Discrete Comput Geom 33:717–729 (2005)
DOI: 10.1007/s00454-004-1108-4

Discrete & Computational
Geometry

Covering Things with Things*

Stefan Langerman[1] and Pat Morin[2]

[1]Département d'informatique, Université Libre de Bruxelles,
Avenue Franklin D. Roosevelt 50, B-1050 Bruxelles, Belgium
stefan.langerman@ulb.ac.be

[2]School of Computer Science, Carleton University,
1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6
morin@cs.carleton.ca

Abstract. An abstract NP-hard covering problem is presented and fixed-parameter tractable algorithms for this problem are described. The running times of the algorithms are expressed in terms of three parameters: n, the number of elements to be covered, k, the number of sets allowed in the covering, and d, the combinatorial dimension of the problem. The first algorithm is deterministic and has a running time of $O'(k^{dk}n)$. The second algorithm is also deterministic and has a running time of $O'(k^{d(k+1)} + n^{d+1})$. The third is a Monte-Carlo algorithm that runs in time $O'(k^{d(k+1)} + c2^d k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor} n \log n)$ and is correct with probability $1 - n^{-c}$. Here, the O' notation hides factors that are polynomial in d. These algorithms lead to fixed-parameter tractable algorithms for many geometric and non-geometric covering problems.

1. Introduction

This paper pertains to the parametrized complexity of some geometric covering problems, many of which are NP-hard. A prototype for these problems is the following HYPERPLANE-COVER problem: Given a set S of n points in $\mathbb{R}^d$, does there exist a set of k hyperplanes such that each point of S is contained in at least one of the hyperplanes? It it known that HYPERPLANE-COVER is NP-hard, even in $\mathbb{R}^2$ (i.e., covering points with lines) [17].

* This research took place while both authors were visitors at the School of Computer Science of McGill University and was partly funded by NSERC, MITACS, FCAR and CRM. The first author is chargé de recherches du FNRS.

Parametrized complexity [6] is one of the recent approaches for finding exact solutions to NP-hard problems. In a typical application of this approach, one tries to identify one or more natural (usually numerical) parameters of the problem which, if small, make the problem tractable.

With many problems, such as deciding if a graph with n vertices contains a clique of size k, it is easy to devise algorithms with running time of the form $O(n^k)$, so the problem is polynomial for any constant k. While this is somewhat helpful, it is not completely satisfying since the degree of the polynomial grows quickly as k grows. Fixed-parameter tractable algorithms have more strict requirements. Namely, the running time must be of the form $O(f(k)n^c)$, where f is an arbitrarily fast growing function of k, and c is a constant not depending on k or n.

A typical example is the VERTEX-COVER problem: Given a graph $G = (V, E)$ with n vertices, does G have a subset $V' \subseteq V$ of k vertices such that every edge is incident to at least one vertex of V'? After much work on this problem, its running time has been reduced to $O(1.2852^k + kn)$ which, with current computing technology, allows for the efficient solutions of problems with values of k as large as 200 and arbitrarily large n [5].

The current article is an attempt to apply the general techniques of parametrized complexity to geometric NP-hard problems. Indeed, many geometric problems have natural parameters that can be used to limit their complexity. The most common such parameter is the geometric dimension of the problem.

This has been observed, for example, in the study of linear programming in R^d where the parameter is the dimension d. For this problem, there exists an algorithm with running time, e.g., $O(d^2n + e^{O(\sqrt{d}\ln d)})$ [16], so the problem is fixed-parameter tractable. Another remarkable example of this phenomenon is *k-piercing of rectangles*: Given a set of n rectangles in the plane, is there a set of k points such that each rectangle contains at least one of the points? This problem has $O(n \log n)$ time algorithms for $k = 1, \ldots, 5$ [14], [18], [20], [21], which makes it believable that it could admit an algorithm with a running time of the form $O(f(k)n \log n)$ for arbitrary values of k. For now, no such result is known, and the best known algorithm has a running time of $n^{\Omega(k)}$.

In this paper we define an abstract set covering problem called DIM-SET-COVER that includes HYPERPLANE-COVER and many similar problems. As part of this definition, we introduce a new notion of geometric set system hierarchy, in many points similar to the range spaces of Vapnik and Chervonenkis [22], along with a concept of dimension for each set system, which is always at least as large as its VC-dimension. We then give *fixed-parameter tractable* algorithms for DIM-SET-COVER. These are algorithms that have running times of the form $O(f(k, d)n^c)$ where c is a constant, d is the combinatorial dimension of the problem, and $f(k, d)$ is an arbitrarily fast growing function of k and d.

We give three algorithms for DIM-SET-COVER. The first is deterministic and runs in $O'(k^{dk}n)$ time. The second is deterministic and runs in $O'(k^{d(k+1)} + n^{d+1})$ time. The third is a Monte-Carlo algorithm that runs in $O'(k^{d(k+1)} + c2^d k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor} n \log n)$ time and returns a correct answer with probability at least $1 - n^{-c}$. Here and throughout, the O' notation is identical to the standard O notation except that it hides factors that are polynomial in d.

