Near-Linear Algorithms for Geometric Hitting Sets and Set Covers

Pankaj K. Agarwal
Duke University
pankaj@cs.duke.edu

Jiangwei Pan
Duke University
jwpan@cs.duke.edu

ABSTRACT

Given a finite range space $\Sigma = (\mathsf{X}, \mathcal{R})$, with $N = |\mathsf{X}| + |\mathcal{R}|$, we present two simple algorithms, based on the multiplicative-weight method, for computing a small-size hitting set or set cover of Σ. The first algorithm is a simpler variant of the Brönnimann-Goodrich algorithm but more efficient to implement, and the second algorithm can be viewed as solving a two-player zero-sum game. These algorithms, in conjunction with some standard geometric data structures, lead to near-linear algorithms for computing a small-size hitting set or set cover for a number of geometric range spaces. For example, they lead to $O(N\text{polylog}(N))$ expected-time randomized $O(1)$-approximation algorithms for both hitting set and set cover if X is a set of points and $\mathcal{R}$ a set of disks in $\mathbb{R}^2$.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures, geometric problems and computations*

General Terms

Algorithms, Theory

Keywords

Geometric range space, Hitting set, Set cover, Multiplicative weight method, Approximation algorithms

1. INTRODUCTION

Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a finite range space where X is a finite set of *objects* and $\mathcal{R}$ is a family of subsets of X called *ranges*. Set $n = |\mathsf{X}|, m = |\mathcal{R}|$, and $N = n + m$. A subset $\mathsf{H} \subseteq \mathsf{X}$ is called a *hitting set* of Σ if H intersects every range of $\mathcal{R}$, and a subset $\mathcal{C} \subseteq \mathcal{R}$ is called a *set cover* of Σ if the union of ranges in $\mathcal{C}$ is equal to X. The hitting-set (resp. set-cover) problem is to find the smallest hitting set (resp. set cover) of Σ. Both problems are well-known to be NP-Complete [31] and have been extensively studied. Let $\kappa := \kappa(\Sigma)$ and $\chi := \chi(\Sigma)$ denote the size of an optimal hitting set and set cover, respectively, of Σ. Sometimes, we also use OPT to denote either κ or χ and the meaning will be clear from the context.

In this paper, we are primarily interested in the hitting-set and set-cover problems for geometric range spaces, where X is a finite set of points in $\mathbb{R}^d$ and $\mathcal{R}$ is a finite family of simply-shaped regions, chosen from some infinite class (e.g., rectangles, balls, simplices, halfspaces). In this case, ranges are $\mathsf{X} \cap R$ for $R \in \mathcal{R}$. With a slight abuse of notation, we denote the range space $\Sigma = (\mathsf{X}, \{X \cap R \mid R \in \mathcal{R}\})$ as $(\mathsf{X}, \mathcal{R})$. The hitting-set and set-cover problems are NP-Complete even for simple geometric range spaces, e.g., when $\mathcal{R}$ is a set of unit disks or unit squares in $\mathbb{R}^2$ [28]. This has led to the development of polynomial-time approximation algorithms for these problems. Traditionally, the focus has been on developing polynomial-time algorithms with the smallest possible approximation ratio. A number of recent applications (e.g., in sensor networks, database systems, computer vision) call for repeated computation of hitting sets and set covers. For such applications, it is desirable to have near-linear-time algorithms for these problems even if it means sacrificing a little on the approximation ratio. Motivated by these applications, we study the problem of computing, in near linear time, near-optimal hitting sets and set covers for geometric range spaces.

Related work. The well-known greedy algorithm gives a polynomial-time $O(\log n)$-approximation for a hitting set or set cover [23]. The known lower bound results imply that this is the best one can hope for, within a constant factor, assuming certain conjectures in complexity theory [26]. However, by exploiting the underlying geometry, polynomial-time algorithms with better approximation factors can be obtained for many geometric range spaces; see [6, 20, 22, 41] and the references therein for various results of this kind. These algorithms employ and adapt a wide range of novel techniques, including the usage of ε-nets [34].

Given a parameter $\varepsilon \in (0, 1]$, an *ε-net* for range space $\Sigma = (\mathsf{X}, \mathcal{R})$ is a subset $\mathsf{N} \subseteq \mathsf{X}$ that intersects every range $R \in \mathcal{R}$ whose size is at least $\varepsilon|\mathsf{X}|$. In other words, N is a hitting set for all the "heavy" ranges. Haussler and Welzl [34] proved that a range space of VC-dimension [1] δ has an ε-net of size $O((\delta/\varepsilon)\log(\delta/\varepsilon))$. The bounds on the size of ε-nets have been improved for several geometric range spaces. For example, $O(1/\varepsilon)$-size ε-nets exist when X is a set of points and $\mathcal{R}$ is a set of halfspaces in $\mathbb{R}^2$ or $\mathbb{R}^3$, pseudo-disks $\mathbb{R}^2$, or translates of a fixed convex polytope in $\mathbb{R}^3$ [36, 39, 44]. Aronov *et al.* [6] gave an $O(n\log^{d-1} n)$-time randomized algorithm to construct an ε-net of size $O((1/\varepsilon)\log\log(1/\varepsilon))$ for the case when $\mathcal{R}$ is a set of axis-parallel d-rectangles for $d = 2, 3$. A remarkable result of Pach and Tardos [42] shows that this bound is tight in the worst case. See also [4, 25].

[1]The *VC-dimension* of a range space $\Sigma = (\mathsf{X}, \mathcal{R})$ is the size of the largest subset $A \subseteq X$ s.t. $|\{A \cap R \mid R \in \mathcal{R}\}| = 2^{|A|}$.

SoCG'14, June 8–11, 2014, Kyoto, Japan.

Building on an iterative-reweighting technique by Clarkson [20], Brönnimann and Goodrich [13] showed that if an ε-net of a range space Σ of size $O((1/\varepsilon)g(1/\varepsilon))$ can be computed in polynomial time, where $g(\cdot)$ is a monotonically-nondecreasing sublinear function, then a hitting set of size $O(\kappa g(\kappa))$ also can be computed in polynomial time. Since a set cover of Σ is a hitting set of the dual range space of Σ, their algorithm also extends to set cover. If Σ has finite VC-dimension, then the dual range space also has finite VC-dimension, and thus a hitting set (resp. set cover) of Σ of size $O(\kappa \log \kappa)$ (resp. $O(\chi \log \chi)$) can be computed in polynomial time. The size of hitting set reduces to $O(\kappa)$ if Σ admits an ε-net of size $O(1/\varepsilon)$ (e.g. when $\mathcal{R}$ is a set of halfspaces in $\mathbb{R}^3$, or a set of pseudo-disks in $\mathbb{R}^2$), and to $O(\kappa \log\log \kappa)$ if $\mathcal{R}$ is a set of rectangles in $\mathbb{R}^2$ or $\mathbb{R}^3$. The Brönnimann-Goodrich algorithm can be viewed as first solving the LP relaxation of the hitting-set problem and then obtaining an approximate hitting set from the fractional LP solution via ε-net. A more general LP solver can also be used to obtain the same results [37]. See [22, 6, 46] for improved approximation ratios for set covers of geometric range spaces.

The algorithms by Clarkson [20] and Brönnimann-Goodrich [13] are instances of the so-called *multiplicative weight* (MW) method. The MW method, which has been repeatedly discovered, goes back to the 1950's and has been used in numerous fields including machine learning [29], linear programming [11, 35, 43, 48], semidefinite programming [8], graph algorithms [9, 19], game theory [30, 32], on-line algorithms [27], and computational geometry. In computational geometry, besides hitting-set/set-cover, the MW method has been used for linear programming [21], constructing spanning trees of low stabbing number [17, 47], and constructing partition trees for range searching [38]. See the survey by Arora *et al.* [10] for a comprehensive review of this method.

Returning to the hitting-set and set-cover problems, both the greedy and the Brönnimann-Goodrich algorithms work in $O(\kappa \log N)$ stages. A straightforward implementation of both algorithms takes $O(mn)$ time per stage, which can be improved to $O(N \operatorname{polylog}(N))$ in some cases using geometric data structures. Thus these algorithms run in time $\Omega(N\kappa)$.

Agarwal *et al.* [3] studied near-linear algorithms for the hitting-set problem. They proposed a variant of the greedy algorithm that performs $O(\log n)$ stages and chooses, in near-linear time, $O(\kappa\phi(\kappa))$ points in each stage for the hitting set if the union of any subset $\mathcal{F} \subseteq \mathcal{R}$ has complexity $O(|\mathcal{F}|\phi(|\mathcal{F}|))$, where $\phi(\cdot)$ is a sublinear function. They also presented an efficient implementation of the Brönnimann-Goodrich algorithm for the special case when $\mathcal{R}$ is a set of d-rectangles for $d = 2, 3$. Their algorithm is rather complex and computes a hitting set of size $O(\kappa \log\log \kappa)$ in $O(N + \kappa^{d+1})\operatorname{polylog}(n))$ time. Not only is it not efficient for large values of κ, but it also does not extend to other range spaces.

