat{S}$.

The COVER algorithm first sorts the points so that the STRIPCOVER algorithm can iterate through the points one by one in increasing order of their x_1-coordinates. It then computes S and only searches for the solution in the subsets of S.

For each value of $\vec{a} \in [0..k)^{d-1}$, $Part(\vec{a})$ defines one partition of points in P, where each part is a hyperstrip of height k with at least one point in it. Therefore, with different values of $\vec{a}$ the algorithm partitions the points in different ways. For each value of $\vec{a} \in [0..k)^{d-1}$, the COVER algorithm solves the problem according to the partition defined by $Part(\vec{a})$, by separately solving the problem for each part of the points $\hat{P}$ in $Part(\vec{a})$, and taking the union of the results as $R(\vec{a})$. The result of the COVER algorithm is one of $R(\vec{a})$s with a minimum number of shapes among all $R(\vec{a})$, $\vec{a} \in [0..k)^{d-1}$. To solve the problem for each part $\hat{P}$ of the points, the COVER algorithm uses the STRIPCOVER algorithm.

As we will prove later, the STRIPCOVER algorithm exactly solves the covering problem when the points are in a hyperstrip with height k. We call this problem the *strip cover problem*. The STRIPCOVER algorithm takes the input points as a sequence $\hat{P} = p_1, .., p_{\hat{n}}$, which are sorted in increasing order of their x_1-coordinates. Let v_i be the $(d-1)$-dimensional hyperplane that passes through p_i and is normal to the x_1-axis. Given a subset C of S, the algorithm uses a filtering procedure to

obtain $C|_i$, i.e., those shapes in C that intersect v_i. Let $N_i(C)$ be the minimum number of required shapes in S such that they cover the points $\{p_1, .., p_i\} \backslash P(C)$, where $C \subseteq S|_i$. Intuitively, C maintains those shapes in the solution that may further affect the solution of subproblems. Clearly we look for $N_{\hat{n}}(\emptyset)$. The algorithm computes $N_i(C)$ recursively as follows. If p_i is already covered by C we just need to compute $N_{i-1}(C|_{i-1})$. Otherwise, we must add a shape $c \in S$ that covers p_i. The algorithm tests every candidate shape in $A_i = \{c \in S : p_i \in P(c)\}$ and returns a solution with the minimum number of shapes. In the base case, we have $N_0(C) = 0$ for any set of shapes C. In addition, as will turn out, the algorithm rules out those subproblems with $|C| > \hat{m}(2, k)$ to keep its running time polynomial.

We first analyze the COVER algorithm, treating the STRIPCOVER algorithm as a black box.

Lemma 4. *Suppose the STRIPCOVER algorithm is correct and runs in $O(T_c(\hat{n}, |S|, k))$ time, where $T_c(\hat{n}, |S|, k)$ is a polynomial function of $\hat{n}$. Then, algorithm COVER is also correct and runs in $O(n \log n + T_s(n) + k^{d-1}(nd + T_c(n, |S|, k)))$ time, where $T_s(n)$ is the required time to compute the set S.*

Proof. For correctness, we must prove that the algorithm returns a set of shapes that cover all the points. By Lemma 1, if we restrict the search to the subsets of S, we do not lose any feasible solution. For a given $\vec{a} \in [0..k)^{d-1}$, $Part(\vec{a})$ partitions the point set P, and $R(\vec{a})$ is the union of the solutions returned by the STRIPCOVER algorithm for each part of the points. Thus, by our assumption that STRIPCOVER is correct, $R(\vec{a})$ covers all the points in P.

For the running time, the first two steps take $O(n \log n + T_s(n))$ time. For each value of $\vec{a} \in [0..k)^{d-1}$, computing $Part(\vec{a})$ takes $O(nd)$ time using the rounding-and-hashing technique described in [22]. Having $Part(\vec{a})$, the computation of $R(\vec{a})$ takes $\sum_{\hat{P} \in Part(\vec{a})} T_c(|\hat{P}|, |S|, k)$ and since the non-empty sets $\hat{P} \in Part(\vec{a})$ form a partition of P, and T_c is a polynomial function of $\hat{n}$, it sums to $T_c(n, |S|, k)$. □

Algorithm COVER partitions the input points into a set of hyperstrips and use algorithm STRIPCOVER to exactly solves the problem in each of these hyperstrips. This produces one possible solution for the problem. The COVER algorithm then repeatedly shifts these partitions in different directions and solves the problem according to each generated partition. After this process, for each partition, one solution is generated for the problem. Assume there are ns solutions generated. Finally, the algorithm chooses a minimum-size solution among these ns solutions. This approach produces a bounded-ratio approximation because each of the shapes in an optimal solution intersects with a limited number of those hyperstrips, say ni of them. Therefore, in overall, the solutions produced by the COVER algorithm contain at most ni shapes for each shape in the optimal solution. Choosing a minimum-size solution ensures that the returned solution has an approximation factor of at most ni/ns. The following two lemmas state these facts formally.

