

Almost Optimal Set Covers in Finite VC-Dimension*

H. Brönnimann¹ and M. T. Goodrich²

¹ Department of Computer Science, Princeton University,
Princeton, NJ 08544, USA
hbr@cs.princeton.edu

² Department of Computer Science, Johns Hopkins University,
Baltimore, MD 21218, USA
goodrich@cs.jhu.edu

Abstract. We give a deterministic polynomial-time method for finding a set cover in a set system (X, \mathcal{R}) of dual VC-dimension d such that the size of our cover is at most a factor of $O(d \log(dc))$ from the optimal size, c . For constant VC-dimensional set systems, which are common in computational geometry, our method gives an $O(\log c)$ approximation factor. This improves the previous $\Theta(\log|X|)$ bound of the greedy method and challenges recent complexity-theoretic lower bounds for set covers (which do not make any assumptions about the VC-dimension). We give several applications of our method to computational geometry, and we show that in some cases, such as those arising in three-dimensional polytope approximation and two-dimensional disk covering, we can quickly find $O(c)$ -sized covers.

1. Introduction

A *set system* (X, \mathcal{R}) is a set X along with a collection \mathcal{R} of subsets of X , which are sometimes called *ranges* [25]. Such entities have also been called *hypergraphs* and *range spaces* in the computational geometry literature (e.g., see [5], [10]–[16], [20], [24], [25], [34]–[36], and [38]–[41]), and they can be used to model a number of interesting computational geometry problems.

* The first author was supported in part by NSF Grant CCR-90-02352 and Ecole Normale Supérieure. The second author's research was supported by the NSF and DARPA under Grant CCR-8908092, and by the NSF under Grants IRI-9116843 and CCR-9300079.

There are a host of NP-hard problems defined on set systems, with one of the chief such problems being that of finding a *set cover* of minimum size (e.g., see [21] and [23]), where a set cover is a subcollection $C \subseteq \mathcal{R}$ whose union is X and the size of C is simply the number of sets in C . A related (in fact, dually equivalent) problem is that of finding a *hitting set* of smallest size, where a hitting set is a subset $H \subseteq X$ such that H has a nonempty intersection with every set R in \mathcal{R} . There are a number of problems that can be reduced to these two problems, and many of these problems are formulated as computational geometry problems. Thus, our interest is in finding minimum set covers and smallest hitting sets as quickly as possible.

Unfortunately, the corresponding decision problems, SET COVER and HITTING SET, as we call them, are both NP-complete [23], [29]. Moreover, even if each element of X is contained in just two sets, the HITTING SET problem is still NP-complete [29] (and is known as VERTEX COVER). Thus, unless $P = NP$, if we desire a polynomial-time algorithm, we must content ourselves with an approximation algorithm, which, if we let c denote the size of a minimum set cover, produces a set cover of size αc for some (hopefully) small approximation factor α . The only known polynomial-time approximation algorithms for SET COVER with guaranteed performance ratios are the greedy algorithm [17], [28], [32], which achieves an approximation factor $\alpha = (1 + \ln A)$, where A denotes the size of the largest set in \mathcal{R} ; randomized parallel versions, the best given in [44] being in RNC⁴, with $\alpha = 16(1 + \ln A)$; and an algorithm of Hochbaum [26], which achieves an approximation factor $\alpha = \max_{x \in X} |\mathcal{R}_x|$, where \mathcal{R}_x denotes the set of all sets R of \mathcal{R} containing x . Note that in the former cases α can be as large as $(1 + \ln|X|)$ and in the latter α can be as large as $|\mathcal{R}|$. Thus, as worst-case bounds, the greedy algorithm achieves the best approximation factor, and Johnson [28] shows there are set systems where the greedy algorithm does indeed produce a set cover with approximation factor $\alpha = \Omega(\ln|X| + 1)$. Interestingly, Lund and Yannakakis [33] have recently shown that, unless $NP \subseteq DTIME[n^{\text{poly log } n}]$, no polynomial-time algorithm can approximate SET COVER within a factor better than $\alpha = \delta \ln|X|$ in the worst case, for any constant $\delta < \frac{1}{4}$, and unless $NP \subseteq DTIME[n^{O(\log \log n)}]$, Bellare *et al.* [4] showed that no polynomial-time deterministic algorithm can approximate SET COVER within a factor better than $\alpha = \delta \ln|X|$ in the worst case, for any constant $\delta < \frac{1}{8}$. Furthermore, there is nothing special about determinism here, as the same bounds are true under similar assumptions on the corresponding probabilistic classes (*BPTIME* instead of *DTIME*).

Our Results. In this paper we give a (deterministic) polynomial-time algorithm that finds a set cover of size within a factor $\alpha = O(d \log(dc))$ of the size c of the minimum set cover, where d stands for the VC-exponent of the dual set system (see Section 2 for definitions). Thus, for set systems with a bounded VC-exponent, we challenge the complexity-theoretic lower bounds of [4] and [33] (which make no assumptions about the VC-exponent). In Section 4 we indicate some examples on which Hochbaum's method or the greedy method perform poorly, and show that our method outperforms them on those instances.

Our algorithm, which is actually for the dual HITTING SET problem, is based upon a deterministic analogue of a randomized *natural selection* technique used by

Clarkson [18], [19], Littlestone [31], and Welzl [48]. This technique can be viewed as “algorithmic Darwinism,” for we iteratively select, in polynomial time, a small-sized subset N of X (known in the literature as an ε -net [25]) that intersects all highly weighted sets in \mathcal{R} , and, if N is not a hitting set, then we increase the weight of the elements in a set R of \mathcal{R} missed by N . By continuing this process over several “generations” we guarantee that the “fittest” elements, which belong to the optimal hitting set, are eventually included. Fortunately, the number of generational iterations we must perform is at most $O(c \log(|X|/c))$; hence, this approach yields a (relatively simple) polynomial-time algorithm.