Our results. We present two simple algorithms, both based on the MW method, for computing a small-size hitting set or set cover of a range space Σ of finite VC-dimension. They first compute a near-optimal fractional solution using the MW method and then transform this fractional solution to an integral solution using ε-nets (e.g., as in [37]). As mentioned earlier, a set cover of Σ is a hitting set of the dual range space $\Sigma^\perp$, so we only describe our algorithms in terms of computing a hitting set.

The first algorithm, described in Section 3, is a simpler but more efficient variant of the Brönnimann-Goodrich algorithm [13]. To expedite the algorithm, it divides the stages into $O(\log n)$ rounds so that each range is processed only once in each round, and it computes an ε-net twice, instead of $O(\kappa \log n)$ times as in [13]. Assuming that an ε-net of Σ can be computed in $O(n \operatorname{polylog}(n))$ time and range queries on Σ can be answered in $\operatorname{polylog}(n)$ time, the expected running time of the algorithm is $O(N \operatorname{polylog}(N))$. See Theorem 3.2 for a more precise bound.

The second algorithm, described in Section 4, is a Monte Carlo algorithm that computes a small-size hitting set or set cover with high probability (i.e., with probability at least $1 - 1/N^{\Omega(1)}$). It can be viewed as solving a two-player zero-sum game. In this game, one player Alice has the points in X as her pure strategies and the other player Bob has the ranges in $\mathcal{R}$ as his pure strategies. If Alice plays $x \in \mathsf{X}$ and Bob plays $R \in \mathcal{R}$, then Bob pays Alice 1 if $x \in R$ and 0 otherwise. An optimal mixed strategy for Alice gives an optimal fractional solution to the hitting-set, from which one can use ε-net to obtain a hitting set of small size. Using the MW method, the algorithm computes near-optimal mixed strategies for both Alice and Bob, which gives a near-optimal fractional solution to the hitting-set problem. We remark that the MW method has been used to solve the zero-sum game approximately [30, 32], but our algorithm is somewhat different, tailored to computing a hitting set.

We next describe in Section 5 consequences of these algorithms for a number of geometric range spaces. The results are presented in terms of the second algorithm and thus achieve the approximation ratios with high probability. We can obtain Las Vegas algorithms for these problems using the first algorithm, possibly paying an additional logarithmic factor in the running time, that are guaranteed to obtain these approximation ratios. More specifically,

- for axis-parallel rectangles in $\mathbb{R}^d$ ($d \leq 3$), an $O(\log\log \mathrm{OPT})$-approximate hitting set or an $O(\log \mathrm{OPT})$-approximate set cover can be computed in $O(N \log^{d+1} N \log\log\log \mathrm{OPT})$ expected time;
- for halfspaces in $\mathbb{R}^3$, an $O(1)$-approximate hitting set or set cover can be computed in $O(N \log^4 N)$ expected time;
- for disks, an $O(1)$-approximate hitting set or set cover can be computed in $O(N \log^4 N)$ expected time;
- for fat triangles, an $O(\log \mathrm{OPT})$-approximate hitting set (resp. set cover) can be computed in $O(N \log^5 N \log\log \mathrm{OPT})$ (resp. $O(N \log^7 N \log\log \mathrm{OPT})$) expected time.

Either no near-linear algorithms with guaranteed approximation ratio were known for these problems, or near-linear algorithms attained a worse approximation ratio. For example, the algorithm by Agarwal *et al.* [3] computes only an $O(\log n)$-approximate hitting set for disks in $O(N \log^3 N)$ time, and it does not extend to set cover. We note that in some cases, polynomial-time algorithms with better approximation ratios are known. For instance, the local-search technique by Mustafa and Ray [41] computes $(1+\varepsilon)$-approximate hitting sets for halfspaces in $\mathbb{R}^3$ and pseudo-disks in $\mathbb{R}^2$, but its running time is $N^{O(\varepsilon^{-2})}$. We remark that our algorithms rely on standard range searching data structures. It might be possible to improve polylog factors in the running time by optimizing these data structures, but we feel it is not worth the effort.

Finally, in Section 6, we extend the second algorithm to compute, in near-linear time, an $O(1)$-approximate (discrete) *maximum independent set* for disks. In this problem, we are also given a discrete range space $\Sigma = (\mathsf{X}, \mathcal{R})$, where X is a set of points and $\mathcal{R}$ a set of disks in $\mathbb{R}^2$, and the goal is to compute the largest subset $\mathcal{I}$ of disks such that no point of X is contained in more than one disk of $\mathcal{I}$. We first use our second algorithm to compute a fractional solution to this problem, and then round the fractional solution to an integral one using the algorithm by Chan and Har-Peled [15]. The algorithm in [15] employs an existing LP solver and runs in quadratic time.

2. PRELIMINARIES

Range space and ε-nets. Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a finite range space, as defined above. The dual range space of Σ, denoted by $\Sigma^\perp = (\mathsf{X}^\perp, \mathcal{R}^\perp)$ is defined as follows. There is an object in $\mathsf{X}^\perp$ for each range in $\mathcal{R}$, and $\mathcal{R}^\perp$ contains a range R_x for each point $x \in \mathsf{X}$, namely, $R_x = \{R \in \mathcal{R} \mid R \ni x\}$. It is well known that a set cover of Σ is a hitting set of $\Sigma^\perp$ and vice-versa. It is also known that if Σ has finite VC-dimension, then so does $\Sigma^\perp$ [33].

A hitting set $\mathsf{H} \subseteq \mathsf{X}$ is called *c-approximate*, for $c \geq 1$, if $|\mathsf{H}| \leq c\kappa(\Sigma)$. Similarly, a set cover $\mathcal{C} \subseteq \mathcal{R}$ is called *c-approximate* if $|\mathcal{C}| \leq c\chi(\Sigma)$.

Given a weight function $w : \mathsf{X} \to \mathbb{R}_{\geq 0}$, for a subset $A \subseteq \mathsf{X}$, we use $w(A)$ to denote the total weight of points in A. Given w and a parameter $\varepsilon \in (0, 1]$, we call a range $R \in \mathcal{R}$ *ε-heavy* if $w(R) \geq \varepsilon w(\mathsf{X})$ and *ε-light* otherwise. A subset $\mathsf{N} \subseteq \mathsf{X}$ is called an *ε-net* with respect to weight function w if $\mathsf{N} \cap R \neq \varnothing$ for every ε-heavy range R of $\mathcal{R}$.

Hitting-set algorithm. Since our first algorithm is similar to that of Brönnimann-Goodrich [13] (and Clarkson [20]), we briefly describe their main idea. Let k be an integer such that $k/2 < \kappa \leq k$. Initialize the weight of each point to 1, i.e., $w(x) = 1$ for all $x \in \mathsf{X}$, and repeat the following step until every range is $\frac{1}{2k}$-heavy:

> *find a $\frac{1}{2k}$-light range R and double the weights of all points in R.*

We refer to this step as a *weight-doubling* step. When the process stops, return a $\frac{1}{2k}$-net N of Σ with respect to the final weights. Since each range in Σ is $\frac{1}{2k}$-heavy, N is a hitting set of Σ. Hence, if a $\frac{1}{2k}$-net of size $O(kg(k))$ can be computed efficiently, the above algorithm computes a hitting set of size $O(\kappa g(\kappa))$. The following lemma, by now a well-known argument (see e.g. [10, 13, 17, 20, 21]), is the key to the performance of the algorithm. For completeness, we also give the proof here. [2]

LEMMA 2.1. *If Σ has a hitting set of size at most k, then the algorithm performs at most $\mu_k := 4k\log(n/k)$ weight-doubling steps. The final weight of X is at most n^4/k^3.*

PROOF. Let H be a hitting set of size k. Since each weight-doubling step is performed on a $\frac{1}{2k}$-light range R, the total weight of X increases by a factor of at most $(1 + \frac{1}{2k})$. Thus, after z weight-doubling steps,

$$w(\mathsf{X}) \leq n(1 + \tfrac{1}{2k})^z \leq ne^{\frac{z}{2k}}.$$

On the other hand, $\mathsf{H} \cap R \neq \varnothing$, thus at least one point of H get its weight doubled after the weight-doubling step on R. Suppose each element of $h \in \mathsf{H}$ is doubled z_h times after z weight-doubling steps, we have

$$w(\mathsf{H}) = \sum_{h \in \mathsf{H}} 2^{z_h} \geq k2^{z/k}.$$

Since $w(\mathsf{H}) \leq w(\mathsf{X})$, we get

$$k2^{z/k} \leq ne^{\frac{z}{2k}} \leq n2^{\frac{3z}{4k}},$$

from which $z \leq \mu_k = 4k\log(n/k)$ follows. The final weight of X follows from the first inequality. □

By Lemma 2.1, if the algorithm does not terminate within μ_k steps, we can conclude that Σ does not have a hitting set of size at most k. An exponential search is used to guess the value of k such that $k/2 < \kappa \leq k$.