Lemma 5. *For $\vec{a} \in [0..k)^{d-1}$ and $\vec{b} \in \mathbb{Z}^{d-1}$, let OPT be an optimal solution to the covering problem and $OPT(\vec{a}, \vec{b})$ be a minimal subset of OPT that covers points in $P(\vec{a}, \vec{b})$. For each shape $s \in OPT$, there are at most $(k+1)^{d-1}$ values of $\vec{a}$ and $\vec{b}$ where $s \in OPT(\vec{a}, \vec{b})$.*

Proof. We prove this lemma using induction on the dimension d. Let $\eta(\vec{a}, s) = |\{\vec{b} : \vec{b} \in \mathbb{Z}^{d-1} \wedge s \in OPT(\vec{a}, \vec{b})\}|$, where $s \in OPT$. First consider the base case $d = 2$. For $0 \leq a < k$ and $b \in \mathbb{Z}$, note that since $k \geq 1$, $P(a, b)$ determines a set of points in a horizontal strip of height no less than one. By Corollary 2, s can be bounded by a unit square. Thus, by the minimality of $OPT(a, b)$, for each shape $s \in OPT$, there is at most one value of a where $\eta(a, s) \leq 2$ and for the other values of a, $\eta(a, s) = 1$. Thus, there at at most $2 + (k-1) = k+1$ values for a and b where $s \in OPT(a, b)$.

Assuming the claim holds for $d-1$, in d dimensions, first project all the shapes and points on the x_d-axis. Let $s(d)$ be the projection of s on the x_d-axis and $s'(d)$ be the projection of s on the $(x_1, x_2, ..., x_{d-1})$-subspace. Similar to the argument of the base case, for each shape $s \in OPT$, there is at most one value of a_d where $\eta(a_d, s(d)) \leq 2$, i.e. there are at most two values of b_d where $s \in OPT(a_d, b_d)$, and with other values of a_d, $\eta(a_d, s(d)) = 1$, i.e. there is one value of b_d where $s \in OPT(a_d, b_d)$. In each case, by the induction hypothesis, there are at most $(k+1)^{d-2}$ values for $\vec{a}' = (a_2, ..., a_{d-1})$ and $\vec{b}' = (b_2, ..., b_{d-1})$ where $s'(d) \in OPT(\vec{a}', \vec{b}')$. Thus, there are at most $2(k+1)^{d-2} + (k-1)(k+1)^{d-2}$ values for $\vec{a} = (a_2, ..., a_d)$ and $\vec{b} = (b_2, ..., b_d)$ where $s \in OPT(\vec{a}, \vec{b})$. □

Lemma 6. *Suppose the STRIPCOVER algorithm exactly solves the covering problem when the points are in a height-bounded hyperstrip. Then, for a given integer $k \geq 1$, the COVER algorithm is a $(1 + 1/k)^{d-1}$-approximation algorithm.*

Proof. Let OPT be an optimal solution to the covering problem and $OPT(\vec{a}, \vec{b})$ be a minimal subset of OPT that covers the points in $P(\vec{a}, \vec{b})$. Let R be the solution returned by the COVER algorithm, $R(\vec{a})$ be the solution according to $Part(\vec{a})$, and $R(\vec{a}, \vec{b})$ be the solution of the STRIPCOVER algorithm applied to the points in $P(\vec{a}, \vec{b})$. We have the following averaging argument:

$$\begin{aligned} k^{d-1}|R| &\leq \textstyle\sum_{a \in [0..k)^{d-1}} |R(\vec{a})| \\ &\leq \textstyle\sum_{a \in [0..k)^{d-1}} \sum_{\vec{b} \in \mathbb{Z}^{d-1}} |R(\vec{a}, \vec{b})| \\ &\leq \textstyle\sum_{a \in [0..k)^{d-1}} \sum_{\vec{b} \in \mathbb{Z}^{d-1}} |OPT(\vec{a}, \vec{b})| \\ &\leq (k+1)^{d-1}|OPT| \end{aligned}$$