We show the utility of our approximation algorithm by giving several applications of our method to problems in computational geometry and learning theory, including polytope approximation, geometric point-probe decision-tree construction, and disk cover. Our methods improve the running times and/or the approximation factors of previous methods. In fact, for three-dimensional polytope approximation and two-dimensional disk covering we show how to adapt our method to achieve a constant approximation factor. We also review an implementation of our approach.

The Models of Computation. It is customary in computational geometry to use the *real RAM model* [43], [46], where each memory cell can hold a real number with infinite precision (i.e., as high as needed). Polynomial-time algorithms in this model are sometimes called *strongly polynomial*, or *combinatorial*, algorithms. Due to the relevance of our results in the domain of NP-hardness, however, we have to be careful to use a model that also limits the size of the numbers it can manipulate. The model of computation we choose for our main theorem is the so-called *log-RAM model* [46], where a memory cell holds $O(\log N)$ bits (where N is the total size of the input) and the arithmetic operations performed on the numbers held in those cells are executed in constant time. Note that the complexities of our algorithms are multiplied by at most $N^{O(1)}$ when measured in the more traditional *bit model* [46] (where all the bit operations constituting an elementary arithmetic operation in the log-RAM model are accounted for separately), and therefore the running times of our algorithms are still polynomial in the size of the input in the bit model.

Nevertheless, when returning to the traditional computational geometry framework in Section 5.1, we assume the real RAM model. Before we describe our methods, however, we first review the relevant properties and definitions regarding the notion of VC-dimension.

2. Set Systems of Finite VC-Dimension

In this section we recall the basic facts about set systems of finite VC-dimension. For references, the reader is referred to the original papers by Vapnik and Červonenkis [47] (from whom have derived the initials VC) and Haussler and Welzl [25], or to the survey by Assouad [2].

Let (X, \mathcal{R}) be a given set system. Given $Y \subseteq X$, all the subsets of Y obtained as the intersection of Y and R ranging over \mathcal{R} , form a set system called the system induced by \mathcal{R} on Y , and denoted by $\mathcal{R}|_Y$. Y is *shattered* by \mathcal{R} if $\mathcal{R}|_Y = 2^Y$. (X, \mathcal{R}) is

said to have *VC-dimension* d if d is the smallest integer such that no $d + 1$ point subset $Y \subseteq X$ can be shattered. If Y is finite, it is well known [45], [47] that the number of sets of \mathcal{R}_Y is less than

$$\binom{|Y|}{0} + \dots + \binom{|Y|}{d} \leq |Y|^d,$$

where d is the VC-dimension of (X, \mathcal{R}) . We call the *VC-exponent*¹ the infimum of all numbers s such that $|\mathcal{R}_Y| = O(|Y|^s)$ for any finite subset Y of X . The aforementioned result shows that $s \leq d$, but equality does not always occur, as there are systems in which the VC-exponent is nonintegral [2]. However, the VC-dimension is finite if and only if the VC-exponent is finite as well; s never takes values in the open interval $(0, 1)$, and it is zero if and only if \mathcal{R} is finite [2] (whereas d is zero if and only if \mathcal{R} consists of one set). In particular, the VC-exponent concept only makes sense for an infinite set system, whereas the VC-dimension is more general. Consider, for the sake of an example, a common set system often arising in computational geometry applications—the *half-space* set system. In this system X is taken as a set of half-spaces in \mathbb{R}^d and \mathcal{R} is taken to be all combinatorially distinct ways of intersecting half-spaces in X by a simplex. It is well known (e.g., see [2]) that this system has a VC-dimension and a VC-exponent of d .

The dual set system (\mathcal{R}, X^*) is defined by $X^* = \{\mathcal{R}_x : x \in X\}$, where \mathcal{R}_x consists of all the sets R of \mathcal{R} that contain x . One way to look at dual systems is to consider (X, \mathcal{R}) as a (possibly infinite) boolean matrix M that has a column for each element x of X and each row corresponds to an incidence relation for a set R of \mathcal{R} , and view the dual as a transpose of this matrix. It is also well known (e.g., see [2]) that the VC-dimension of the dual (\mathcal{R}, X^*) is less than 2^{d+1} , where d is the VC-dimension of the primal (X, \mathcal{R}) .

Let (X, \mathcal{R}) be a finite² set system. If a subset $N \subseteq X$ intersects each set R of \mathcal{R} of size bigger than $\varepsilon|X|$, then we call N an ε -net [25]. As has been observed by Haussler and Welzl [25], set systems of VC-dimension d admit $(1/r)$ -nets of size $O(dr \log(dr))$. This bound has been improved by Blumer *et al.* [7] to $O(dr \log r)$ and this has been proved tight by Komlós *et al.* [30]. We can generalize this definition by putting an additive³ weight function w on 2^X . In this formulation an ε -net is required to intersect every set of weight at least $\varepsilon w(X)$. To allow for efficient computation, we need the set system to be described somewhat more efficiently than by the list of its subsets.

Definition 2.1. We say that a set system (X, \mathcal{R}) has a *subsystem oracle of degree* D if there is an algorithm which, given a finite subset $Y \subseteq X$, returns (Y, \mathcal{R}_Y) (e.g., as a boolean matrix) in time $O(|Y|^{D+1})$. It is a *witness oracle* if, for any set R of \mathcal{R}_Y , it can provide a set R' of \mathcal{R} such that $R = R' \cap Y$ in $O(|X|)$ time.

¹ This value has also gone by the names *real density* [2] and *scaffold dimension* [24].

² We could also give definitions for an infinite set system, given a probability distribution on X . The results of this paragraph would then still be valid.

³ The term additive refers to the fact that $w(Y) = \sum_{y \in Y} w(y)$, with the usual abuse of notation $w(x) = w(\{x\})$.