The Brönnimann-Goodrich algorithm computes a $\frac{1}{2k}$-net H in each step and identifies a range that does not intersect H as a light range; see the original paper for details.

[2]Throughout this paper, we use $\log x$ to denote $\log_2 x$.

3. HITTING SET IN ROUNDS

Algorithm. The first algorithm is a simpler but more efficient variant of the Brönnimann-Goodrich algorithm [13]. As in [13], assume we have an integer k such that $\kappa \in (k/2, k]$; we perform an exponential search to find such a value of k. The algorithm is similar to [13] except that it works in rounds. Initially, it sets $w(x) = 1$ for all $x \in \mathsf{X}$. Each round performs at most $2k$ weight-doubling steps, as follows. It processes each range $R \in \mathcal{R}$ one by one. If R is $\frac{1}{2k}$-light, it doubles the weights of all points in R. This weight-doubling step on R is performed repeatedly until R becomes $\frac{1}{2k}$-heavy or $2k$ weight-doubling steps have been performed in the current round. Once R becomes $\frac{1}{2k}$-heavy, it is not processed again in the current round, even though it may become $\frac{1}{2k}$-light again later in the current round while other ranges are being processed.

If $2k$ weight-doubling steps have been performed in the current round, the algorithm aborts the current round and moves to the next round. On the other hand, if all ranges have been processed with less than $2k$ weight-doubling steps, the algorithm stops and returns a $\frac{1}{2ke}$-net N of Σ as a hitting set of Σ.

By Lemma 2.1, if Σ has a hitting set of size at most k, then the algorithm performs at most $\mu_k = 4k\log(n/k)$ weight-doubling steps. Since each except the last round performs $2k$ weight-doubling steps, the number of rounds is at most $\mu_k/2k + 1 = 2\log(n/k) + 1$.

Correctness. The correctness of the algorithm follows from the following lemma.

LEMMA 3.1. *All ranges are $\frac{1}{2ke}$-heavy when the algorithm terminates.*

PROOF. Suppose the algorithm stops in round i. Let W_i be the total weight $w(\mathsf{X})$ in the beginning of round i, and let W_f be the total weight when the algorithm stops. Since at most $2k$ weight-doubling steps are performed in round i and each of them increases the total weight by a factor of at most $1 + \frac{1}{2k}$,

$$W_f \leq (1 + \tfrac{1}{2k})^{2k} W_i \leq eW_i.$$

After a range R has been processed in round i, it is $\frac{1}{2k}$-heavy with respect to the current weight of X, which is at least W_i. Therefore

$$w(R) \geq \tfrac{1}{2k} W_i \geq \tfrac{1}{2ke} W_f,$$

implying that R is $\frac{1}{2ke}$-heavy. Since the algorithm terminates in round i only after all ranges have been processed in that round, all ranges are $\frac{1}{2ke}$-heavy. □

If an ε-net of Σ of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ can be computed, then by Lemma 3.1, the above algorithm returns a hitting set of size $O(\kappa g(\kappa))$.

Running time. To expedite the weight-doubling step, we perform the following preprocessing step before running the algorithm: compute a $\frac{1}{2k}$-net N_0 of Σ, and set $\mathsf{X} = \mathsf{X} \setminus \mathsf{N}_0$ and $\mathcal{R} = \{R \mid R \cap \mathsf{N}_0 = \varnothing\}$. After this preprocessing, each range in $\mathcal{R}$ contains at most $\frac{n}{2k}$ points. The algorithm returns $\mathsf{N}_0 \cup \mathsf{N}$ as a hitting set of Σ.

We assume the existence of the following three procedures:

(P1) A net finder that can compute an ε-net of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ in $\varphi(N)$ time.

(P2) A range-counting data structure that, given a range R, returns YES if R is $\frac{1}{2k}$-light and NO otherwise, and that can also update the weight of a point. Let $\tau(n) = \Omega(\log n)$ be the time taken by each of these two operations. We assume that the data structure can be built in $O(n\tau(n))$ time.

(P3) A range-reporting data structure that reports all s points of a range in $O(\tau(n)+s)$ time, and that can be constructed in $O(n\tau(n))$ time.

Using (P2) and (P3), a range $R \in \mathcal{R}$ can be processed in each round as follows. First, using (P2) we check in $\tau(n)$ time whether R is $\frac{1}{2k}$-light. If the answer is YES, we use (P3) to report all points of R in time $\tau(n) + |R| \leq \tau(n) + \frac{n}{2k}$. Recall that at most $2k$ weight-doubling steps are performed in each round, so $O(n)$ points are reported in a round. For each reported point, it takes $\tau(n)$ time to double its weight in the range-counting data structure. We thus conclude that a round takes $O(N\tau(n))$ time. Summing over all $2\log(n/2k)+1$ rounds, the total time spent by the algorithm, including the time spent in the preprocessing phase, is $O(N\tau(n)\log n + \varphi(N))$. We repeat the algorithm $O(\log\kappa)$ times to perform the exponential search. Putting everything together, we obtain the following.

THEOREM 3.2. *Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a finite range space with $|\mathsf{X}| + |\mathcal{R}| = N$. Suppose procedures (P1), (P2) and (P3) exist for Σ, then an $O(g(\kappa))$-approximate hitting set of Σ can be computed in $O((N\tau(N)\log N + \varphi(N))\log\kappa)$ time.*

Recalling that a set cover of Σ is a hitting set of the dual range space $\Sigma^{\perp}$, we obtain the following.

THEOREM 3.3. *Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a finite range space with $|\mathsf{X}| + |\mathcal{R}| = N$. Suppose procedures (P1), (P2) and (P3) exist for $\Sigma^{\perp}$, then an $O(g(\chi))$-approximate set cover of Σ can be computed in $O((N\tau(N)\log N + \varphi(N))\log\chi)$ time.*

Remarks. (i) An approximate range counting data structure can be used in (P2), namely, a data structure that always returns YES if R is $\frac{1}{4k}$-light and always returns NO if R is $\frac{1}{2k}$-heavy. It may return YES or NO if $w(R) \in [\frac{1}{4k}w(\mathsf{X}), \frac{1}{2k}w(\mathsf{X})]$. So if the algorithm returns YES, then R must be $\frac{1}{2k}$-light; on the other hand, if the algorithm returns NO, then R must be $\frac{1}{4k}$-heavy. A weight-doubling step is performed if the procedure returns YES. The same argument as in Lemma 3.1 implies that each range is at least $\frac{1}{4k\epsilon}$-heavy when the algorithm terminates, after $O(\log(n/k))$ rounds. Hence, the size of hitting set is affected by a factor of at most 2. This is convenient because faster data structures are known for approximate range counting for some range spaces (e.g., halfspace range counting).

(ii) By Lemma 2.1, the final weight of X is bounded by $O(n^4/k^3)$, so $O(\log n)$ bits suffice for maintaining the weight of each point during the algorithm.

4. HITTING SET AS 2-PLAYER GAME

Algorithm. We now describe the second algorithm for computing a hitting set. As earlier, suppose we have an integer k such that $k/2 < \kappa \leq k$; we perform a backward exponential search to find such a value of k as described later. The algorithm now maintains weights on both X and $\mathcal{R}$, and it also works in rounds. For $i \geq 1$, let $\pi^i : \mathsf{X} \to \mathbb{R}_{\geq 0}$ and $\omega^i : \mathcal{R} \to \mathbb{R}_{\geq 0}$ be the weights of points and ranges, respectively, in the beginning of round i. Initially, $\pi^1(x) = 1$ for all $x \in \mathsf{X}$ and $\omega^1(R) = 1$ for all $R \in \mathcal{R}$. Let Π^i (resp. Ω^i) denote the probability distribution induced by π^i (resp. ω^i), i.e.,

$$\Pi^i = \left\langle \tfrac{\pi^i(x_j)}{\pi^i(\mathsf{X})} \mid 1 \leq j \leq n \right\rangle, \quad \Omega^i = \left\langle \tfrac{\omega^i(R_j)}{\omega^i(\mathcal{R})} \mid 1 \leq j \leq m \right\rangle.$$

We set

$$\mu := \tfrac{2}{\ln 2} k \ln(m^2 n).$$

The algorithm performs μ rounds of the following two steps. In round i, it samples a point $\bar{x}_i \in \mathsf{X}$ from the distribution Π^i and a range $\bar{R}_i \in \mathcal{R}$ from the distribution Ω^i. For each point $x \in \bar{R}_i$, we double its weight, i.e., $\pi^{i+1}(x) = 2\pi^i(x)$, and for each range R that contains $\bar{x}_i$, we halve its weight, i.e., $\omega^{i+1}(R) = \omega^i(R)/2$.