The remainder of the paper is organized as follows. In Section 2 we introduce the DIM-SET-COVER problem. In Section 3 we give a deterministic algorithm for DIM-SET-COVER. In Section 4 we present two more algorithms for DIM-SET-COVER. In Section 5

we describe several covering problems that can be expressed in terms of DIM-SET-COVER. Finally, in Section 6 we summarize and conclude with open problems.

2. Abstract Covering Problems

In this section we present the abstract covering problem DIM-SET-COVER that models many geometric and non-geometric covering problems. Throughout this section we illustrate all concepts using the HYPERPLANE-COVER example. Since these illustrations are deviations from the main text, we denote them with vertical bars.

Let U be a (possiblity infinite) universe and let $S \subseteq U$ be a *ground set* of size $n < \infty$. The universe U is covered by a set R of subsets of U. The goal of any covering problem is, given S, R, and an integer k, to find $r_1, \ldots, r_k \in R$ such that $S \subseteq \bigcup_{i=1}^{k} r_i$ or report that no such $r_1, \ldots, r_k$ exist. In our (geometric) setting, R is partitioned into d sets $R_0, \ldots, R_{d-1}$ with the property defined below. For convenience, we also define $R_{-1} = \emptyset$ and $R_d = \{U\}$.

In our HYPERPLANE-COVER example, U is $\mathbb{R}^d$ and S is a set of n points in $\mathbb{R}^d$. We say that an *i-flat* is the affine hull of $i+1$ affinely independent points [19]. The set R_i, $0 \leq i \leq d$, is the set of all i-flats in $\mathbb{R}^d$.

Given a set $A \subseteq U$ we define the *dimension* of A as

$$\dim(A) = \min\{i : \exists r \in R_i \text{ such that } A \subseteq r\}$$

and we define the *cover* of A as

$$\text{cover}(A) = \{r \in R_{\dim(A)} \text{ such that } A \subseteq r\}.$$

Since $R_d = \{U\}$, it is clear that $\dim(A)$ and $\text{cover}(A)$ are well defined and $\text{cover}(A)$ is non-empty for any $A \subseteq U$. We require that the sets $R_0, \ldots, R_{d-1}$ also satisfy the following property:

Property 1 (Intersection Reduces Dimension). For any $r_1 \in R_i$ and $r_2 \in R_j$, $0 \leq i, j \leq d$, such that $r_1 \not\subseteq r_2$ and $r_2 \not\subseteq r_1$, $\dim(r_1 \cap r_2) < \min\{i, j\}$.

In our HYPERPLANE-COVER example, the dimension $\dim(A)$ of a set A is the dimension of the affine hull of A. The set $\text{cover}(A)$ is a singleton containing the unique $\dim(A)$-flat that contains A. Property 1 says that the intersection of an i-flat and a j-flat, neither of which contains the other, is an l-flat for some $l < \min\{i, j\}$. (We consider the empty set to be a (-1)-flat.)

A *k-cover* of S is a set $\{r_1, \ldots, r_k\} \subseteq R$ such that $S \subseteq \bigcup_{i=1}^{k} r_i$.

Definition 1 (DIM-SET-COVER). Given S, R, and $R_0, \ldots, R_d$ having Property 1, the DIM-SET-COVER problem is that of determining whether S has a k-cover.

Property 1 has some important implications. The following lemma says that cover(A) really only contains one interesting set.

Lemma 1. *For any $A \subset U$ and any finite $S \subseteq U$ there is a* set(A) $\in$ cover(A) *such that $r \cap S \subseteq$ set(A) $\cap S$ for all $r \in$ cover(A).*

Proof. Suppose, by way of contradiction, that there is no such set in cover(A). Then two sets $r_1, r_2 \in$ cover(A) such that $r_1 \cap S \not\subseteq r_2 \cap S$ and $r_2 \cap S \not\subseteq r_1 \cap S$ must exist. Then it must be that $r_1 \not\subseteq r_2$ and $r_2 \not\subseteq r_1$ so, by Property 1, $\dim(r_1 \cap r_2) < \dim(A)$. However, this is a contradiction since $A \subseteq r_1 \cap r_2$. □

> In our HYPERPLANE-COVER example, we could also have defined R_i as the set of all point sets in $\mathbb{R}^d$ whose affine hull has dimension i. Then Lemma 1 says that for any $A \subseteq \mathbb{R}^d$ whose affine hull has dimension i, there exists one i-flat F containing every point set whose affine hull has dimension i and that contains A.

The following lemma says that we can increase the dimension of A by adding a single element to A.