where the first inequality comes from the fact that R is one of the $R(\vec{a})$s with a minimum number of shapes, the second inequality is based on this fact that $R(\vec{a})$ is computed from the union of $R(\vec{a}, \vec{b})$ over all $\vec{b}$s, the third inequality is due to the fact that STRIPCOVER is an exact algorithm and hence $|R(\vec{a}, \vec{b})| \leq |OPT(\vec{a}, \vec{b})|$, and the last inequality is the result of Lemma 5. □

To prove the correctness of the STRIPCOVER algorithm we need the following packing lemma, which is based on an observation made by Gonzalez [20] and Feder and Green [13]:

Lemma 7. *There is an optimal solution $O\hat{P}T$ to the strip cover problem such that $|O\hat{P}T|_i| \leq \hat{m}(2,k)$, for $1 \leq i \leq \hat{n}$.*

Proof. Let $O\hat{P}T$ be an optimal solution for the strip cover problem. If there is an i, $1 \leq i \leq \hat{n}$, such that $|O\hat{P}T|_i| > \hat{m}(2,k)$ we can do the following. Since, by Corollary 2, each shape in OPT can be bounded in a hypercube with the side length of one, those intersecting v_i can cover the points in a region of size at most $B = B(2,k,k,...,k)$. Since we can cover the points in B with at most $m(B)$ shapes, we would obtain a new solution with fewer shapes, a contradiction. □

Lemma 8. *Algorithm STRIPCOVER exactly solves the strip cover problem.*

Proof. By Lemma 7, ignoring those $N_i(C)$s with $|C|_i| > \hat{m}(2,k)$ maintains the exactness of the algorithm. When we want to compute $N_i(C)$, either $p_i \in C$ or $p_i \notin C$. If $p_i \in C$, then $N_i(C) = N_{i-1}(C|_{i-1})$. This is because p_i is already covered and if any bounded shape $s \in C$ wants to cover a point before p_i, it must intersect v_{i-1} and thus $s \in C|_{i-1}$. On the other hand, if $p_i \notin C$, p_i must be covered by one of the shapes $c \in A_i$. In this case, the size of the solution will be increased by 1. Again, if a shape from $C \cup \{c\}$ wants to cover a point before p_i, it must intersect v_{i-1}. □

Lemma 9. *Algorithm STRIPCOVER runs in $O(\hat{n}|S|^{\hat{m}(2,k)+1})$ time.*

Proof. Since we use dynamic programming, we need to compute the value of $N_i(C)$, for each possible values of i and C, only once. Thus, the total running time is the product of the number of different $N_i(C)$s and the time needed for computing each $N_i(C)$.

There are $O(\hat{n}|S|^{\hat{m}(2,k)})$ different $N_i(C)$s, since we have $O(|S|^{\hat{m}(2,k)})$ different subsets C of S with $|C| \leq \hat{m}(2,k)$ and $\hat{n}$ different values for i.

Now we prove that the computation of each $N_i(C)$ takes $O(|S|)$ time. By our assumption, the computation of m is performed in constant time. Thus, the computation of the first three cases of the $N_i(C)$ formula takes $O(1)$ time. The most time computing case is the computation of the last case. By checking the containment of each point in each shape, one can compute all A_i, for $1 \leq i \leq n$, at once, which takes $O(\hat{n}|S|)$ time, assuming containment can be checked in constant time. Also note that the computation of $(C \cup c)|_{i-1}$ can be done by first computing $C|_{i-1}$ only once and then checking the inclusion of each c separately. Therefore, since $|A_i| \leq |S|$ and $|C| \leq |S|$ and assuming the intersection can be checked in constant time, the forth case of the $N_i(C)$ formula can be computed in $O(|S|)$ time. □

The combination of the above lemmas proves Theorem 3.

3. Classical Geometric Covering Problems

In this section, we illustrate the usage of our framework, by solving some of the existing geometric covering problems in the literature. In the covering-with-squares problem, the aim is to cover a given set of points with a minimum number of unit-length axis-aligned squares (unit squares for short). In terms of our framework, the input for this problem is defined as:

(1) P : a set of n points in the plane.
(2) U: the set of all possible unit squares in the plane.