Let $\tilde{\mathsf{X}} = \langle \bar{x}_1, \ldots, \bar{x}_\mu \rangle$ be the multi-set of points chosen by the algorithm. Let $\tilde{\Pi}$ be the distribution on X induced by $\tilde{\mathsf{X}}$, i.e., if a point $x \in \mathsf{X}$ appears μ_x times in $\tilde{\mathsf{X}}$, then set $\Pr(x \sim \tilde{\Pi}) = \mu_x/\mu$. We compute an $\frac{1}{8k}$-net N of Σ with respect to $\tilde{\Pi}$. If N is a hitting set of Σ, we return N; otherwise, we repeat the above algorithm.

Correctness. We view the hitting-set problem for Σ as a two-player zero-sum game, and the above algorithm computes a near-optimal mixed strategy for each of the two players. More precisely, let Alice and Bob be two players who play the following game: Alice chooses a point $x \in \mathsf{X}$, Bob chooses a range $R \in \mathcal{R}$ and Bob pays $I(x,R)$ to Alice, where

$$I(x,R) = \begin{cases} 1 & \text{if } x \in R, \\ 0 & \text{if } x \notin R. \end{cases}$$

Then the algorithm can be interpreted as Alice and Bob playing the game for μ rounds. Initially, both players have uniform distributions over their pure strategies. In each round, Alice samples a point x from the current distribution over points and Bob samples a range R from the current distribution over ranges. According to the definition of the game, points in R are good for Alice because had she played one of them, she would have won 1 dollar. Therefore, Alice doubles the weights of points in R so that she can choose them more often in the future. Similarly, ranges containing x are bad for Bob, thus he halves the weights of those ranges.

For a probability distribution Π over X and for a range $R \in \mathcal{R}$, let $I(\Pi, R)$ denote the expected payoff to Alice if Bob chooses R, i.e.,

$$I(\Pi, R) = \sum_{x \in \mathsf{X}} \Pr(x \sim \Pi) I(x,R).$$

Similarly, we define $I(x,\Omega)$ for a point $x \in \mathsf{X}$ and a distribution Ω over $\mathcal{R}$. Let λ^* be the value of the above game, then by the min-max theorem [40],

$$\lambda^* = \max_{\Pi} \min_{R \in \mathcal{R}} I(\Pi, R) = \min_{\Omega} \max_{x \in \mathsf{X}} I(x, \Omega), \tag{1}$$

where Π, Ω are probability distributions over X and $\mathcal{R}$, respectively.

Let $\mathsf{H}^* \subseteq \mathsf{X}$ be an optimal hitting set of Σ of size κ, and let Π_{H^*} be the distribution where $\pi(x) = 1/\kappa$ if $x \in \mathsf{H}^*$ and 0 otherwise. Then

$$\min_{R \in \mathcal{R}} I(\Pi_{\mathsf{H}^*}, R) \geq 1/\kappa$$

because $R \cap \mathsf{H}^* \neq \varnothing$ for all $R \in \mathcal{R}$. Hence $\lambda^* \geq 1/\kappa \geq 1/k$. Let

$$\Pi^* = \arg\max_{\Pi} \min_{R \in \mathcal{R}} I(\Pi, R)$$

be the optimal (mixed) strategy for Alice. If we can compute Π^*, then we can simply return a $(1/k)$-net N of Σ under the distribution Π^*: N is a hitting set of Σ because the weight of any $R \in \mathcal{R}$ under Π^* is at least $\min_{R \in \mathcal{R}} I(\Pi^*, R) = \lambda^* \geq 1/k$. We show that $\tilde{\Pi}$ is an approximation of Π^* in the sense that with constant probability,

$$\min_{R \in \mathcal{R}} I(\tilde{\Pi}, R) \geq \tfrac{1}{8k},$$

and thus a $\frac{1}{8k}$-net of Σ under $\tilde{\Pi}$ is a hitting set of Σ.

We begin by proving two lemmas, which follow from standard arguments for the MW method. The first one states that the total expected payoff to Alice over μ rounds is not much less than the payoff she would get by the best pure strategy.

LEMMA 4.1. *For every $x \in \mathsf{X}$,*

$$\sum_{t=1}^{\mu} I(\Pi^t, \bar{R}_t) \geq -\ln n + \ln 2 \sum_{t=1}^{\mu} I(x, \bar{R}_t).$$

PROOF. Let $\pi^t(\mathsf{X})$ be the total weight of X in the beginning of round t. Then by construction,

$$\begin{aligned}
\pi^{t+1}(\mathsf{X}) &= \sum_{x \in \mathsf{X}} \pi^t(x)(1 + I(x, \bar{R}_t)) \\
&= \pi^t(\mathsf{X})\left(1 + \sum_{x \in \mathsf{X}} \frac{\pi^t(x)}{\pi^t(\mathsf{X})} I(x, \bar{R}_t)\right) \\
&= \pi^t(\mathsf{X})(1 + I(\Pi^t, \bar{R}_t)) \leq \pi^t(\mathsf{X}) \exp(I(\Pi^t, \bar{R}_t)) \\
&\leq n \exp\left(\sum_{i=1}^{t} I(\Pi^i, \bar{R}_i)\right). \qquad (2)
\end{aligned}$$

The last inequality follows because $\pi^1(\mathsf{X}) = n$. However, for every $x \in \mathsf{X}$,

$$\pi^{\mu+1}(\mathsf{X}) \geq \pi^{\mu+1}(x) = \exp\left(\ln 2 \sum_{t=1}^{\mu} I(x, \bar{R}_t)\right). \qquad (3)$$

The lemma follows from (2) and (3). □

Since Lemma 4.1 holds for every x, it also holds for the optimal mixed strategy Π^*:

COROLLARY 4.2. $\sum_{t=1}^{\mu} I(\Pi^t, \bar{R}_t) \geq -\ln n + \ln 2 \sum_{t=1}^{\mu} I(\Pi^*, \bar{R}_t)$.

A similar argument proves the following lemma, which states that the expected cost to Bob is not much worse than that by his best pure strategy.

LEMMA 4.3. *For every $R \in \mathcal{R}$,*

$$\sum_{t=1}^{\mu} I(\bar{x}_t, \Omega^t) \leq 2 \ln m + \ln 4 \sum_{t=1}^{\mu} I(\bar{x}_t, R).$$

PROOF. Let $\omega^t(\mathcal{R})$ be the total weight of $\mathcal{R}$ in the beginning of round t. Then by construction,

$$\begin{aligned}
\omega^{t+1}(\mathcal{R}) &= \sum_{R \in \mathcal{R}} \omega^t(R)(1 - \frac{1}{2} I(\bar{x}_t, R)) \\
&= \omega^t(\mathcal{R})\left(1 - \frac{1}{2} \sum_{R \in \mathcal{R}} \frac{\omega^t(R)}{\omega^t(\mathcal{R})} I(\bar{x}_t, R)\right) \\
&= \omega^t(\mathcal{R})(1 - \frac{1}{2} I(\bar{x}_t, \Omega^t)) \leq \omega^t(\mathcal{R}) \exp\left(-\frac{1}{2} I(\bar{x}_t, \Omega^t)\right) \\
&\leq m \exp\left(-\frac{1}{2} \sum_{i=1}^{t} I(\bar{x}_i, \Omega^i)\right). \qquad (4)
\end{aligned}$$

The last inequality follows because $\omega^1(\mathcal{R}) = m$. However, for every $R \in \mathcal{R}$,

$$\omega^{\mu+1}(\mathcal{R}) \geq \omega^{\mu+1}(R) = \exp\left(-\ln 2 \sum_{t=1}^{\mu} I(\bar{x}_t, R)\right). \qquad (5)$$

The lemma follows from (4) and (5). □

The following lemma follows from the fact that the game Alice and Bob play is a zero-sum game, but we sketch a proof for completeness.