Lemma 2. *For any $A \subseteq U$ and $p \in U \backslash$set(A), $\dim(A \cup \{p\}) > \dim(A)$.*

Proof. By definition, $\dim(A \cup \{p\}) \geq \dim(A)$. Suppose therefore, by way of contradiction, that $\dim(A \cup \{p\}) = \dim(A)$. Then there is a set $r' \in R_{\dim(A)}$ that contains $A \cup \{p\}$. However, then $r' \in$ cover(A) and, by Lemma 1, $r' \subseteq$ set(A), contradicting $p \in U \backslash$set(A). □

> In our HYPERPLANE-COVER example, Lemma 2 implies that if we have a set A whose affine hull is an i-flat and take a point p not contained in the affine hull of A then $A \cup \{p\}$ is not contained in any i-flat, i.e., the dimension of $A \cup \{p\}$ is greater than the dimension of A.

The next lemma says that for any $A \subseteq S$, set(A) has a *basis* consisting of at most $\dim(A) + 1$ elements.

Lemma 3. *For any $r \in R$ such that $r =$ set(A) for some $A \subseteq U$, there exists a basis $A' \subseteq A$ such that $r =$ set(A') and $|A'| \leq \dim(A) + 1$.*

Proof. Any such set r can be generated as follows: Initially, set $A' \leftarrow \emptyset$, so that $\dim(A') = -1$. While, set(A') $\neq$ set(A), repeatedly add an element of $A \backslash$set(A') to A'. By Lemma 2, each such addition increases the dimension of A' by at least one, up to a maximum of $\dim(A)$. Therefore, A' contains at most $\dim(A) + 1$ elements and set(A') $= r$, as required. □

> In our HYPERPLANE-COVER example, Lemma 3 is equivalent to saying that any i-flat is the affine hull of some set of $i + 1$ points.

Throughout this paper we assume, usually implicitly, that $2 \leq k < n$, otherwise the problem is not very interesting.

The sets U, R, and $R_0, \ldots, R_{d-1}$ may be infinite, so they are usually represented implicitly. We assume that any algorithm for DIM-SET-COVER accesses these sets using two operations. (1) Given a set $r \in R_i$ and an element $p \in S$, the algorithm can query if p is contained in r. Such a query takes $O'(1)$ time. (2) Given a set $A \subseteq U$, the algorithm can determine $\text{set}(A)$ and $\dim(A)$ in $O'(|A|)$ time. It follows that the only sets accessible to an algorithm are those sets $r \in R$ such that $r = \text{set}(S')$ for some $S' \subseteq S$. We call such sets *accessible sets*. Throughout the remainder of this paper we consider only k-covers that consist of accessible sets.

> In our HYPERPLANE-COVER example, we can determine in $O(d) = O'(1)$ time if a point is contained in an i-flat. Given $A \subseteq \mathbb{R}^d$, we can compute the affine hull of A in $O'(|A|)$ time by reducing an $|A| \times d$ matrix to row-echelon form.

3. A Deterministic Algorithm

Next we give a deterministic fixed-parameter tractable algorithm for DIM-SET-COVER. The algorithm is based on the *bounded search tree* method [6]. The algorithm works by trying to partition S into sets $S_1, \ldots, S_k$ such that $\dim(S_i) < d$ for all $1 \leq i \leq k$.

Initially, the algorithm sets $S' \leftarrow S$ and $S_1 \leftarrow S_2 \leftarrow \cdots \leftarrow S_k \leftarrow \emptyset$. The algorithm always maintains the invariants that (1) $S', S_1, \ldots, S_k$ form a partition of S and (2) $\dim(S_i) < d$ for all $1 \leq i \leq k$. At the beginning of every recursive invocation the algorithm checks if S' is empty and, if so, outputs *yes*, since $\text{set}(S_1), \ldots, \text{set}(S_k)$ is a k-cover of S. Otherwise, the algorithm chooses an element $p \in S'$. For each $1 \leq i \leq k$, if $\dim(S_i \cup \{p\}) < d$ the algorithm calls itself recursively on $S' \backslash \text{set}(S_i \cup \{p\})$, and $S_1, \ldots, S \cap \text{set}(S_i \cup \{p\}), \ldots, S_k$. If none of the recursive calls gives a positive result the algorithm returns *no*. This is described by the following pseudocode:

BST-DIM-SET-COVER($S', S_1, \ldots, S_k$)

```
1: if S' = ∅ then
2:    output yes and quit
3: else
4:    choose p ∈ S
5:    for i = 1, ..., k do
6:       if dim(S_i ∪ {p}) < d then
7:          BST-DIM-SET-COVER(S'\set(S_i∪{p}), S_1, ..., S∩set(S_i∪{p}), ..., S_k)
8:       end if
9:    end for
10: end if
11: output no
```

Theorem 1. *Algorithm* BST-DIM-SET-COVER *correctly solves the* DIM-SET-COVER *problem in* $O'(k^{dk}n)$ *time.*