If (X, \mathcal{R}) has a subsystem oracle of degree D , and VC-exponent d , it is clear that $d \leq D$. Under this assumption, it has also been shown by Matoušek and coworkers [36], [10] that a $(1/r)$ -net for (X, \mathcal{R}) of size $O(dr \log(dr))$ can be found in $O(d)^{3D} r^D \log^D(rd)|X|$ time, for both the uniform and weighted cases. The scrupulous reader should verify that their algorithm also works in the log-RAM model with identical running time.

3. The Main Algorithm

The goal of this section is to prove our main theorem. Let s be a nondecreasing function, let $n = |X|$, and let $m = |\mathcal{R}|$.

Definition 3.1. A *net finder* of size s for (X, \mathcal{R}) is an algorithm A that, given r and a weight function w on X , returns a $(1/r)$ -net of size $s(r)$ for (X, \mathcal{R}) with weight w . Also, a *verifier* is an algorithm B that, given a subset $H \subseteq X$, either states (correctly) that H is a hitting set, or returns a nonempty set R of \mathcal{R} such that $R \cap H = \emptyset$.

We say that A (resp. B) runs in T_A (resp. T_B) time. That is, A run on input (X, \mathcal{R}) , w and r returns a net in $T_A(n, m, r)$ time, while B run on input (X, \mathcal{R}) and H replies in $T_B(n, m)$ time. We suppose that both T_A and T_B are (nonconstant) polynomials, so that $T(O(x)) = O(T(x))$ and $\sum_{k \leq x} T(2^k) = O(T(2^x))$ for any function T of the form $T_A(\cdot, m, r)$, $T_A(n, \cdot, r)$, $T_A(n, m, \cdot)$, $T_B(\cdot, m)$, and $T_B(n, \cdot)$. We can now state our main result:

Theorem 3.2 (Main). Let (X, \mathcal{R}) be a set system that admits both a net finder A of size s and a verifier B . Then there is an algorithm $\mathcal{A}(A, B)$ that computes a hitting set of size at most $s(4c)$, where c stands for the size of an optimal hitting set, in $O(c \log(n/c))(T_A(n, m, c) + T_B(n, m))$ time in the log-RAM model.

If X has finite VC-dimension, then we may implement the net finder using the greedy method [14], and the verifier by inspecting all of (X, \mathcal{R}) , both in polynomial time (in either the real RAM or log-RAM models). However, under standard computational assumptions, there are better implementations of the net finder and verifier.

Corollary 3.3. Let (X, \mathcal{R}) be a set system given by a witness subsystem oracle of degree D , which admits a hitting set of size c . Let d stand for the VC-exponent of (X, \mathcal{R}) , and assume that \emptyset is not in \mathcal{R} . Then a hitting set for (X, \mathcal{R}) of size $O(dc \log(dc))$ can be found in $O(nc^{D+1} \log^D(dc) \log(n/c))$ time in the log-RAM model.

Proof. It suffices to show how to implement the net finder and the verifier and show their complexity bound. However, a $(1/r)$ -net for (X, \mathcal{R}) of size $O(dr \log(dr))$ can be found in $O(d)^{3D} r^D \log^D(rd)|X|$ time, for both the uniform and weighted cases, using the algorithm of [10]. Here we assume, but it is verified in Section 3.3,

that the weights can be encoded using no more than $O(\log n)$ bits. As for the verifier, the witness subsystem oracle is simply run on H . If the oracle fails to list \emptyset as being in $\mathcal{R}_{|H}$, we may conclude that H is a hitting set. Otherwise, we ask for a witness R in \mathcal{R} of the fact that \emptyset is in $\mathcal{R}_{|H}$. Thus $R \cap H = \emptyset$, and R is nonempty as \emptyset is not in \mathcal{R} . The time spent by the verifier is $O(|H|^{D+1} + |X|) = O(|H|^D |X|)$. Plugging these bounds into Theorem 3.2 yields the corollary. \square

Remarks. (1) Even if the input (X, \mathcal{R}) is given in the form of a boolean matrix, we can construct a witness subsystem oracle in the trivial way (simply select the subsystem on the given columns and eliminate the redundant rows); this oracle does not have a running time indicated as in Definition 2.1 above, however, and the running time of the net finder of [10] deteriorates—but still remains polynomial. Therefore, it can be concluded from the Main Theorem that there is an algorithm that, given a set system (X, \mathcal{R}) by the list of its subsets, returns a hitting set of size $O(dc \log(dc))$ in time polynomial in the size of the input.

(2) If we change the definition of the VC-exponent to fit the VC-dimension concept better, by requiring that for all finite $Y \subseteq X$ we have

$$|R_{|Y}| = O\left(\binom{|Y|}{0} + \dots + \binom{|Y|}{d}\right)$$

which is also $O(|Y|/d + 1)^d$, then we can reduce the size of the hitting set to $O(dc \log c)$, as observed by Chazelle and Matoušek [15].

(3) For a particular c , the algorithm will either say that there is no hitting set of size c , or it will output a hitting set of size $O(dc \log(dc))$, which, of course, is different to saying that *there is* a hitting set of size c .

(4) The corollary can be stated with d as the *dual* VC-exponent, with an oracle for the *dual* set system, so as to yield a *set cover* of size $O(dc \log(dc))$.

(5) The running time of Corollary 3.3 compares favorably with that of the greedy method, which is $O(n^{d+1})$, when c is smaller than $n^{1-\delta}$ for some small $\delta > 0$ (on the order of $1/D$).

We now turn to the proof of our main theorem.

3.1. The Algorithm

We assume, for the time being, that we know the size c of a smallest hitting set (we show later why this is a reasonable assumption).