To be able to solve this problem with the framework, we must specify both S and m. Similar to [20] and [1], we say a square is anchored if it has one point on its left side and one (not necessarily distinct) point on its bottom side. Given the set P of points, let S be the set of all possible anchored squares on these points. Indeed, S satisfies the requirement of the framework since given a unit square in the plane with at least one point in it, one can always shift it up and right until it first becomes anchored without uncovering any previously covered points. $|S| = O(n^2)$ and we can use a simple brute-force method to compute S in $T_s(n) = O(n^2)$ time. For m, clearly we can fully tile an $a_1 \cdot a_2$ rectangle (i.e. $B(a_1, a_2)$) with $\lceil a_1 \rceil \cdot \lceil a_2 \rceil$ unit squares. Since by applying the aforementioned shifting, one can find the same number of anchored squares in S covering the same set of points, $m(a, b) = \lceil a_1 \rceil \cdot \lceil a_2 \rceil$. Using scaling and rotation, we can also solve the more general problem of covering with fixed-size fixed-orientation rectangles. The generalization to d dimensions is also straightforward. Thus, we have the following corollary of Theorem 3:

Corollary 10. *For a given $0 < \varepsilon \leq 1$ and a fixed integer $d \geq 2$, there exists a $(1 + \varepsilon)^{d-1}$-approximation algorithm that solves the d-dimensional covering with fixed-size fixed-orientation hyperrectangles problem in $O(\varepsilon^{1-d} n^{2d\lceil \varepsilon \rceil^{1-d} + d + 1})$ time.*

Similarly, we can solve the covering-with-disks problem. In this problem, one is interested in covering a given set of points in the plane with the minimum number of unit disks. Define S to be the set of unit disks either with at least two points on their boundaries or with one point as their centers. It is easy to see that S satisfies the requirement of the framework, $|S| = O(n^2)$, and $T_s(n) = O(n^2)$. By considering the maximum bounded square inside the unit disks to tile $B(a_1, a_2)$, we conclude that $m(a, b) = O(a_1 \cdot a_2)$. Again, the generalization to covering with the fixed-size fixed-orientation ellipsis, and d dimensions is straightforward, and hence we have the following corollary of Theorem 3:

Corollary 11. *For a given $0 < \varepsilon \leq 1$ and a fixed integer $d \geq 2$, there exists a $(1 + \varepsilon)^{d-1}$-approximation algorithm that solves the d-dimensional covering with fixed-size fixed-orientation hyperellipsis problem in $n^{O(\varepsilon^{1-d})}$ time.*

4. Covering with Disks in the Presence of Obstacles

As a new application, we apply our framework to solve the following problem, known as the covering problem in the presence of obstacles: Let F be a simple polygon in the plane, and let $O_1, ..., O_k$ be the pairwise disjoint simple polygons inside F, and let $O = \bigcup_{1 \le i \le k} O_i$. Having a set P of n points in $F \backslash O$, we want to find a minimum number of unit disks whose centers are in $F \backslash O$ such that they cover all the points in P. In other words, O_i defines an obstacle and F defines the target region and hence $F \backslash O$ is the free space which we are allowed to put the disk centers in. Stating the problem in terms of the framework, we have a set P of points and a set U, which is the set of all unit disks whose centers are inside $F \backslash O$. To solve the problem using the framework we must specify S and m. Let t be the total description complexity of $F \backslash O$. We scale the problem so that all the disks have radius $\frac{1}{2}$.

Lemma 12. *For the problem of covering with disks in the presence of obstacles, S can be computed in $O(tn^2 \log(t + n^2))$ time. Furthermore, $|S| = O(n^2)$.*

Proof. Let D be the set of $\frac{1}{2}$ disks whose centers are all the points in P. Let C be the 2-dimensional cells of the arrangement $\mathrm{A}(D)$. Observe that for a given cell $c \in C$, all $\frac{1}{2}$ disks whose centers are inside c, cover the same set of points, and for two different cells $c_1, c_2 \in C$, a $\frac{1}{2}$ disk whose center is inside c_1 covers a different subset of points than a $\frac{1}{2}$ disk whose center is inside c_2 (see Fig. 1). In other words, cells define combinatorially different classes for the placement of disks. Note that each edge or vertex of $\mathrm{A}(D)$ also defines a class that is equivalent to the class defined by one of its incident cells and hence it is not needed to consider them separately. Thus to compute S, for each cell $c \in C$, we compute $c' = c \cap (F \backslash O)$ and if $c' \neq \emptyset$, we add a $\frac{1}{2}$ disk whose center is arbitrarily placed in c'. Since $|C| = O(n^2)$ we conclude that $|S| = O(n^2)$.