Proof. We begin by proving the correctness of the algorithm. To do this we consider a *restriction* of the problem. A RESTRICTED-SET-COVER instance is a $(k+1)$-tuple $(S, S_1, \ldots, S_k)$ where $S_i \subseteq S$ and $\dim(S_i) < d$ for each $1 \leq i \leq k$. A solution to the instance consists of a k-cover $\{r_1, \ldots, r_k\}$ of S such that $S_i \subseteq r_i$ for all $1 \leq i \leq k$. The *degree of freedom*, of a RESTRICTED-SET-COVER instance is defined as $\mathrm{DF}(S_1, \ldots, S_k) = (d-1)k - \sum_{i=1}^{k} \dim(S_i)$. Note that RESTRICTED-SET-COVER is equivalent to DIM-SET-COVER when $S_1 = \cdots = S_k = \emptyset$, in which case $\mathrm{DF}(S_1, \ldots, S_k) = dk$.

We claim that a call to BST-DIM-SET-COVER$(S, S_1, \ldots, S_k)$ correctly solves the RESTRICTED-SET-COVER instance $(S, S_1, \ldots, S_k)$. To prove this, we use induction on $\mathrm{DF}(S_1, \ldots, S_k)$. If $\mathrm{DF}(S_1, \ldots, S_k) = 0$ then $\dim(S_i) = d-1$ for all $1 \leq i \leq n$, and, by Lemma 2, the RESTRICTED-SET-COVER instance has a solution if and only if S' is empty. Since line 1 of the algorithm checks this condition, the algorithm is correct in this case.

Next suppose that $\mathrm{DF}(S_1, \ldots, S_k) = m > 0$. If S' is empty then $S_1, \ldots, S_k$ form a partition of S and line 1 ensures that the algorithm correctly answers *yes*. Otherwise, by construction, any $p \in S'$ is not contained in any set(S_i). If the answer to the RESTRICTED-SET-COVER instance is yes then, in a restricted k-cover $\{r_1, \ldots, r_k\}$, p is contained in some set r_i. Therefore, the RESTRICTED-SET-COVER instance $(S, S_1, \ldots, S_i \cup \{p\}, \ldots, S_k)$ also has a solution. On the other hand, if the correct answer is no, then none of the restrictions $(S, S_1, \ldots, S_i \cup \{p\}, \ldots, S_k)$ for any $1 \leq i \leq k$ have a solution. By Lemma 2, $\mathrm{DF}(S_1, \ldots, S_i \cup \{p\}, \ldots, S_k) < m$ for all $1 \leq i \leq k$ so, by induction, a call to BST-DIM-SET-COVER$(S_1, \ldots, S_i \cup \{p\}, \ldots, S_k)$ correctly solves the RESTRICTED-SET-COVER instance $(S_1, \ldots, S_i \cup \{p\}, \ldots, S_k)$. Since the algorithm checks all these k RESTRICTED-SET-COVER instances, it must answer correctly.

Finally, we prove the running time of the algorithm. Each invocation of the procedure results in at most k recursive invocations and, by Lemma 2, each recursive call decreases the degree of freedom by at least one. Therefore, the recursion tree has at most k^{dk} leaves and $\sum_{i=0}^{dk-1} k^i = O(k^{dk-1})$ internal nodes. The work done at each leaf is $O'(k)$ and the work done at each internal node is $O'(kn)$. Therefore the overall running time is $O'(k^{dk}n)$, as required. □

4. Kernelization

In this section we give two more algorithms for DIM-SET-COVER that have a reduced dependence on k. The first algorithm is deterministic and runs in $O'(k^{d(k+1)}+n^{d+1})$ time. The second is a Monte-Carlo algorithm that runs in $O'(k^{d(k+1)} + c2^d k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor} n \log n)$ time and outputs a correct answer with high probability. Both algorithms work by reducing the given DIM-SET-COVER instance to an equivalent *kernel* instance that has size $O(k^d)$ and then solving the new instance using the BST-DIM-SET-COVER algorithm. We begin with a structural lemma.

Lemma 4. *Suppose $|r \cap S| \leq m$ for all $r \in \bigcup_{j=0}^{i-1} R_j$ and there exists an accessible set $r' \in R_i$ such that $|r' \cap S| > km$. Then any k-cover of S contains a set r'' such that $r' \subseteq r''$.*[1]

[1] The reader is reminded that we are only considering k-covers consisting of accessible sets.

Proof. By Property 1, any *accessible* set $r \in R$ that does not contain r' has $\dim(r \cap r') < i$ and therefore $|(r \cap r') \cap S| \leq m$, i.e., r contains at most m elements of $r' \cap S$. Therefore, k such sets contain at most $km < |r' \cap S|$ elements of $r' \cap S$. However, in a k-cover, all elements of $r' \cap S$ must be covered. We conclude that any k-cover must contain a set r'' such that $r' \subseteq r''$. □

Lemma 4 is the basis of our *kernelization* procedure. The procedure works by finding sets in R that contain many elements of S and grouping those elements of S together. Given a subset $S' \subseteq S$, a grouping of the DIM-SET-COVER instance S by S' creates a new DIM-SET-COVER instance as follows: (1) The elements of S' are removed from S and a new element s' is added to S. (2) For every set $r \in R$ such that $S' \subseteq r$, the element s' is added to r.