LEMMA 4.4. $\mathsf{E}\left[\sum_{t=1}^{\mu} I(\Pi^t, \bar{R}_t)\right] = \mathsf{E}\left[\sum_{t=1}^{\mu} I(\bar{x}_t, \Omega^t)\right]$, *where $\bar{x}_t, \bar{R}_t$ is the pair of point and range chosen in round t and the expectation is taken over the random sequence of pairs chosen in all rounds.*

PROOF. For $t \geq 1$, let $\mathcal{S}^t$ be the set of all point-range-pair sequences of length t, i.e.,

$$\mathcal{S}^t = \{\langle (x_1, R_1), \ldots, (x_t, R_t) \rangle | x_i \in \mathsf{X}, R_i \in \mathcal{R}, 1 \leq i \leq t\}.$$

If we fix a sequence $S \in \mathcal{S}^{t-1}$, then the distributions Π^t, Ω^t are fixed, which we denote by $\Pi^t_{|S}, \Omega^t_{|S}$.

$$\begin{aligned}
\mathsf{E}[I(\Pi^t, \bar{R}_t)] &= \sum_{S \in \mathcal{S}^{t-1}} \Pr(S) \sum_{R_t \in \mathcal{R}} \Pr(R_t|S) I(\Pi^t_{|S}, R_t) \\
&= \sum_{S \in \mathcal{S}^{t-1}} \Pr(S) \sum_{\substack{x_t \in \mathsf{X} \\ R_t \in \mathcal{R}}} \Pr(x_t, R_t|S) I(x_t, R_t) \\
&= \sum_{S \in \mathcal{S}^{t-1}} \Pr(S) \sum_{x_t \in \mathsf{X}} \Pr(x_t|S) I(x_t, \Omega^t_{|S}) \\
&= \mathsf{E}[I(x_t, \Omega^t)].
\end{aligned}$$

The lemma now follows from the linearity of expectation. □

We are now ready to prove that the distribution $\tilde{\Pi}$ on X implied by $\tilde{\mathsf{X}}$ is close to Π^*. For a range $R \in \mathcal{R}$, let

$$\lambda_R = \tfrac{1}{\mu} \sum_{t=1}^{\mu} I(\bar{x}_t, R) = I(\tilde{\Pi}, R)$$

and

$$\tilde{\lambda} = \min_{R \in \mathcal{R}} \lambda_R.$$

Note that λ_R, for every $R \in \mathcal{R}$, and $\tilde{\lambda}$ are random variables.

LEMMA 4.5. *(i)* $\mathsf{E}[\tilde{\lambda}] \geq \lambda^*/4 \geq \frac{1}{4k}$, *and*
(ii) $\Pr[\tilde{\lambda} > \mathsf{E}[\tilde{\lambda}]/2] \geq 1/7$.

PROOF. Using Lemmas 4.3 and 4.4 and Corollary 4.2,

$$\begin{aligned}
\mathsf{E}[\tilde{\lambda}] &\geq \frac{1}{\mu \ln 4} \mathsf{E}\left[\sum_{t=1}^{\mu} I(\bar{x}_t, \Omega^t)\right] - \frac{\ln m}{\mu \ln 2} \qquad \text{(By Lemma 4.3)} \\
&= \frac{1}{\mu \ln 4} \mathsf{E}\left[\sum_{t=1}^{\mu} I(\Pi^t, \bar{R}_t)\right] - \frac{\ln m}{\mu \ln 2} \qquad \text{(By Lemma 4.4)} \\
&\geq \frac{1}{\mu \ln 4} \mathsf{E}\left[\ln 2 \sum_{t=1}^{\mu} I(\Pi^*, \bar{R}_t) - \ln n\right] - \frac{\ln m}{\mu \ln 2} \\
&\qquad \text{(By Corollary 4.2)} \\
&= \frac{1}{2\mu} \mathsf{E}\left[\sum_{t=1}^{\mu} I(\Pi^*, \bar{R}_t)\right] - \frac{1}{\mu \ln 4} \ln(m^2 n).
\end{aligned}$$

However,

$$I(\Pi^*, \bar{R}_t) \geq \min_{R \in \mathcal{R}} I(\Pi^*, R) = \lambda^*$$

and

$$\mu = \frac{2}{\ln 2} k \ln(m^2 n).$$

Therefore,

$$\mathsf{E}[\tilde{\lambda}] \geq \frac{\lambda^*}{2} - \frac{1}{4k} \geq \frac{\lambda^*}{4} \geq \frac{1}{4k}$$

because $\lambda^* \geq 1/k$. This proves part (i) of the lemma.

We now prove (ii). By (1),

$$\tilde{\lambda} = \min_{R \in \mathcal{R}} \lambda_R = \min_{R \in \mathcal{R}} I(\tilde{\Pi}, R) \leq \lambda^*.$$

Also, according to part (i),

$$\mathsf{E}[\tilde{\lambda}] \geq \lambda^*/4.$$

Let $\mathsf{p} = \Pr[\tilde{\lambda} \leq E[\tilde{\lambda}]/2]$. Then

$$\mathsf{E}[\tilde{\lambda}] \leq \mathsf{p}\mathsf{E}[\tilde{\lambda}]/2 + \lambda^*(1-\mathsf{p}) \leq \mathsf{p}\mathsf{E}[\tilde{\lambda}]/2 + 4\mathsf{E}[\tilde{\lambda}](1-\mathsf{p}),$$

and we obtain $\mathsf{p} \leq 6/7$. This proves part (ii) of the lemma. □

Lemma 4.5 implies that, with constant probability, $\tilde{\lambda} > \frac{1}{8k}$, i.e., every range is $\frac{1}{8k}$-heavy with respect to $\tilde{\Pi}$. Thus, the algorithm indeed returns a hitting set of Σ.

Fast implementation and running time. To expedite the algorithm, we perform the following preprocessing steps. First, as in Section 3, we compute a $\frac{1}{k}$-net N_0 of Σ, set $\mathsf{X} = \mathsf{X} \setminus \mathsf{N}_0$ and $\mathcal{R} = \{R \mid R \cap \mathsf{N}_0 = \varnothing\}$, i.e., remove the ranges hit by N_0. Next, we choose a set $\mathsf{N}_1 \subseteq \mathsf{X}$ of $O(k)$ points so that any point in $\mathsf{X} \setminus \mathsf{N}_1$ intersects at most m/k ranges of $\{R \mid R \cap \mathsf{N}_1 = \varnothing\}$. Since Σ has a hitting set of size at most k, such a set N_1 always exists. We now set $\mathsf{X} = \mathsf{X} \setminus \mathsf{N}_1$, $\mathcal{R} = \{R \mid R \cap \mathsf{N}_1 = \varnothing\}$. After the preprocessing, each point of X lies in at most m/k ranges, and each range of $\mathcal{R}$ contains at most n/k points. The algorithm returns $\mathsf{N}_0 \cup \mathsf{N}_1 \cup \mathsf{N}$ as the hitting set of Σ.

For this algorithm, we need the net finder of (P1), the range-reporting data structure of (P3) and two other procedures:

(P4) A dual range-reporting data structure that reports all s ranges containing a query point in $O(\tau(m) + s)$ time, and that can be constructed in $O(m\tau(m))$ time.

(P5) An algorithm that performs the second preprocessing step in $\varphi(N)$ time.

Finally, we build a balanced binary search tree $\mathcal{T}_{\mathsf{X}}$ on X and another one $\mathcal{T}_{\mathcal{R}}$ on $\mathcal{R}$ so that the weights of a point in X and of a range in $\mathcal{R}$ can be updated in $O(\log n)$ and $O(\log m)$ time, respectively, and so that a random element of X or $\mathcal{R}$ can be chosen within the same time; see e.g. [3].

In round i, we report all points of $\bar{R}_i$ in $O(\tau(n) + n/k)$ time using (P3) and update the weights of these points in the binary tree $\mathcal{T}_{\mathsf{X}}$ in $O(\frac{n}{k}\log n)$ time. Similarly we report the ranges of $\mathcal{R}$ that contain $\bar{x}_i$ in $O(\tau(m) + m/k)$ time using (P4) and update their weights in $\mathcal{T}_{\mathcal{R}}$ in $O(\frac{m}{k}\log m)$ time. Hence, round i takes $O(\tau(N) + k^{-1}N\log(N))$ time. Summing over all μ rounds, adding the preprocessing time, the algorithm takes $O(\varphi(N) + k\log(N)\tau(N) + N\log^2(N)) = O(\varphi(N) + N\log(N)\tau(N))$ time. Since the algorithm succeeds with probability at least 1/7, it is repeated $O(1)$ expected times.

We perform backward exponential search on the value of k as follows. Initially, we set $k = n$. Given the current value of k, we run the algorithm for $\Theta(\log N)$ times. If the algorithm returns a hitting set H, we remember H, halve the value of k, and continue. Otherwise, we stop and return the smallest hitting set remembered. At most $\log n$ values of k are examined during the backward exponential search. If $k \geq \kappa$, the expected number of times the algorithm is run for a fixed value of k is $O(1)$ because each iteration returns a valid hitting set with constant probability. If $k < c\kappa/g(\kappa)$ for some constant c, no hitting set of size $O(kg(k)) < \kappa$ exists, and thus the algorithm will terminate in $\Theta(\log N)$ iterations. For $k \in [c\kappa/g(\kappa), \kappa)$, a hitting set may be found at the end of the $\Theta(\log N)$ iterations in the worst case. Thus, the algorithm is repeated $O(\log(N)\log(g(\kappa)))$ times in the worst case, for this range of k.