Our strategy can be thought of in terms of evolutionary biology, in that it is based upon a notion of “survival of the fittest.” Intuitively, we want to simulate the growth of a population where some elements are advantaged because they hit more sets than others. The idea is to put weights on the elements (initially uniformly) and use the net finder A to select a $(1/2c)$ -net of size $s(2c)$. If it does not hit a particular set R of \mathcal{R} , as returned by the verifier B on the net, we double the weights of the points

in R . Because of the definition of a hitting set, at least one point in an optimal hitting set falls in R , and has its weight doubled. Nevertheless, the property of a $(1/2c)$ -net implies that the weight of R is at most a fraction $1/2c$ of the total weight, hence the total weight does not increase too much. Therefore, we soon expect an optimal hitting set to be included in the chosen set. The next lemma shows that this is indeed the case.

Lemma 3.4. *If there is a hitting set of size c , the doubling process cannot iterate more than $4c \log(n/c)$ times, and the total weight will not exceed n^4/c^3 .*

Proof. Our proof follows arguments of Clarkson [18], [19], Littlestone [31], and Welzl [48]. Let H be a hitting set of size c . Because the set R returned by B at each iteration satisfies $w(R) \leq w(X)/2c$, the weight of X is not multiplied by more than a factor $1 + 1/2c$ in any iteration. Nevertheless, $H \cap R$ is not empty, by the definition of H . Therefore, after k iterations, if each $h \in H$ has been doubled z_h times, we have

$$w(X) \leq n \left(1 + \frac{1}{2c} \right)^k \leq ne^{k/2c}$$

and

$$w(H) = \sum_{h \in H} 2^{z_h}, \quad \text{where} \quad \sum_{h \in H} z_h \geq k.$$

Using the convexity of the exponential function, we conclude that $w(H) \geq c2^{k/c}$. Since $w(H) \leq w(X)$, we finally have

$$c2^{k/c} \leq ne^{k/2c} \leq n2^{3k/4c},$$

from which $k \leq 4c \log(n/c)$ follows. The bound of n^4/c^3 on $w(X)$ is an immediate consequence. □

If the process exceeds this guaranteed number of iterations, that implies that there is no cover of size c . Thus, we can use this procedure to determine an approximate bound for c . To start, we conjecture a value c' of c , which initially can be set to one. In general, if our routine fails to find a hitting set, then there is no hitting set of size c' , so we increase c' by a factor of two. At the stage when we find a hitting set, we have $c' \leq 2c$, so the hitting set returned by the algorithm is of size at most $s(4c)$. This concludes the description of the algorithm.

3.2. ε -Nets With or Without Weights?

Typically, ε -net algorithms are designed for the uniform case and not the weighted case. Here we mention a simple method for reducing the weighted case to an

unweighted one, as outlined by Matoušek [36]. First scale the weights such that $w(X) = n$. Then make $\lceil w(x) + 1 \rceil$ copies of each element $x \in X$. Note that the multiset X' thus obtained contains all the elements of X and has a cardinality of at most $2n$. Then take an ε -net for the set X' : it is also an ε -net for the original set X with weights w . Since this does only involve elementary operations on words of $O(\log n)$ length, by our choice of the log-RAM model, the reduction takes $O(n)$ time.

3.3. Running-Time Analysis

For a given c' , it is clear that there can be at most $4c' \log(n/c')$ iterations, each of which takes time $T_A(n, m, 2c') + T_B(n, m)$. However, the total weight can never exceed n^4/c^3 , which is between n and n^4 . This proves that the length of any weight is at most $4\lceil \log n \rceil$, which fits in the log-RAM model. When c' increases geometrically, by our choice of polynomial functions for T_A and T_B , the sum of all running times is dominated by its last term, for which we have $c \leq c' \leq 2c$. This concludes the proof of Theorem 3.2.

4. Lower Bounds for Other Methods

In this section we give some specific cases where our method improves the previous methods. We begin with the greedy method.

4.1. The Greedy Method

The following example has been communicated to us by Jiří Matoušek (private communication). Let $S = \{p_{i,j} : i = 1, 2, j = 1 \dots 2^{p+1} - 1\}$ be a set of $n = 2^{p+2} - 2$ distinct points, and let its collection of subsets consist of $S_j = \{p_{i,k} : i = 1, 2, k = 2^j \dots 2^{j+1} - 1\}$, for $i \in \{1, 2\}$, $j \in \{0 \dots p\}$, as well as the two particular sets $T_i = \{p_{i,j} : j = 1 \dots 2^{p+1} - 1\}$, $i \in \{1, 2\}$ (see Fig. 1). The greedy method picks the cover $(S_j)_{j=1-p}$, which is of size $p = \log(n + 2) - 2$. The optimal cover is $(T_i)_{i=1,2}$ and is of size two. The dual VC-dimension of our example is two, as every point point is covered by exactly two subsets, and our algorithm (in a dual setting to obtain a set cover) picks the optimal cover for a value of $c = 2$.

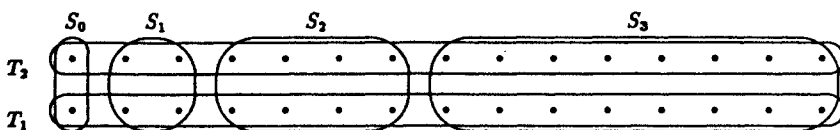


Fig. 1. A greedy method's worst case that has finite VC-dimension.

4.2. Hochbaum's Method

In Hochbaum's method the set system of m sets over n elements is denoted as a 0-1 (n, m) -matrix A , each column A_j of which is the incidence vector of one of the sets. The algorithm then solves the linear program $\max e_n^T \cdot y$ subject to $y^T \cdot A \leq e_m$, $y \geq (0, \dots, 0)$, where e_k is a k -vector whose components are all ones. Let J denote the sets of tight constraints at the optimum y^* (in mathematical notation, j is in J if and only if $(y^*)^T \cdot A_j = 1$). Then J is a set cover within f times the optimum, where f is the maximum sum of a row of A .