It is clear that $\dim(A)$ for any $A \subseteq S$ and Property 1 are preserved under the grouping operation, so that Property 1 holds for the new DIM-SET-COVER instance. Furthermore, Lemma 3 implies that the new element s' can be represented as a list of at most d elements of S. Thus, operations on grouped instances of DIM-SET-COVER can be done with a runtime that is within a factor of d of non-grouped instances, i.e., the overhead of working with grouped instances is $O'(1)$.

The following lemma shows that, if the group is chosen carefully, the new instance is equivalent to the old one.

Lemma 5. *Suppose $|r \cap S| \leq m$ for all $r \in \bigcup_{j=0}^{i-1} R_j$ and there exists an accessible set $r' \in R_i$ with size $|r' \cap S| > km$. Then the grouping of S by $S' = r' \cap S$ has a k-cover if and only if S has a k-cover.*

Proof. If S has a k-cover $\{r_1, \ldots, r_k\}$ then, by Lemma 4, there exists an r_j such that $r' \subseteq r_j$. Consider the sets $r'_1, \ldots, r'_k$ in the grouped instance that correspond to the sets $r_1, \ldots, r_k$, respectively. For each $1 \leq i \leq k$, $r_i \subseteq r'_i$. Therefore, $S \backslash S' \subseteq S \subseteq \bigcup_{i=1}^{k} r'_i$. Furthermore, r'_j contains s'. Therefore $\{r'_1, \ldots, r'_k\}$ is a k-cover for the grouped instance.

On the other hand, if the grouping of S by S' has a k-cover $\{r'_1, \ldots, r'_k\}$, then the corresponding sets $r_1, \ldots, r_k$ in the original instance also form a k-cover for S. This is due to the facts that, for each $1 \leq i \leq k$, $r_i = r'_i \backslash \{s'\}$, $s' \notin S$, and $s' \in r'_i$ implies $S' \subseteq r_i$. □

The following procedure is used to reduce an instance of DIM-SET-COVER involving a set S of size n into a new instance with size at most k^d.

KERNELIZE(S)

1: **for** $i = 0, \ldots, d-1$ **do**
2: **while** R_i contains an accessible set r such that $|r \cap S| > k^i$ **do**
3: group S by $r \cap S$
4: **end while**
5: **end for**

Lemma 6. *A call to* KERNELIZE(S) *produces a new instance of* DIM-SET-COVER *that has a k-cover if and only if the original instance has a k-cover.*

Proof. By the time the procedure considers set R_i in lines 2–4, every r in $\bigcup_{j=0}^{i-1} R_j$ has size $|r \cap S| \leq k^{i-1}$. Therefore, by Lemma 5, any grouping operation performed in line 3 results in a DIM-SET-COVER instance that has a k-cover if and only if the original DIM-SET-COVER instance has a k-cover. □

After a call to KERNELIZE(S) we get a new instance of DIM-SET-COVER for which $|r \cap S| \leq k^{d-1}$ for all $r \in R$. One consequence of this is that, if the new instance has more than k^d elements that need to be covered, then we can be immediately certain that it does not have a k-cover. Therefore, the instance that we have left to solve has size $n' \leq k^d$ and can be solved in time $O'(k^{d(k+1)})$ using the BST-DIM-SET-COVER algorithm.

Lemma 7. *The* DIM-SET-COVER *problem can solved in* $O'(k^{d(k+1)} + K(n, k, d))$ *time where* $K(n, k, d)$ *is the running time of the* KERNELIZE *procedure.*

All that remains is to find an efficient implementation of the KERNELIZE procedure. Lemma 3 implies that any accessible set in R_i can be generated as set(S') where $S' \subseteq S$ has size at most $i + 1$. It follows that all the accessible sets in R_i can be generated in $O(n^{i+1})$ time, and each one can be checked in $O(n)$ time. This gives us the following brute-force result.

Theorem 2. *The* DIM-SET-COVER *problem can be solved in* $O'(k^{d(k+1)} + n^{d+1})$ *time by a deterministic algorithm.*

Although Theorem 2 implies a faster algorithm for some values of k, d, and n, it is not entirely satisfactory. In fact, if we parametrize by k and d then it does not even satisfy the definition of fixed-parameter tractability since d appears in the exponent of n. To obtain a faster algorithm we make use of randomization.