To find those cells of $\mathrm{A}(D)$ that are not completely covered by O, we use the well-known plane sweep algorithm that computes the overlay of two subdivisions [11]. The two subdivisions are $\mathrm{A}(D)$ and $F \backslash O$. We can compute $\mathrm{A}(D)$ using the algorithm of Chazelle and Lee [7], which works in $O(n^2)$ time. The plane sweep algorithm assumes a total ordering among the edges at each event point for finding the intersections. To align with this assumption, we use the common trick of splitting each circle in $\mathrm{A}(D)$ into two half-circles. Let $\bar{f}$ be the face in $F \backslash O$ that corresponds to the free space in it. The overlay computing algorithm, in its last step, labels each face of the overlay according to the constituting faces from the corresponding two subdivisions. For each face of the overlay, if the constituting face from $F \backslash O$ is $\bar{f}$, we add the constituting face from $\mathrm{A}(D)$ to a set R. R is initially empty and clearly, at the end, R contains those faces of $\mathrm{A}(D)$ that are not completely covered by O. The description complexity of $\mathrm{A}(D)$ is $O(n^2)$ and the description complexity of $F \backslash O$ is t. The description complexity of their overlay is $O(tn^2)$, and hence the algorithm takes $O(tn^2 \log(t + n^2))$ time. □

Lemma 13. *In the problem of covering with disks in the presence of obstacles,* $m(a_1, a_2) = \lceil 2\sqrt{2}a_1 \rceil \lceil 2\sqrt{2}a_2 \rceil$.

Proof. For $B = B(a_1, a_2)$, if $P(B) = \emptyset$, $m(B) = 0$. Otherwise, we partition B into a regular grid of spacing $\sqrt{2}/4$. For each grid cell s, if $P(s) \neq \emptyset$, by the definition of the problem, the points in $P(s)$ are all in $s' = s \cap (F \backslash O)$ which implies that $s' \neq \emptyset$ and thus we can place a $\frac{1}{2}$ disk r whose center is in s'. Clearly, $s \subseteq r$ and hence r covers all the points in s. Thus, using at most $\lceil 2\sqrt{2}a_1 \rceil \lceil 2\sqrt{2}a_2 \rceil$, $\frac{1}{2}$ disks whose centers are in $F \backslash O$ we can cover all the points in B. Note that although we did not explicitly choose disks from S, we can simply convert the selected disks to the corresponding disks in S without increasing the number of disks, and thus $m(a_1, a_2) = \max m(B) = \lceil 2\sqrt{2}a_1 \rceil \lceil 2\sqrt{2}a_2 \rceil$. □

Thus, we have the following corollary of Theorem 3:

Corollary 14. *For a given* $0 < \varepsilon \leq 1$*, there exists an algorithm that solves the covering-with-disks problem in the presence of obstacles with an approximation factor of* $(1 + \varepsilon)$ *in* $O(tn^2 \log(t + n^2) + \varepsilon^{-1} n^{\lceil 6\sqrt{2}\varepsilon^{-1}+3 \rceil})$ *time.*

5. Covering with Sectors

In this section, we solve the problem of covering a point set with a minimum number of sectors. The radius of each sector is one and it has a fixed angle. To use the framework for solving this problem, we must specify S and m. First we define anchored sectors and present the related Lemma 15 which is already given in [29].

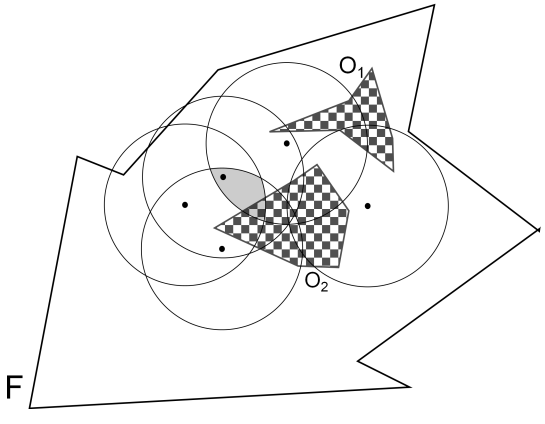

Fig. 1. Demonstration of the proof of Lemma 12. The two obstacles are filled with checkerboards. There are five points in P and D contains the depicted five disks. A typical two-dimensional cell of the arrangement $\mathrm{A}(D)$ is highlighted with the gray color.