Take k to be any constant. We put

$$n = \binom{m}{k}$$

and take for A the (n, m) -matrix whose rows are made up of all possible vectors with $m - k$ ones and k zeros (in no particular order). The corresponding set system has n elements and m sets. By symmetry, the method will pick all the sets in the cover, which gives a cover of size m , guaranteed with $m - k$ from the optimum. Any $k + 1$ elements will constitute an optimal cover.

This example has both dual VC-exponent and dual VC-dimension k , so our algorithm will return a cover of size $O(k^2 \log k)$.

Remark. Interestingly enough, the bad example for the greedy (resp. Hochbaum's) method can be solved efficiently with our as well as Hochbaum's (resp. the greedy) method. We do not know of an example for which our method outperforms both of them.

5. Applications

In this section we cover mainly computational geometry applications, therefore it is customary to assume the real RAM model where all the integers and their elementary arithmetic operations have unit complexity.

5.1. Separating Polyhedra

Let $Q \subseteq P$ be two convex polytopes in \mathbb{R}^d . The problem of finding a convex polytope between P and Q (which from now on we call a *separator*) with as few facets as possible is believed to be NP-hard, even in the three-dimensional case [22], but good approximations can be found for it. Mitchell and Suri cast the problem as a hitting-set problem [42]: Let $\mathcal{H} = \mathcal{H}(Q)$ denote the set of hyperplanes supporting a facet of Q ; let the *cover set system* be $(\mathcal{H}, \{\mathcal{H}_p: p \in \partial P\})$, where ∂P denotes the boundary of P and, for any point p , \mathcal{H}_p consists of those hyperplanes in \mathcal{H} for which p and Q lie on opposite sides. (For definiteness, we agree that if p is on h , then h is not in \mathcal{H}_p .) They show that a hitting set for the cover set system gives a subset of

facets of Q whose bounding half-spaces define a convex polytope between Q and P . A convex polytope Q' such that each facet of Q' contains a facet of Q is called *canonical*. The smallest (with respect to the number of facets) canonical separator can be obtained from a minimal hitting set of this set system and is within d times the optimal separator (in number of faces). Therefore the greedy strategy returns a separator of size within $O(d^2 \log|Q|)$ of the optimal. In a recent twist, Clarkson [19] applies ideas he used for small-dimensional linear programming to give a polynomial-time randomized method that produces an approximation within $O(d^2 \log(dc))$ of the optimal c . In fact, the algorithm of this paper is a deterministic analogue of Clarkson's method in our more general framework.⁴ Interestingly, however, for the important case of $d = 3$, we are able to improve on Clarkson's result to achieve an approximation factor that is $O(1)$.

However, we first describe our method for general $d \geq 4$. We need to exhibit an appropriate net finder and a verifier. Note that since P contains Q , \emptyset is not in the cover set system. The basic observation is that the cover set system is a subsystem of the half-space set system, and therefore has VC-dimension and VC-exponent d (since $d + 1$ hyperplane define at most $2^{d+1} - 1$ regions of the space, and the complexity of an arrangement of m half-spaces is $O(m^d)$). Therefore our hope is to find a net finder of size s , where $s(x) = O(dx \log(dx))$.

To find a $(1/r)$ -net for the weighted cover set system, we simply use the algorithm of [10], which computes a weighted $(1/r)$ -net in $O(nr^d \log^d(dr))$ time, using the reduction of Section 3.2 to take care of the weighted case. This provides a $(1/r)$ -net for the more general set system consisting of *all* half-spaces, but this is *a fortiori* a $(1/r)$ -net for the cover set system, and the bound on its size is still $O(dr \log(dr))$.

As for the verifier, let Y^\cap denote the canonical polytope with k faces associated with the hyperplanes of some $Y \subseteq \mathcal{H}$ of size k . In particular, $\mathcal{H}(Q)^\cap = Q$. All we are asking for is whether Y^\cap is entirely contained in P , or else to give a point of ∂P which is in Y^\cap . We can simply compute the intersection of all the half-spaces defined by the hyperplanes of $Y \cup \mathcal{H}(P)$ and test whether there is a facet that belongs to P . If so, we can return \mathcal{H}_p where p is, e.g., the centroid of this face. Otherwise, Y^\cap is certainly contained in P .

We can find a $(1/r)$ -net in $O(r^d \log^d(dr)|Q|)$ time and verify k half-spaces in $O(k + |P|)^{\lfloor d/2 \rfloor}$ time (if $d \geq 4$) [11] or in $O((k + |P|) \log(k + |P|))$ time (for $d = 2, 3$) [43]. Plugging those bounds into our main theorem, we get:

Theorem 5.1. *Let $Q \subseteq P$ be two convex nested polytopes in \mathbb{R}^d ($d \geq 4$) with a total of n facets. It is possible to find a separator of size within $O(d^2 \log c)$ of the optimum c , in a deterministic time of $O(n^{\lfloor d/2 \rfloor} + nc^d \log^d(dc))c \log(n/c)$, which is always $O(n^{d+2} \log^d n)$.*

Note that the case where $c = \Omega(n^\delta)$ for some small $\delta > 0$ is of little interest because, in this instance, our algorithm offers no better a performance ratio than the greedy method (at least asymptotically). Thus we could make the algorithm

⁴ Clarkson's exposition is restricted to polytopal problems.

faster (for the same guaranteed ratio) by switching to the greedy method when c becomes too large.