Define a *heavy covering set* as an accessible set $r \in R_i$ such that $|r \cap S| > n/2k^{d-i}$. Consider the following alternative implementation of KERNELIZE:

HALVING-KERNELIZE(S)

```
1: while |S| > 2k^d do
2:   n ← |S|
3:   for i = 0, ..., d − 1 do
4:     while R_i contains a heavy covering set r do
5:       group S by r ∩ S
6:     end while
7:   end for
8:   if |S| > n/2 then
9:     output no
10:  end if
11: end while
```

It is easy to verify, using Lemma 5, that each grouping operation results in an instance of DIM-SET-COVER that has a k-cover if and only if the original instance has a k-cover. Now, after each iteration of the outer while loop, $|r \cap S| \leq n/2k$ for all $r \in S$. Therefore,

if $|S|$ is greater than $n/2$ we can be sure that S has no k-cover, so the algorithm only outputs *no* in line 9 when S has no k-cover. If this is not the case, then $|S|$ decreases by a factor of at least 2 during each iteration.

It seems that HALVING-KERNELIZE is no easier to implement than the original KERNELIZE procedure. However, the difference between the two is that HALVING-KERNELIZE attempts to find heavy covering sets, which contain relatively large fractions of S. This helps, because if we choose $i+1$ elements of S at random, there is a good chance that they will all belong to a heavy covering set in R_i, if a heavy covering set exists.

Suppose that $|r \cap S| \leq n/(2k^{d-j})$ for all $r \in R_j$, $j < i$. Let $r \in R_i$ be a heavy covering set and consider the following experiment. Initially we set our *sample* S' equal to the empty set. We then repeatedly choose an element p from S at random and add it to our sample S'. If p is contained in $(r \cap S) \backslash \text{set}(S')$ we are successful and we continue, otherwise we are unsuccessful and we stop. The experiment ends when we are either unsuccessful or $\text{set}(S') = r$. In the former case we call the experiment a *failure*. In the latter case we call it a *success*.

If $\dim(S') = -1$, i.e., $S' = \emptyset$, then the probability that we are successful is at least

$$\frac{|r \cap S|}{n} \geq \frac{1}{2k^{d-i}}.$$

Otherwise, if $0 \leq \dim(S') = j < i$, then the probability that we are successful is at least

$$\frac{|r \cap S| - |\text{set}(S') \cap S|}{n} \geq \frac{1}{2k^{d-i}} - \frac{1}{2k^{d-j}} = \frac{k^i - k^j}{2k^d},$$

and each successful step increases $\dim(S')$ by at least one. Therefore, the probability that the entire experiment is a success is at least

$$\begin{aligned}
p_i &\geq \frac{1}{2k^{d-i}} \times \prod_{j=0}^{i-1} \left(\frac{k^i - k^j}{2k^d} \right) && (1)\\
&= \frac{1}{2^{i+1}k^{d+di-i}} \times \prod_{j=0}^{i-1} \left(k^i - k^j \right) && (2)\\
&= \frac{k^{i^2}}{2^{i+1}k^{d+di-i}} \times \prod_{j=0}^{i-1} \left(1 - k^{j-i} \right) && (3)\\
&= \frac{1}{2^{i+1}k^{d+di-i-i^2}} \times \prod_{j=1}^{i} \left(1 - k^{-j} \right) && (4)\\
&\geq \frac{1}{2^{i+1}k^{d+di-i-i^2}} \times \prod_{j=1}^{\infty} \left(1 - k^{-j} \right) && (5)\\
&\geq \frac{1}{2^{i+3}k^{d+di-i-i^2}} && (6)\\
&\geq \frac{1}{2^{d+2}k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor}}. && (7)
\end{aligned}$$

Inequality (6) follows from Euler's pentagonal number theorem [7, Chapter 16] (see [2]). If we repeat the above experiment x times, then the probability that all of the experiments

are failures is at most

$$(1 - p_i)^x.$$

Setting $x = 2^{d+2}k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor}$, this probability is less than $1/2$. If we repeat this procedure $cx \log n$ times, the probability that all experiments are failures is no more than n^{-c}. Thus, we have an algorithm that runs in $O'(cxn \log n)$ time and finds r with probability at least $1 - n^{-c}$.

Lemma 3 implies that the total number of accessible sets in R_i, and hence the total number of heavy covering sets in R_i is at most $\binom{n}{i+1} \leq n^{i+1}$. Thus, if we choose $c = i + 1 + c'$ then the probability that there exists a heavy covering set in R_i that is not found by repeating the above sampling experiment $cx \log n$ times is at most $n^{-c'}$. Therefore, one execution of lines 4–6 of the HALVING-KERNELIZE algorithm can be implemented to run in $O'(c'x|S| \log n)$ time and fails with probability at most $n^{-c'}$. Since $|S|$ is halved during each iteration of the outer loop, the entire algorithm runs in time $O'(c'xn \log n)$ and the algorithm is correct with probability at least $1 - n^{-c'} d \log n$. Choosing c' large enough yields our main result on randomized algorithms.