Let s be a sector with radius one and angle θ. We denote the center of s as c and the arc of s as *arc*. Walking on the arc clockwise around c, the first edge encountered is called e_1 and the second edge encountered is called e_2 (see Fig. 2).

We consider i_1 as the intersection of *arc* and e_1 and i_2 as the intersection of *arc* and e_2. Let c' be the middle point of *arc*. The vector $\overrightarrow{c'c}$ is the direction vector of s and we refer to it by dv. Figure 2 represents the specification of a sector.

A sector s is anchored if one of the following cases occurs:

(1) If s contains one point located on c (Fig. 3(a)).
(2) If s contains two points, one of these points is located on i_1 or i_2, and the other point is located on e_1 or e_2 (Fig. 3(b)) or on *arc* (Fig. 3(c)).
(3) If s contains three or more points, we may have two points on *arc* and one point on e_1 or e_2 (Fig. 3(d)), or two points on either e_1 or e_2 and one point on *arc* (Fig. 3(e)), or one point on e_1 and one point on e_2 and one point on *arc* (Fig. 3(f)).

We define three types of transitions, which we apply on a sector to obtain an anchored sector.

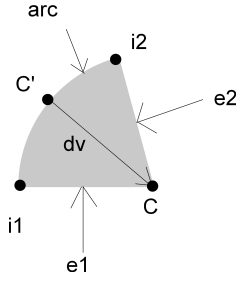

Fig. 2. Specification of a sector.

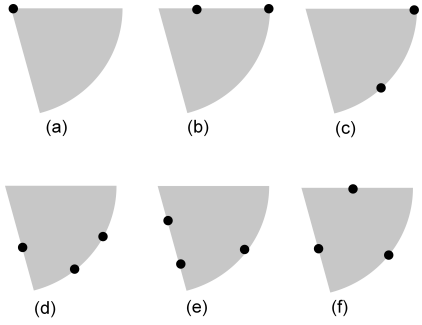

Fig. 3. Anchored sectors with one, two, or three points.

(1) Move the sector along dv until at least one point hits arc. Call this point p.
(2) Rotate the sector clockwise around c until at least one point hits e_1 or p hits i_1.
(3) Let c_p be a circle with radius one and its center in point p. Move the center of the sector clockwise on c_p until one point hits e_1, e_2 or arc.

Figure 4 shows these three types of transitions.

Lemma 15. *Every sector can be anchored such that the anchored sector covers the same set of points.*

Proof. Let s be a sector that covers a subset of P, which is denoted by $s(P)$. After applying the introduced transitions to s, we obtain an anchored sector s' which covers the same set of points $s(P)$. First, we apply transition 1 to s. If two points reach arc we only apply transition 2 to s. In this case, s is anchored similar to Fig. 3(d). If one point reaches arc, we call this point p_1 and then we apply transition 2 to s. If more than one point reach e_1 then s is anchored as in Fig. 3(e). If one point reaches e_1 then we apply transition 3 to s. If one point reaches e_1, then s is anchored like Fig. 3(e) and if one point reaches e_2 then s is anchored as in Fig. 3(f). None of these transitions uncover a point from $s(P)$. □

Lemma 16. *The set S in the covering-with-sectors problem can be computed in $O(n^3)$ time. Furthermore, $|S| = O(n^3)$ and $T_s(n) = O(n^3)$.*

Proof. Based on Lemma 15, every sector can be anchored with at most three points and hence the number of all anchored sectors in P is in $O(n^3)$. We can obtain S by considering every one, two, or three points of S and obtaining the anchored sectors with the selected points in $O(1)$ time. Hence, S can be computed in $O(n^3)$ time for covering-with-sectors problem. □

Lemma 17. *For the problem of covering with sectors, $m(a_1, a_2) = 2\left\lceil \frac{a_1}{\cos\theta} \right\rceil \left\lceil \frac{a_2}{\sin\theta} \right\rceil$.*