However, in three dimensions, we can actually do much better. In particular, we recall from Matoušek and coworkers [37], [40] that the three-dimensional half-space set system admits $(1/c)$ -nets of size $O(c)$, and, as indicated in Section 3.2, their algorithm can be extended to the weighted case as well. The cost of computing a $(1/c)$ -net of size $O(c)$ is $O(nc)$, if $c \leq n^\delta$ [37], and the cost of verifying that it is a hitting set is $O(n \log n)$ as argued above. Therefore we obtain the following strengthening of our previous theorem:

Theorem 5.2. *Let $Q \subseteq P$ be two nested polyhedra in \mathbb{R}^3 with a total of n facets, one of them being convex. It is deterministically possible to find a separator of size within $O(1)$ from the size c of an optimum separator, in $O(nc(c + \log n) \log(n/c))$ time if $c \leq n^\delta$ for some $\delta > 0$.*

The algorithm of Theorem 5.2 has been animated into a video [8]. In particular, it should be noted that the running time is always $O(n^{1+2\delta} \log n)$, which improves on the $O(n^4)$ time bound of Mitchell and Suri's method [42] for the general case. Unfortunately, for large $c \geq n^\delta$, we have to switch to an $O(n^3)$ method to find the net [40], yielding $O(n^3 c \log(n/c))$ time. Note that this is still $O(n^4 \log n)$, whereas the time consumed by the greedy method is $O(n^4)$ (though Mitchell and Suri reduce it to $O(n^3)$ when both polyhedra are convex).

5.2. Decision Trees

Arkin *et al.* [1] describe how to construct a point-probe decision tree for a set of nondegenerate polygons in the plane whose height is $s - 1 + \lceil \log(k/(s - 1)) \rceil$, when given a hitting set of size s . For some applications, the result serves to decide the position of a polygon in a scene [1]. While their approach is more general, the next result shows that we can get a better performance ratio in the height of a decision tree for nondegenerate (i.e., general position) inputs. We define the *point set system* of a family S of plane objects by all subsets S_p of S for all points p , where S_p consists of all objects in S containing p .

Lemma 5.3. *Let P be a polygon in the plane, let S be a family of congruent copies of P , and let (S, \mathcal{R}) be the point set system of S . Then (S, \mathcal{R}) has VC-exponent at most two (the constant two cannot be improved in general), and admits a witness subsystem oracle of degree two. The exponent is still two if P is convex.*

Proof. Clearly, the number of sets of (S, \mathcal{R}) is at most the number of cells in the arrangement of S . If all m polygons in S have k vertices each, their arrangement has complexity $O(k^2 m^2)$, which corresponds to the complexity of the arrangement of their supporting lines. The complexity drops to $O(km^2)$ if the polygons are convex, which provides somewhat better constants for the shatter function. Nevertheless, it is easy to come up with examples for which those bounds are tight and also reflect the

number of different subsets in the point set cover. This also provides the construction of the witness oracle: The oracle constructs the arrangement and simply picks a point in each of the cells. If asked to provide a witness, the point is tested against *all* the polygons, in $O(m \log k)$ time. (Here, we treat k as a constant, and we are only interested in the number of elements of S .) \square

Our algorithm, combined with the above observation, and the result of Arkin *et al.* [1] gives us a decision tree of smaller approximation factor than the greedy method for the nondegenerate arrangement of Arkin *et al.* (For degenerate arrangements they design a new greedy strategy, which seems not to be describable as a set-cover strategy.) Moreover, our method can also be used to derive point-probe decision trees of better approximation factor for any set of k -gons in general position in the plane, for constant k , since this point set system also has bounded VC-dimension (as the proof above shows as well).

Remark. Clearly, the VC-dimension depends on the number of sides of P . If S consists of congruent copies of a single convex set, even a convex polygon with infinitely many sides, then its point set system (S, \mathcal{A}) need not have a finite VC-exponent (resp. VC-dimension), as illustrated in the following example, slightly modified from János Pach (private communication). For every subset $I \subset \mathbb{N}$, let

$$\theta(I) = \sum_{i \in I} 2^{-2^i} \in [0, 1).$$

To each I , we associate the point on the circle $p_I = e^{2\pi i \theta(I)}$. Obviously, all these points are distinct. We construct a family $S = (C_i)_{i \in \mathbb{N}}$ of convex sets as follows. Let r_{ji} be the rotation around the origin that takes $p_{(i)}$ to $p_{(j)}$. (Note that r_{ji} composed with r_{kj} yields r_{ki} .) Let D_i be the convex hull of all points p_I such that $i \in I$. Finally, let C_i be the convex hull of all the $r_{ik}(D_k)$'s, for all integers k . It remains to see that all the C_i 's are congruent, but this is immediate, as $C_i = r_{ij}(C_j)$. Last but not least, p_I belongs exactly to those C_i 's for which $i \in I$. This is implied by the fact that the p_I 's are in general position (no three $\theta(I)$'s form an arithmetic progression). Therefore S itself is shattered by its point set system.

5.3. Disk Cover

Let S be a set of n points in the plane and let \mathcal{D} be a family of disks. The *disk-cover* problem is to find a minimum number of disks in \mathcal{D} that cover the points in S [27]. This problem is motivated by VLSI design criteria as well as viewing problems in astronomy. Matoušek *et al.* [40] show that this set system admits an $O(r)$ -sized $(1/r)$ -net and, as indicated in Section 3.2, their algorithm can be extended to weighted point sets. The resulting algorithm runs in $O(nr \log n)$ time, and implies the following:

Theorem 5.4. *Let S be a set of points in the plane, and \mathcal{D} be a set of disks whose union contains S . It is possible to find a disk cover of S from \mathcal{D} (a subset of \mathcal{D} whose*

union still contains P) whose size is no more than $O(1)$ times the optimal size c of such a cover in $O(c^2 n \log n \log(n/c))$ time.