Unlike Section 3, the weight of points can become quite large and those of ranges can become quite small, so algebraic complexity of the algorithm can be large. However, it suffices to maintain the weights approximately using $O(\log N)$ bits as it introduces a relative error of at most $O(1/N^2)$ at each node of the trees $\mathcal{T}_{\mathsf{X}}$ and $\mathcal{T}_{\mathcal{R}}$. We omit the analysis of this approximation and conclude the following.

THEOREM 4.6. *Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be a finite range space with $|\mathsf{X}| + |\mathcal{R}| = N$.*

(i) Suppose procedures (P1) and (P3)-(P5) exist for Σ, then an $O(g(\kappa))$-approximate hitting set of Σ can be computed in expected time $O((\varphi(N) + N\tau(N)\log N)\log N\log(g(\kappa)))$, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.

(ii) Suppose procedures (P1) and (P3)-(P5) exist for $\Sigma^\perp$, then an $O(g(\chi))$-approximate set cover of Σ can be computed in expected time $O((\varphi(N) + N\tau(N)\log N)\log N\log(g(\chi)))$, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.

Remarks. (i) The algorithm described here can be viewed as a "kinder and gentler" version of the greedy algorithm. A point lying in many "uncovered" ranges is likely to have higher weight and thus higher probability of being chosen. Instead of removing the ranges covered by a chosen point, we simply halve their weights. By the time algorithm stops, the weight of every range is very small.

(ii) It is worth contrasting this algorithm with the previous one. This one is simpler and does not require a dynamic data structure for range counting to verify whether a range is light. However, it requires a range-reporting data structure for both Σ and $\Sigma^\perp$, and an additional preprocessing step to compute N_1 so that no point lies in more than m/k ranges.

5. FAST ALGORITHMS FOR GEOMETRIC INSTANCES

In this section, we show that the algorithms described in Sections 3 and 4 yield near-linear hitting-set and set-cover algorithms for a number of geometric range spaces, for which the required procedures (P1)-(P5) can be performed efficiently. Let X be a set of n points in $\mathbb{R}^d$ and $\mathcal{R}$ a set of m geometric shapes such as rectangles, balls, and simplices. As mentioned in Introduction, we will use $R \in \mathcal{R}$ to denote a shape as well as the subset $\mathsf{X} \cap R$. It will be clear from the context which of the two we are referring to.

We mainly describe the implementations of the second algorithm, which require procedures (P1) and (P3)-(P5) and give the desired approximation ratios with high probability. One can also use the first algorithm, which may add a logarithmic factor in the running time in some cases, but it always returns a hitting set or set cover with the approximation ratio stated in the corresponding theorems.

Rectangles in 2D and 3D. Let $\mathcal{R}$ be a set of m axis-parallel rectangles in $\mathbb{R}^d$ for $d = 2, 3$. Aronov *et al.* [6] have shown that an ε-net of Σ of size $O((1/\varepsilon)\log\log(1/\varepsilon))$ can be constructed in $O(n\log^{d-1} n)$ randomized expected time, thus $g(\kappa) = \log\log(\kappa)$ and $\varphi(N) = O(N\log^{d-1} N)$ in procedure (P1).

For the range reporting data structure in (P3), we construct a d-dimensional range tree on X in $O(n\log^{d-1} n)$ time and a range reporting query can be answered in $O(\log^{d-1} n + s)$ time, where

s is the output size. For (P4), we need a dual range-reporting data structure that reports all rectangles of $\mathcal{R}$ containing a query point. This can be done in $O(\log^{d-1} m + s)$ time, where s is the output size, after $O(m \log^{d-1} m)$ preprocessing [16]. Hence, $\tau(n) = O(\log^{d-1} n)$.

Finally, we also need an algorithm for (P5) that computes a set N_1 of $O(k)$ points so that no point in $\mathsf{X} \setminus \mathsf{N}_1$ lies in more than m/k rectangles of $\{R \in \mathcal{R} \mid R \cap \mathsf{N}_1 = \varnothing\}$. This can be accomplished using a d-dimensional segment tree on $\mathcal{R}$, which can be constructed in $O(m \log^d m)$ time. For each point $p \in \mathsf{X}$, the segment tree is queried in $O(\log^d m)$ time to check whether p lies in more than m/k rectangles of $\mathcal{R}$. If so, we add p to N_1 and delete all the rectangles of $\mathcal{R}$ that contain p from the segment tree. Obviously, at most k points are reported, and the time spent is $\varphi(N) = O(N \log^d N)$. [3]

Plugging the bounds of $\tau(\cdot), g(\cdot)$ and $\varphi(\cdot)$ in Theorem 4.6, we obtain the following.

THEOREM 5.1. *Let X be a set of n points in $\mathbb{R}^d$ and $\mathcal{R}$ a set of m rectangles in $\mathbb{R}^d$, for $d = 2, 3$, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(\log\log\kappa)$-approximate hitting set of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^{d+1} N \log\log\log\kappa)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

For the set cover problem, an ε-net of $\Sigma^\perp$ required in (P1) can be computed simply by choosing a random sample of size $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$, with probability at least 1/2 [34]. The procedures (P3) and (P4) remain the same as (P4) and (P3) in the hitting-set case. The set $\mathsf{N}_1 \subseteq \mathcal{R}$ in procedure (P5) can be computed in a similar way, except that a d-dimensional range tree, instead of a segment tree, is used. The running time is $\varphi(N) = O(N \log^d N)$. Hence, we obtain the following.

THEOREM 5.2. *Let X be a set of points in $\mathbb{R}^d$ and $\mathcal{R}$ a set of rectangles in $\mathbb{R}^d$, for $d = 2, 3$, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(\log\chi)$-approximate set cover of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^{d+1} N \log\log\log\chi)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

Halfspaces in 3D. Let X be a set of n points and $\mathcal{R}$ a set of m halfspaces in $\mathbb{R}^3$. Using the algorithm by Matoušek [39], an ε-net of $\Sigma = (\mathsf{X}, \mathcal{R})$ of size $O(1/\varepsilon)$ can be computed in $O(n \log n)$ expected time, thus $g(\cdot) = 1$ and $\varphi(N) = O(N \log N)$ in (P1). Using the data structure by Afshani and Chan [1], the halfspace range-reporting query in (P3) can be answered in $O(\log n + s)$ time, after $O(n \log n)$ preprocessing. By duality transform, the dual range-reporting query in (P4) can be answered using the same data structure in $O(\log m + s)$ time, after $O(m \log m)$ preprocessing. Hence $\tau(n) = O(\log n)$.

Finally, computing N_1 of (P5) is more involved in this case. We basically use the approach described in Agarwal *et al.* [3, Section 3]. The algorithm works in $\lceil \log k \rceil$ rounds. At the beginning of round i, we have a set $\mathsf{X}_i \subseteq \mathsf{X}$ of points and a subset $\mathcal{R}_i \subseteq \mathcal{R}$ of halfspaces. The maximum depth of a point in X_i is at most $\ell_i = m/2^i$. We set $r_i = 2^{i+1}$ and compute a $(1/r_i)$-cutting Ξ_i of at most ℓ_i levels of the arrangment of $\mathcal{R}_i$; $|\Xi_i| = O(2^i)$. Since the maximum depth of a point of X_i is ℓ_i, each point of X_i lies in a simplex of Ξ_i. For each simplex $\Delta \in \Xi_i$, if $\mathsf{X}_i \cap \Delta \neq \varnothing$, we choose one point from Δ and add it to N_1. Let P_i be the set of points chosen in round i, and let $\bar{\mathcal{R}}_i = \{R \mid R \cap P_i \neq \varnothing\}$. We set $\mathsf{X}_{i+1} = \mathsf{X}_i \setminus P_i$ and $\mathcal{R}_{i+1} = \mathcal{R}_i \setminus \bar{\mathcal{R}}_i$. Following the algorithm described in [3], Ξ_i, P_i, and $\bar{\mathcal{R}}_i$ can be computed in $O((|\mathsf{X}_i|+|\mathcal{R}_i|)\log^2 r_i) = O((|\mathsf{X}_i|+|\mathcal{R}_i|)i^2)$ expected time. Since each simplex $\Delta \in \Xi_i$ intersects the boundary plane of at most $|\mathcal{R}_i|/2^{i+1}$ halfspaces, it can be checked that any point of X_{i+1} lies in at most $|\mathcal{R}_i|/2^{i+1} \leq m/2^{i+1}$ halfspaces of $\mathcal{R}_{i+1}$.