Proof. Two adjacent sectors, called sector rectangle, as shown in Fig. 5, can cover a rectangle with sides lengths of $\sin\theta$ and $\cos\theta$. An $a_1 \cdot a_2$ rectangle can be covered

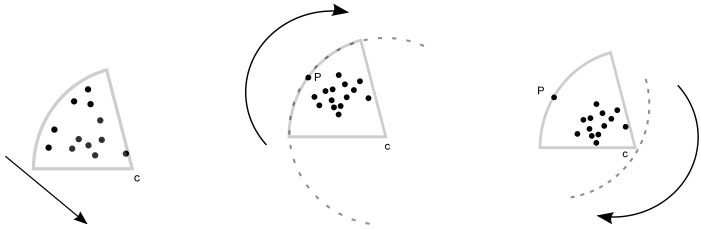

Fig. 4. Different types of anchor transitions.

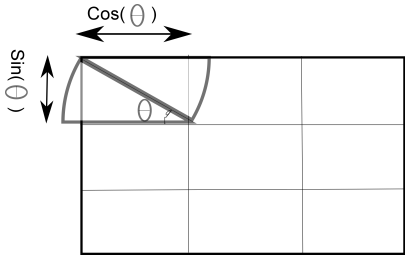

Fig. 5. Obtaining the function $m(a_1, a_2)$ for the covering-with-sectors problem.

by $\lceil \frac{a_1}{\cos\theta} \rceil \lceil \frac{a_2}{\sin\theta} \rceil$ sector rectangles, or consequently by $2\lceil \frac{a_1}{\cos\theta} \rceil \lceil \frac{a_2}{\sin\theta} \rceil$ sectors. Thus, $m(a_1, a_2) = 2\lceil \frac{a_1}{\cos\theta} \rceil \lceil \frac{a_2}{\sin\theta} \rceil$. □

Finally, we have the following corollary of Theorem 3:

Corollary 18. *For a given $0 < \varepsilon \leq 1$ and a fixed constant θ, there exists an algorithm that solves the covering-with-sectors problem with an approximation factor of $(1+\varepsilon)$ in $O(n \log n + n^3 + \varepsilon^{-1} n^{3\lceil \frac{2}{\cos\theta} \rceil \lceil \frac{\varepsilon^{-1}}{\sin\theta} \rceil + 4})$ time.*

6. Conclusion

By slightly generalizing previous results, we developed an algorithmic framework for solving the geometric covering problems. The framework requires two problem specific procedures: an algorithm to compute S and an algorithm to compute $m(a_1, a_2, ..., a_d)$. Having these two procedures for a problem, one can devise a PTAS for it. We showed how these two procedures can be obtained for the problems of covering with disks in the presence of obstacles and covering with sectors. As future work, solving the problem of covering with sectors in the presence of obstacles seems interesting. In addition, the framework can possibly be extended to solve even more general covering problems such as the partial and capacitated covering problems.

Acknowledgments

We thank the referees for their helpful comments, which improved the paper structure and content. The authors also would like to thank Institute for Research in Fundamental sciences (IPM) for their support of this project.

References

[1] P. K. Agarwal and C. M. Procopiuc, Exact and approximation algorithms for clustering, *Algorithmica* **33** (2002) 201–226.

[2] B. Aronov, E. Ezra and M. Shair, Small-size ε-nets for axis-parallel rectangles and boxes, *Proceedings of the forty-first annual ACM symposium on Theory of computing* (ACM, 1536501), pp. 639–648.

[3] J. Bar-Ilan, G. Kortsarz and D. Peleg, Generalized submodular cover problems and applications, *Theoretical Computer Science* **250**(1-2) (2001) 179–200.

[4] M. Bischoff, T. Fleischmann and K. Klamroth, The multi-facility location-allocation problem with polyhedral barriers, *Computers and Operations Research* **36**(5) (2009) 1376–1392, 0305-0548 doi: 10.1016/j.cor.2008.02.014.

[5] H. Bronnimann and M. T. Goodrich, Almost optimal set covers in finite vc-dimension, *Discrete and Computational Geometry* **14** (1995) 463–479.

[6] T. M. Chan and E. Grant, Exact algorithms and apx-hardness results for geometric set cover, 23*rd Canadian Conference on Computational Geometry* (*CCCG*), pp. 431–436.