Note that our method never takes more than $O(n^3 \log n)$ time. This bound contrasts a bound of Hochbaum and Maas [27] for finding a constant-factor approximation to a disk cover, as their method requires all the disks in \mathcal{D} to have the same radius and, even when all the parameters in their method are optimized for the running time, their method takes $O(n^5)$ time.

5.4. Learning a Union of Half-Spaces in Fixed Dimension

Let H be a set of s half-spaces in \mathbb{R}^d , and write H^+ for their union, and H^- for its complement. Let D be an (unknown) probabilistic distribution on \mathbb{R}^d . A *learning algorithm* A for H^+ , given m examples drawn randomly with a distribution D , outputs another set of half-spaces, G , called the *hypothesis*, such that with confidence $1 - \delta$ (on the examples fed to the algorithm), G verifies $D(H^+ \Delta G^+) < \varepsilon$. (Δ denotes the symmetric difference.) An example is labeled positive if it falls in H^+ , negative otherwise. In a learning situation, H^+ is unknown as a polytope, and can only be known through the examples' labels. The purpose of the hypothesis is to provide a model for H^+ , and the property of the learning algorithm says that with probability at least $1 - \delta$, for subsequent queries, the label as computed with G^+ will be correct (according to H^+) with probability at least $1 - \varepsilon$.

Blum and Rivest [6] prove that this problem admits no proper learning algorithm (for which the hypothesis also consists of s half-spaces) whose running time is polynomial in $d, s, \varepsilon^{-1}, \delta^{-1}$, even if $s = 2$ (unless $P = NP$). Baum [3] argues that this only shows that we should look for hypotheses with more half-spaces. Indeed, Blumer *et al.* [7] and Baum [3] have proposed algorithms that output a hypothesis of size $O(s \log m)$. These algorithms run in polynomial time in fixed dimension (though the dependence in the dimension is exponential). Unfortunately, with the purpose of training a neural net in mind, we see that each half-space of the hypothesis corresponds to a gate of the neural net, and therefore the size of the neural net increases with precision.

The algorithm of Blumer *et al.* proceeds by first labeling the examples, then forming a set system of the half-spaces containing only positive examples, and finally returning a small set of half-spaces covering all the positive examples. For this last step we can use our method: The hypothesis is guaranteed to have size $O(ds \log(ds))$, independent of the number of examples. Moreover, we can improve on the results in two and three dimensions. This proves:

Theorem 5.5. *For any fixed $d > 1$, given s half-spaces H in \mathbb{R}^d , and any distribution on points in \mathbb{R}^d , there is a neural net of size $O(ds \log(ds))$ which can be trained, in time polynomial in s, ε, δ and with confidence $1 - \delta$, to recognize whether a point belongs to the union of H with a probability of error less than ε . The size of the neural net can be brought down to $O(s)$ if $d = 2, 3$.*

The result, of course, is also valid for learning the union of concepts taken from a concept class that has finite VC-dimension.

6. Experimental Results

Our algorithm has been implemented and the results are described in [9]. In general, there are many possible choices for the implementation, as we have many ways to compute an ε -net in practice, and they result in different performances.

The first consists in choosing a random sample of size $O(dc \log c)$. It turns out that this performs badly against the greedy method, as the values of n we consider are never big enough to justify the use of our method with the random sampling against the greedy method.

The second possible choice is to use a computed ε -net, with the algorithm of [10], and this seems to yield results comparable (as for the performance ratio) with the greedy method. However, the lack of an efficient subsystem oracle for the random set systems considered in their experimentation made this implementation much slower.

The third method of choice is to compute the net using the greedy method [14]. Even though this does not guarantee the same performance as the second choice, it performs well in practice, and never returns anything bigger than that returned by the greedy set-cover method. Moreover, the lower bound as returned by our method is substantially higher than that returned by the greedy method. Therefore, if it does not produce better hitting sets, at least it is able to give better lower bounds in practice. However, instead of using the greedy method once, we use it many times in computing the nets (although on smaller set systems), thereby making this implementation run rather slowly.

For all three choices, the fear that weights might increase beyond the finite precision of the machine and therefore require a multiprecision treatment is dispelled by empirical observation [9]. In the largest example tried (of size about 10^6), after some 700 iterations, the maximum weight was a bare 16,384—as opposed to a possible 10^{12} which would still have required only 42 bits. This strongly suggests that the normal precision (of 32 bits) will suffice in all reasonable instances of the problem. This is due, of course, to the fact that only small weights tend to be doubled through the algorithm.

One last issue concerns the generation of “random” set systems of finite VC-dimension. This seems to be a hard problem; the set systems tried in [9] include random subsystems of known finite VC-dimension as well more natural set systems, such as the cover-set system of two polyhedra.

7. Conclusions

We have shown an approximation algorithm for the hitting-set and set-cover problems, and proved that it performs well if the VC-dimension of the set system is bounded. If the set system is described by a subsystem oracle, our algorithm is faster than other existing methods which demand a full description of the set system. For

example, if there is a constant-sized hitting set, the algorithm runs in linear time. Furthermore, our algorithm seems to defeat known complexity-theoretic lower bounds [4], [33] (which make no assumptions about the VC-exponent), by being adaptive: the competitive ratio depends on the size of the optimal solution. However, the proof methods of [4] and [33] require (to the best of our understanding) that the optimal hitting set is large. It is thus not clear what the applicability of their results to ours is.

We have exhibited a variety of computational geometry problems that reduce to or use hitting sets, and for which the set systems have bounded VC-dimension. Interestingly enough, the algorithm was prompted by one of these problems for which Clarkson gave a randomized algorithm [19] (which is essentially the same as ours when specialized to the polytope separation problem, except that the net there is picked as a random sample).

Our algorithm has been implemented, and the practical results are encouraging. Practically, the method of choice seems to be a “hybrid” algorithm, which starts by calling our method but switches to the greedy method when the performance ratio offered by Theorem 3.2 exceeds the factor $\alpha = (1 + \ln A)$ guaranteed by the greedy method.