The total time spent in this process over all $i \leq \log k$ is $O((m + n)\log^3 k)$, and $|\mathsf{N}_1| = O(k)$. Hence $\varphi(N)$ becomes $O(N \log^3 N)$.

Using the duality transform, the set-cover problem in the primal space becomes the hitting-set problem in the dual space. Plugging the bounds in Theorem 4.6, we obtain the following.

THEOREM 5.3. *Let X be a set of points in $\mathbb{R}^3$ and $\mathcal{R}$ a set of halfspaces in $\mathbb{R}^3$, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(1)$-approximate hitting set or set cover of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^4 N)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

By a standard lifting transform [24], we have the following result for disks in $\mathbb{R}^2$.

COROLLARY 5.4. *Let X be a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ a set of disks in $\mathbb{R}^2$, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(1)$-approximate hitting set or set cover of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^4 N)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

Fat triangles. Let $\mathsf{X} \subseteq \mathbb{R}^2$ be a set of n points and $\mathcal{R}$ a set of α-fat triangles, for some constant $\alpha \geq 1$, i.e., the aspect ratio of every triangle in $\mathcal{R}$ is at most α. An ε-net of Σ in procedure (P1) can be obtained by randomly sampling $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ points from X. A range-reporting query (P3) for Σ can be answered in $O(\log^3 n + s)$ time, after $O(n \log^3 n)$ preprocessing [45, Section 4.5], and a dual range-reporting query (P4) for Σ can also be answered in $O(\log^3 m + s)$ time [2]. The set N_1 in (P5) for Σ can be computed using the same method as that for 3D halfspaces [3, Section 3]; the running time is $O(N \log^2 N \log^* N)$ using the improved bound on the union of fat triangles [5]. Thus, $g(1/\varepsilon) = \log(1/\varepsilon)$, $\tau(N) = O(\log^3 N)$, $\varphi(N) = O(N \log^2 N \log^* N)$, and we obtain the following.

THEOREM 5.5. *Let X be a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ a set of α-fat triangles, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(\log\kappa)$-approximate hitting set of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^5 N \log\log\kappa)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

For the set cover problem, an ε-net is computed by randomly sampling $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ triangles from $\mathcal{R}$. To compute $\mathsf{N}_1 \subseteq \mathcal{R}$ for $\Sigma^\perp$, a (9/8)-approximate range-counting data structure $\mathcal{D}$ for fat triangles is used, which can be built in $O(n \log^3 n)$ time by applying the range emptiness data structure [45] as a black box to the data structure in [7, Section 5], and can answer a query in $O(\log^4 n)$ time. Then $\log k$ steps are performed. In the i-th step, we build such a data structure $\mathcal{D}$ on X and for each triangle $R \in \mathcal{R}$, we check whether R contains more than $n/2^i$ points of X. If so, we add R to N_1 and remove the points in R from X. Instead of removing those points from $\mathcal{D}$, we insert them into another (9/8)-approximate range-counting data structure $\mathcal{D}'$ using the logarithmic method [12], and the answer of a range-counting query is the difference of the query answers from $\mathcal{D}$ and $\mathcal{D}'$. It can be checked that the error of the query is at most $n/2^{i+1}$ and $O(2^i)$ points are added to N_1 in the i-th step. Thus, $|\mathsf{N}_1| = O(k)$ and the total time of this preprocessing step is $O(N \log^6 N)$. We conclude the following.

THEOREM 5.6. *Let X be a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ a set of α-fat triangles, with $|\mathsf{X}| + |\mathcal{R}| = N$. An $O(\log\chi)$-approximate set cover of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^7 N \log\log\chi)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

[3] Using a sweep-plane technique, the running time of the algorithm for (P5) can be improved by a logarithmic factor. However, since this step is not the bottleneck, we use the above approach, which is simpler.

Remarks. (i) Better bounds on the sizes of primal and dual ε-nets exist for fat triangles [6]. We use the simple random-sampling approach to compute an ε-net since it can be done in near-linear time. With a careful implementation, it might be possible to implement the ε-net construction algorithms in [6] in near-linear time, in which case the approximation ratios of the hitting-set and set-cover problems for fat triangles can be improved.

(ii) The range-searching data structures for $\Sigma^{\perp}$ works even if $\mathcal{R}$ is a set of locally fat objects or convex pseudo-disks in $\mathbb{R}^2$ of constant description complexity. Using the first algorithm, $O(\log \chi)$-approximate set covers for these range spaces can be computed in near-linear time.

6. INDEPENDENT SET FOR DISKS

Given a finite range space $\Sigma = (\mathsf{X}, \mathcal{R})$, an *independent set* is a subset $\mathcal{I} \subseteq \mathcal{R}$ of ranges such that for any $R_1, R_2 \in \mathcal{I}$, $R_1 \cap R_2 \cap \mathsf{X} = \varnothing$, i.e., any point of X lies in at most one range of $\mathcal{I}$. The goal is to find a maximum-size independent set (MIS).

The main observation is that, an optimal mixed strategy for Bob in the second algorithm gives a fractional solution for the independent-set problem. So we proceed as follows. Suppose we have an integer k such that $\delta/2 < k \leq \delta$, where δ is the size of an optimal independent set. We set $\mu = 3k \log N$ and run the second algorithm for μ rounds. Let $\mathcal{F} = \langle \bar{R}_1, \ldots, \bar{R}_\mu \rangle$ be the sequence of ranges chosen by Bob. Let $\tilde{\Omega}$ be the distribution of $\mathcal{R}$ induced by $\mathcal{F}$, i.e., $\Pr(R \sim \tilde{\Omega})$ is proportional to the number of times R appears in $\mathcal{F}$. Using Lemmas 4.1, 4.3 and 4.4, we can prove that

$$\mathsf{E}\left[\max_{x \in \mathsf{X}} I(x, \tilde{\Omega})\right] \leq 3/k.$$

By Markov inequality, with probability at least 1/2,

$$\max_{x \in \mathsf{X}} I(x, \tilde{\Omega}) \leq 6/k.$$

Let y_i be the variable corresponding to the range $R_i \in \mathcal{R}$. Then by setting $y_i = \frac{k}{6} \Pr(R_i \sim \tilde{\Omega})$, with probability at least 1/2, we obtain a fractional solution for the independent-set problem with objective value $\sum y_i = k/6$. Next, we convert the fractional solution into an integral solution using the approach of Chan and Har-Peled [15]: we process the ranges of $\mathcal{R}$ in an arbitrary order, and maintain an independent set $\mathcal{I}$ of ranges. A range R_i is processed as follows: if R_i does not contain any of the points lying in a range of the current $\mathcal{I}$, then R_i is added to $\mathcal{I}$ with probability y_i. Otherwise, it is discarded.

Chan and Har-Peled [15] have shown that if $\mathcal{R}$ is a set of disks in $\mathcal{R}^2$ and the fractional solution has value ν, then the above rounding scheme returns an independent set of size $\Omega(\nu)$ with constant probability. Hence, the set $\mathcal{I}$ has size $\Omega(k)$ with constant probability.

In order to expedite the execution of the algorithm, we perform the same preprocessing as in Section 4 to compute a set N of at most $2k/5$ points such that in the range space $\bar{\Sigma} = (\mathsf{X} \setminus \mathsf{N}, \{R \in \mathcal{R} \mid R \cap \mathsf{N} = \varnothing\})$, each range of $\bar{\Sigma}$ contains $O(n/k)$ points and each point of $\mathsf{X} \setminus \mathsf{N}$ lies in $O(m/k)$ ranges. Furthermore, one can argue that if Σ has an independent set of size δ, then $\bar{\Sigma}$ has one of size at least $3\delta/5$. Hence, we run the above algorithm on $\bar{\Sigma}$, and the total time for computing the fractional solution, including preprocessing, is $O(N \log^4 N)$.

To convert the fractional solution into an integral one, a dynamic insertion-only disk-emptiness data structure $\mathcal{D}$ is used, which can be implemented by using a static disk-emptiness data structure [2] with the logarithmic method [12]. Initially, $\mathcal{D}$ contains no point. For each range R_i, we perform a disk-emptiness query in $\mathcal{D}$ in $O(\log^2 n)$ time to determine whether R_i can be added to $\mathcal{I}$; if it is added, every point in R_i is inserted into $\mathcal{D}$ in $O(\log^2 n)$ amortized time. The total running time is $O(N \log^2 N)$. Hence, we obtain the following.