[7] B. M. Chazelle and D. T. Lee, On a circle placement problem, *Computing* **36** (1986) 1–16.

[8] J. Chuzhoy and J. Naor, Covering problems with hard capacities, *SIAM Journal on Computing* **36**(2) (2006) 498–515.

[9] K. L. Clarkson and K. R. Varadarajan, Improved approximation algorithms for geometric set cover, *Discrete and Computational Geometry* **37**(1) (2007) 43–58.

[10] F. Claude, G. K. Das, R. Dorrigiv, S. Durocher, R. Fraser, A. Lopez-Ortiz, B. G. Nickerson and A. Salinger, An improved line-separable algorithm for discrete unit disk cover, *Discrete Mathematics, Algorithms and Applications* **2** (2010) 77–87.

[11] M. De Berg, O. Cheong and M. Van Kreveld, *Computational geometry: algorithms and applications* (Springer-Verlag, New York, 2008).

[12] T. Erlebach, E. van Leeuwen, M. Serna, R. Shaltiel, K. Jansen and J. Rolim, *PTAS for Weighted Set Cover on Unit Squares*, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science, Vol. 6302 (Springer, Berlin/Heidelberg, 2010), pp. 166–177. 978-3-642-15368-6.

[13] T. Feder and D. H. Greene, Optimal algorithms for approximate clustering, *Proceedings of the* 20*th Annual ACM Symposium on the Theory of Computing*, pp. 434–444.

[14] U. Feige, A threshold of ln n for approximating set cover, *Journal of the ACM* **45** (1998) 634–652.

[15] M. Franceschetti, M. Cook and J. Bruck, A geometric theorem for approximate disk covering algorithms, Report ETR035, Caltech (2001).

[16] B. Fu, Z. Chen and M. Abdelguerfi, *An Almost Linear Time* 2.8334-*Approximation Algorithm for the Disc Covering Problem*, *Algorithmic Aspects in Information and Management* eds. M.-Y. Kao and X.-Y. Li, Lecture Notes in Computer Science, Vol. 4508 (Springer, Heidelberg/Berlin, 2007), book section 30, pp. 317–326.

[17] R. Gandhi, S. Khuller and A. Srinivasan, Approximation algorithms for partial covering problems, *Journal of Algorithms* **53**(1) (2004) 55–84.

[18] K. D. Gautam, F. Robert, L. Alejandro, O. pez and G. N. Bradford, On the discrete unit disk cover problem (2011), 1966191 146-157.

[19] H. Ghasemalizadeh and M. Razzazi, An improved approximation algorithm for the most points covering problem, *Theory of Computing Systems* **50**(3) (2012) 545–558, 1432-4350.

[20] T. F. Gonzalez, Covering a set of points in multidimensional space, *Information Processing Letters* **40**(4) (1991) 181–188.

[21] D. Halperin, M. Sharir and K. Goldberg, The 2-center problem with obstacles, *Journal of Algorithms* **42**(1) (2002) 109–134.

[22] S. Har-Peled and S. Mazumdar, Fast algorithms for computing the smallest k-enclosing circle, *Algorithmica* **41**(3) (2005) 147–157.

[23] D. S. Hochbaum and W. Maass., Approximation schemes for covering and packing problems in image processing and vlsi, *Journal of the ACM* **32**(1) (1985) 130–136.

[24] K. Klamroth, *Single-Facility Location Problems with Barriers* (Springer-Verlag, New York, 2002).
[25] E. J. v. Leeuwen, Optimization and approximation on systems of geometric objects, thesis, Universiteit van Amsterdam (2009).
[26] N. Mustafa and S. Ray, Improved results on geometric hitting set problems, *Discrete and Computational Geometry* **44**(4) (2010) 883–895, 0179-5376.
[27] P. Slavik, A tight analysis of the greedy algorithm for set cover, *Journal of Algorithms* **25**(2) (1997) 237–254.
[28] L. A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem, *Combinatorica* **2** (1982) 385–393.
[29] H. Xiaofeng, C. Xiang, E. L. Lloyd and S. Chien-Chung, Deploying directional sensor networks with guaranteed connectivity and coverage, *Sensor, Mesh and Ad Hoc Communications and Networks,* 2008. *SECON '*08. 5*th Annual IEEE Communications Society Conference on*, pp. 153–160.