The main open question pertaining to this research is whether our algorithm can be made to run in parallel in polylogarithmic time and polynomial number of processors. It seems that the main problem lies in the iteration process: if we are to double the weights of many sets at once, we are unable to enforce that the weights of many points in an optimal hitting set double at each stage. There are, of course, other open problems as well.

An ε -net is a hitting set for the heavy subsets. The algorithm can be thus adapted to find an almost minimum-sized ε -net. A related problem that would be of some interest is to find a similar algorithm for ε -approximations. It is not even known if finding a minimum-sized approximation is NP-complete, much less if there is any algorithm that returns an almost optimal ε -approximation.

The greedy method is shown to attain an $O(\log n)$ approximation ratio in the polytope separation, but it might be possible to use the geometry to show that it in fact yields a constant ratio. To the best of our knowledge, such a claim has neither been proved nor disproved.

Finally, in the decision-tree problem, we were unable to find a strategy that would work for degenerate arrangements. Is such an adaptation possible?

Acknowledgments

We would like to thank Dan Boneh, Bernard Chazelle, S. Rao Kosaraju, Jiří Matoušek, Joseph Mitchell, János Pach, and Neal Young for helpful discussions concerning the topics of this paper.

References

1. E. M. Arkin, H. Meijer, J. S. B. Mitchell, D. Rappaport, and S. S. Skiena. Decision trees for geometric models. *Proc. 9th Ann. ACM Symp. on Computational Geometry*, pp. 369–378, 1993.

2. P. Assouad. Densité et dimension. *Ann. Institut Fourier* (Grenoble), 3:232–282, 1983.
3. E. Baum. On learning the union of halfspaces. *J. Complexity*, 6:67–101, 1990.
4. M. Bellare, S. Goldwasser, C. Lund, and A. Russel. Efficient probabilistically checkable proofs and applications to approximation. *Proc. 25th Ann. ACM Symp. on Theory of Computing*, pp. 294–304, 1993.
5. B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 54–59, 1989.
6. A. Blum and R. Rivest. Training a 3-node neural network is NP-complete. *Proc. 1st Workshop on Computer Learning Theory*, pp. 9–18, 1988.
7. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik–Chervonenkis dimension. *J. Assoc. Comput. Mach.*, 36:929–965, 1989.
8. H. Brönnimann. Almost optimal polyhedral separators. *Proc. 10th Ann. ACM Symp. on Computational Geometry*, pp. 393–394, 1994. Accompanying video.
9. H. Brönnimann. An implementation of MIN HITTING SET heuristics. Manuscript, 1994.
10. H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. *Proc. 34th Ann. IEEE Symp. on Foundations of Computer Science (FOCS 93)*, pp. 400–409, 1993.
11. B. Chazelle. An optimal convex hull algorithm and new results on cuttings. *Discrete Comput. Geom.*, 10:377–409, 1993.
12. B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.
13. B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, and M. Sharir. Improved bounds on weak ε -nets for convex sets. *Proc. 25th Ann. ACM Symp. on Theory of Computing (STOC 93)*, pp. 495–504, 1993.
14. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.
15. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *Proc. 4th ACM–SIAM Symp. on Discrete Algorithms*, pp. 281–290, 1993.
16. B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.
17. V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4:233–235, 1979.
18. K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 452–456, 1988.
19. K. L. Clarkson. Algorithms for polytope covering and approximation. *Proc. 3rd Workshop on Algorithms and Data Structures*, pp. 246–252. *Lecture Notes in Computer Science*, vol. 709. Springer-Verlag, Berlin, 1993.
20. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.
21. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
22. G. Das. Approximation schemes in computational geometry. Ph.D. thesis, University of Wisconsin, 1990.
23. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
24. M. T. Goodrich. Geometric partitioning made easier, even in parallel. *Proc. 9th Ann. ACM Symp. on Computational Geometry*, pp. 73–82, 1993.
25. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.
26. D. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982.
27. D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *J. Assoc. Comput. Mach.*, 32:130–136, 1985.
28. D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

29. R. M. Karp, Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pp. 85–103. Plenum, New York, 1972.
30. J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ε -nets. *Discrete Comput. Geom.*, 7:163–173, 1992.
31. N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithms. *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pp. 68–77, 1987.
32. L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.
33. C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Proc. 25th Ann. ACM Symp. on Theory of Computing*, pp. 286–293, 1993.
34. J. Matoušek. Construction of ε -nets. *Discrete Comput. Geom.*, 5:427–448, 1990.
35. J. Matoušek. Approximations and optimal geometric divide-and-conquer. *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pp. 505–511, 1991. Also to appear in *J. Comput. System Sci.*
36. J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.
37. J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.
38. J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.
39. J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.
40. J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: small ε -nets for disks and halfspaces. *Proc. 6th Ann. ACM Symp. on Computational Geometry*, pp. 16–22, 1990.
41. J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε -approximations for bounded VC-dimension. *Proc. 32nd Ann. IEEE Symp. on Foundations of Computer Science*, pp. 424–430, 1991.
42. J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral surfaces. *Proc. 3rd ACM-SLAM Symp. on Discrete Algorithms*, pp. 296–306, 1992.
43. F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, 1985.
44. S. Rajagopalan and V. Vazirani. Primal–dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs. *Proc. 34th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 322–331, 1993.
45. N. Sauer. On the densities of families of sets. *J. Combin. Theory*, 13:145–147, 1972.
46. R. E. Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.
47. V. N. Vapnik and A. Ya. Červonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.
48. E. Welzl. Partition trees for triangle counting and other range searching problems. *Proc. 4th Ann. ACM Symp. on Computational Geometry*, pp. 23–33, 1988.

Received June 1994, and in revised form January 1995.