THEOREM 6.1. *Let X be a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ a set of disks in $\mathbb{R}^2$, with $|\mathsf{X}|+|\mathcal{R}| = N$. An $O(1)$-approximate independent set of $(\mathsf{X}, \mathcal{R})$ can be computed in $O(N \log^4 N)$ expected time, with probability at least $1 - \frac{1}{N^{\Omega(1)}}$.*

7. CONCLUSION

In this paper, we presented two simple, efficient algorithms, based on the MW method, to compute small-size hitting sets and set covers for range spaces with finite VC-dimension. The first algorithm is Las Vegas and the second one is Monte Carlo. They yield near-linear time algorithms for many geometric range spaces. Our second algorithm can be used to compute an $O(1)$-approximate solution for the (discrete) maximum independent set problem for disks in near-linear time. We conclude by mentioning two open problems:

(i) Can our approach be extended to compute a multi-cover of range spaces [18]?

(ii) Is there an $O(\log \log \chi)$-approximate algorithm for computing the set cover when $\mathcal{R}$ is a set of rectangles in $\mathbb{R}^2$?

(iii) Let X be a set of points and $\mathcal{R}$ a set of unit squares in $\mathbb{R}^2$. Can a small-size hitting set of $\mathcal{R}$ be maintained under insertion/deletion of points and squares?

(iv) Is there a fast $O(1)$-approximate algorithm for the independent set problem when $\mathcal{R}$ is a set of rectangles in $\mathbb{R}^2$. The best known algorithm computes an $O(\log \log n)$-approximate independent set [14].

Acknowledgments. Work on this paper is supported by NSF under grants CCF-09-40671, CCF-10-12254, and CCF-11-61359, by Grant 2012/229 from the U.S.-Israel Binational Science Foundation, by an ARO contract W911NF-13-P-0018, and by an ERDC contract W9132V-11-C-0003.

8. REFERENCES

[1] P. Afshani and T. M. Chan, Optimal halfspace range reporting in three dimensions, *Proc. 20th Annual ACM-SIAM Sympos. Discrete Algorithms*, 2009, pp. 180–186.

[2] P. K. Agarwal and J. Erickson, Geometric range searching and its relatives, *Advances in Discrete and Computational Geometry* (B. Chazelle, J. Goodman, and R. Pollack, eds.), (1998), pp. 1–56.

[3] P. K. Agarwal, E. Ezra, and M. Sharir, Near-linear approximation algorithms for geometric hitting sets, *Algorithmica*, 63 (2012), 1–25.

[4] N. Alon, A non-linear lower bound for planar epsilon-nets, *Discrete Comput. Geom.*, 47 (2012), 235–244.

[5] B. Aronov, M. de Berg, E. Ezra, and M. Sharir, Improved bounds for the union of locally fat objects in the plane, *to appear in SIAM J. Comput.*

[6] B. Aronov, E. Ezra, and M. Sharir, Small-size ε-nets for axis-parallel rectangles and boxes, *SIAM J. Comput.*, 39 (2010), 3248–3282.

[7] B. Aronov and S. Har-Peled, On approximating the depth and related problems, *SIAM J. Comput.*, 38 (2008), 899–921.

[8] S. Arora, E. Hazan, and S. Kale, Fast algorithms for approximate semidefinite programming using the multiplicative weights update method, *Proc. 46th Annual IEEE Sympos. Found. Comput. Sci.*, 2005, pp. 339–348.

[9] S. Arora, E. Hazan, and S. Kale, $o(\sqrt{\log n})$ approximation to sparsest cut in $\tilde{O}(n^2)$ time, *SIAM J. Comput.*, 39 (2010), 1748–1771.

[10] S. Arora, E. Hazan, and S. Kale, The multiplicative weights update method: a meta-algorithm and applications, *Theory of Comput.*, 8 (2012), 121–164.

[11] B. Awerbuch and R. Khandekar, Stateless distributed gradient descent for positive linear programs, *SIAM J. Comput.*, 38 (2009), 2468–2486.

[12] J. L. Bentley and J. B. Saxe, Decomposable searching problems I. Static-to-dynamic transformation, *J. Algorithms*, 1 (1980), 301–358.

[13] H. Brönnimann and M. Goodrich, Almost optimal set covers in finite VC-dimension, *Discrete Comput. Geom.*, 14 (1995), 463–479.

[14] P. Chalermsook and J. Chuzhoy, Maximum independent set of rectangles, *Proc. 20th Annual ACM-SIAM Symp. Discrete Algorithms*, 2009, pp. 892–901.

[15] T. M. Chan and S. Har-Peled, Approximation algorithms for maximum independent set of pseudo-disks, *Proc. 25th Annual Symp. Comput. Geom.*, 2009, pp. 333–340.

[16] B. Chazelle, Filtering search: A new approach to query answering, *SIAM J. Comput.*, 15 (1986), 703–724.

[17] B. Chazelle and E. Welzl, Quasi-optimal range searching in spaces of finite VC-dimension, *Discrete Comput. Geom.*, 4 (1989), 467–489.

[18] C. Chekuri, K. L. Clarkson, and S. Har-Peled, On the set multicover problem in geometric settings, *ACM Trans. Algorithms*, 9 (2012), 9:1–9:17.

[19] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs, *Proc. 43rd Annual ACM Sympos. Theory of Comput.*, 2011, pp. 273–282.

[20] K. Clarkson, Algorithms for polytope covering and approximation, in: *Proc. 7th Workshop Algorithms Data Struct.*, 1993, pp. 246–252.

[21] K. L. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is small, *J. ACM*, 42 (1995), 488–499.

[22] K. L. Clarkson and K. R. Varadarajan, Improved approximation algorithms for geometric set cover, *Discrete Comput. Geom.*, 37 (2007), 43–58.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3rd ed.)*, McGraw-Hill, 2009.

[24] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 2008.

[25] E. Ezra, Small-size relative (p, ε)-approximations for well-behaved range spaces, *Proc. 29th Annual Symp. Comput. Geom.*, 2013, pp. 233–242.

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

[27] D. P. Foster and R. Vohra, Regret in the on-line decision problem, *Games and Economic Behavior*, 29 (1999), 7–35.

[28] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Inf. Process. Lett.*, 12 (1981), 133 – 137.

[29] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *J. Comput. System Sci.*, 55 (1997), 119 – 139.

[30] Y. Freund and R. E. Schapire, Adaptive game playing using multiplicative weights, *Games and Economic Behavior*, 29 (1999), 79 – 103.

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[32] M. D. Grigoriadis and L. G. Khachiyan, A sublinear-time randomized approximation algorithm for matrix games, *Oper. Res. Lett.*, 18 (1995), 53 – 58.

[33] S. Har-Peled, *Geometric Approximation Algorithms*, American Mathematical Society, Boston, MA, USA, 2011.

[34] D. Haussler and E. Welzl, ε-nets and simplex range queries, *Discrete Comput. Geom.*, 2 (1987), 127–151.

[35] C. Koufogiannakis and N. Young, Beating simplex for fractional packing and covering linear programs, *Proc. 48th Annual IEEE Symp. Found. Comput. Sci.*, 2007, pp. 494–504.

[36] S. Lauen, Geometric set cover and hitting sets for polytopes in R^3, *25th Int. Symp. Theoretical Aspects of Comput. Sci.*, Vol. 1, 2008, pp. 479–490.

[37] P. Long, Using the pseudo-dimension to analyze approximation algorithms for integer programming, in: *Proc. 15th Workshop Algorithms Data Struct.*, 2001, pp. 26–37.

[38] J. Matoušek, Efficient partition trees, *Discrete Comput. Geom.*, 8 (1992), 315–334.

[39] J. Matoušek, Reporting points in halfspaces, *Comput. Geom. Theory Appl.*, 2 (1992), 169–186.

[40] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

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

[42] J. Pach and G. Tardos, Tight lower bounds for the size of epsilon-nets, *J. American Mathematical Society*, 26 (2013), 645–658.

[43] S. A. Plotkin, D. B. Shmoys, and E. Tardos, Fast approximation algorithms for fractional packing and covering problems, *Math. Oper. Res.*, 20 (1995), pp. 257–301.

[44] E. Pyrga and S. Ray, New existence proofs for ε-nets, *Proc. 24th Annual Symp. Comput. Geom.*, 2008, pp. 199–207.

[45] H. Shaul, *Range Searching: Emptiness, Reporting, and Approximate Counting*, Ph.D. Thesis, Tel Aviv University, 2011.

[46] K. Varadarajan, Epsilon nets and union complexity, *Proc. 25th Annual Symp. Comput. Geom.*, 2009, pp. 11–16.

[47] E. Welzl, Partition trees for triangle counting and other range searching problems, *Proc. 4th Annual Symp. Comput. Geom.*, 1988, pp. 23–33.

[48] N. Young, Sequential and parallel algorithms for mixed packing and covering, *Proc. 42nd Annual IEEE Symp. Found. of Comput. Sci.*, 2001, pp. 538–546.