Theorem 3. *There exists a Monte-Carlo algorithm for the* DIM-SET-COVER *problem that runs in* $O'(k^{d(k+1)} + c2^d k^{\lceil (d+1)/2 \rceil \lfloor (d+1)/2 \rfloor} n \log n)$ *time and answers correctly with probability at least* $1 - n^{-c}$.

5. Applications

In this section we present a number of covering problems, both geometric and non-geometric, that can be modeled as instances of DIM-SET-COVER and hence solved by the algorithms of the previous two sections. For the geometric applications, the value of d is closely related to the dimension of the geometric space.

5.1. *Covering Points with Hyperplanes and Vice Versa*

Given a set S of n points in $\mathbb{R}^d$, does there exist a set of k hyperplanes such that each point of S is contained in at least one hyperplane?

This problem has received considerable attention in the literature and appears to be quite difficult. It is known to be NP-hard even when $d = 2$ [17]. The optimization problem of finding the smallest value of k for which the answer is yes has recently been shown to be APX-hard [3], [13], so unless P = NP there does not exist a $(1 + \varepsilon)$-approximation algorithm. Algorithms for restricted versions and variants of this problem have been considered by Agarwal and Procopiuc [1] and Hassin and Megiddo [9].

We have seen, in our running example, that this problem can be modeled as follows: $U = \mathbb{R}^d$ and, for each $0 \leq i \leq d - 1$, R_i is the set of i-flats embedded in $\mathbb{R}^d$. To see that this fits into our model, observe that the intersection of an i-flat and a j-flat, neither of which contains the other, is an l-flat, for some $l < \min\{i, j\}$ so this system satisfies Property 1. Thus, this is a DIM-SET-COVER instance in which the parameter d is equal to the dimension of the underlying Euclidean space.

In a dual setting, we are given a set S of n hyperplanes in $\mathbb{R}^d$ and asked if there exists a set of k points such that each hyperplane in S contains at least one of the points. By a standard point–hyperplane duality, this problem is equivalent to the previous problem and can be solved as efficiently.

At this point we remark that in the plane, i.e., covering points with lines, it is possible to achieve a deterministic algorithm that performs slightly better than our randomized algorithm.[2] This is achieved by replacing our randomized procedure for finding heavy covering sets with an algorithm of Guibas et al. [8] that can find all lines containing at least m points of S in time $O((n^2/m)\log(n/m))$. Using this yields a deterministic algorithm with running time $O(k^{2k+2} + kn)$.

5.2. *Covering Points with Spheres*

Given a set S of n points in $\mathbb{R}^d$, does there exist a set of k hyperspheres such that each point in S is on the surface of at least one of the hyperspheres?

For this problem we take $R_{d+1} = U = \mathbb{R}^d$. The set R_i, $0 \leq i \leq d$ consists of all i-spheres, so that R_0 consists of points, R_1 consists of pairs of points, R_2 consists of circles, and so on. Again, the intersection of an i-sphere and a j-sphere, neither of which contains the other, is an l-sphere for some $l < \min\{i, j\}$, so this system satisfies Property 1. This yields an instance of DIM-SET-COVER whose dimension is one greater than that of the underlying Euclidean space.

5.3. *Covering Points with Polynomials*

Given a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n points in $\mathbb{R}^2$ does there exist a set of k polynomial functions $f_1, \ldots, f_k$, each with maximum degree d, such that for each $1 \leq i \leq n$, $y_i = f_j(x_i)$ for some $1 \leq j \leq k$? In other words, is every point in S contained in the graph of at least one of the functions?

This problem is a generalization of the problem of covering points with lines. As such, it is NP-hard and APX-hard, even when $d = 1$.

We say that two points in $\mathbb{R}^2$ are x-distinct if they have distinct x-coordinates. For this problem, R_0 is the set of all points, R_1 is the set of all x-distinct pairs of points, R_2 is the set of all x-distinct triples of points, and so on, until R_{d+1} which is the set of all degree d polynomials. Since there exists a degree d polynomial that contains any set of $d + 1$ or fewer x-distinct points, each set in R corresponds to points that can be covered by a degree d polynomial (possibly with some coefficients set to zero).

The intersection of a finite set of i points and a (possibly infinite) set of $j \geq i$ points, neither of which contains the other, results in a set of $l < \min\{i, j\}$ points. The intersection of two non-identical degree d polynomials is a set of at most d points. Therefore, the above system of sets satsifies Property 1 and can be solved with our algorithms.

---

[2] This result can be generalized to $\mathbb{R}^d$ when the point set S is *restricted*, i.e., no $i + 1$ elements of S lie on a common i-flat, for any $0 \leq i < d - 1$. However, since any input that has a k-cover necessarily has many points on a common $(d - 1)$-flat, this does not seem like a reasonable assumption.

5.4. *Covering by Sets with Intersection at Most d*

Given a set S of size n and a set C containing subsets of S such that no two elements of C have more than d elements in common, is there a set of k elements in C whose union contains all the elements of S?

This problem is a special case of the classic SET-COVER problem, which was one of the first problems shown to be NP-hard [12]. The version when $d = 1$ is a generalization of covering points by lines and is therefore APX-hard. In fact, finding an $o(\log n)$ approximation for the corresponding optimization problem is not possible unless NP $\subseteq$ ZTIME$(n^{O(\log \log n)})$ [13]. On the other hand, Johnson [11] shows that, even with no restrictions on d, an $O(\log n)$-approximation can be achieved with a simple greedy algorithm.

To model this problem as a DIM-SET-COVER instance, we let R_i, $0 \leq i < d$, be all subsets of S of size $i + 1$ that are contained in one element of C and we let $R_d = C$. It is easy to verify that these sets satisfy Property 1 and hence we have an instance of DIM-SET-COVER that can be solved with our algorithms. If the sets are given explicitly, the Kernelize procedure runs in $O(n')$ time where $n' = \sum_{r \in C} |r|$ is the input size. Thus, for this case we obtain an algorithm that runs in time $O(n' + k^{d(k+1)})$.

6. Conclusions

We have presented an abstract covering problem on sets that includes many NP-hard geometric and non-geometric covering problems. We have obtained fixed-parameter tractable algorithms for this problem that yield fixed-parameter tractable algorithms for the concrete problems.

Our third algorithm is a Monte-Carlo algorithm that relies on randomization to find heavy covering sets. It seems feasible that the algorithm could be derandomized with the use of ε-nets [4], [10], [15]. Indeed, setting $\varepsilon = 1/2k^{d-i}$ guarantees that an ε-net contains at least one element from each heavy covering set in R_i. However, we require a net that contains a basis for every heavy covering set in R_i. Perhaps such a net could be obtained by repeated applications of ε-nets.

One type of problem that cannot be modeled by our abstraction is the class of *k-piercing* problems, where we are given a set of objects, say rectangles in the plane, and asked to find a set of k points such that each rectangle contains at least one point. The main difficulty here seems to be that the intersection of two rectangles is another rectangle, so the resulting system does not satisfy Property 1. Nevertheless, $O(n \log n)$ time algorithms exist for $k = 1, \ldots, 5$ [14], [18], [20], [21], so it is plausible that fixed-parameter tractable algorithms for piercing problems exist.

References

1. P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. In *Proceedings of the* 9*th ACM–SIAM Symposium on Discrete Algorithms* (*SODA* 1998), pages 658–667, 1998.
2. G. E. Andrews. *The Theory of Partitions*. Addison-Wesley, Reading, MA, 1976.

3. B. Brodén, M. Hammar, and B. J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *Proceedings of the* 13*th Canadian Conference on Computational Geometry*, pages 45–48, 2001.
4. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Journal of Algorithms*, 21(3):579–597, November 1996.
5. J. Chen, I. Kanj, and W. Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41:280–301, 2001.
6. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999.
7. L. Eulero. *Introductio in Analysin Infinitorum*. Tomus Primus, Lausanne, 1748.
8. L. J. Guibas, M. H. Overmars, and J.-M. Robert. The exact fitting problem for points. *Computational Geometry*: *Theory and Applications*, 6:215–230, 1996.
9. R. Hassin and N. Megiddo. Approximation algorithms for hitting objects by straight lines. *Discrete Applied Mathematics*, 30:29–42, 1991.
10. D. Haussler and E. Welzl. ε-Nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
11. D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
12. R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 85–103. Plenum, New York, 1972.
13. V. S. A. Kumar, S. Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *Proceedings of the* 27*th International Colloquium on Automata*, *Languages and Programming*, pages 624–635, 2000.
14. C. Makris and A. Tsakalidis. Fast piercing of iso-oriented rectangles. In *Proceedings of the* 9*th Canadian Conference on Computational Geeometry*, pages 217–222, 1997.
15. J. Matoušek. Approximations and optimal geometric divide-and-conquer. *Journal of Computer and System Sciences*, 50(2):203–208, April 1995.
16. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.
17. N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1:194–197, 1982.
18. D. Nussbaum. Rectilinear p-piercing problems. In *ISSAC '*97. *Proceedings of the* 1997 *International Symposium on Symbolic and Algebraic Computation*, *July* 21–23, 1997, *Maui*, *Hawaii*, pages 316–323, 1997.
19. F. P. Preparata and M. I. Shamos. *Computational Geometry*: *An Introduction*. Springer-Verlag, New York, 1985.
20. M. Segal. On piercing of axis-parallel rectangle and rings. *International Journal of Computational Geometry and Applications*, 9(3):219–233, 1999.
21. M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry* (*ISG '*96), pages 122–132, 1996.
22. V. N. Vapnik and A. Y. A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Received September 28, 2003, *and in revised form March* 5, 2004. *Online publication July* 9, 2004.