Near-Linear Algorithms for Geometric Hitting Sets and Set Covers

Pankaj K. Agarwal
Dept. of Computer Science
Duke University
Durham, NC, USA
pankaj@cs.duke.edu

Jiangwei Pan
Dept. of Computer Science
Duke University
Durham, NC, USA
jwpan@cs.duke.edu

ABSTRACT

Given a finite range space $\Sigma = (\mathsf{X}, \mathcal{R})$, we present two simple algorithms, based on the multiplicative-weight method, for computing a small-size hitting set or set cover of Σ. One of them is a simpler variant of the algorithm in [9] but more efficient to implement, and the second algorithm can be viewed as a two-player zero-sum game. These algorithms lead to near-linear algorithms for computing a small-size hitting set or set cover for a number of geometric range spaces such as when X is a set of points in $\mathbb{R}^2$ and $\mathcal{R}$ is a set of rectangles, disks, fat objects, or pseudo-disks. 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 in $\mathbb{R}^2$ and $\mathcal{R}$ is a set of disks in $\mathbb{R}^2$; here $N = |\mathsf{X}| + |\mathcal{R}|$. No such algorithm was known earlier.

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 $|\mathsf{X}| = n, |\mathcal{R}| = m$, and $N = n + m$. A subset $\mathsf{H} \subseteq \mathsf{X}$ is called a *hitting set* of Σ if it 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 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 [25] 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.

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 $\mathbb{R}^2$ [22]. This has led to development of polynomial-time approximation algorithms for these problems. Traditionally, the focus has been on developing algorithms with the smallest possible approximation ratio that run in polynomial time. 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 [17]. 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 [20]. However, by exploiting the underlying geometry, polynomial-time algorithms with better approximation factors can be obtained for many geometric range spaces; see [5, 16, 30, 31, 33, 36, 37] 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.

Given a parameter $\varepsilon \in (0, 1]$, an *ε-net* for range space $\Sigma = (\mathsf{X}, \mathcal{R})$ is a subset $\mathsf{N} \subseteq \mathsf{X}$ such that N 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 [27] proved that a range space of *finite VC-dimension* δ (see [27] for the definition) 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 in the plane, or translates of a fixed convex polytope in $\mathbb{R}^3$ [29, 30, 36]. Aronov *et al.* [5] 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 [34] shows that this bound is worst-case tight.

Building on a technique by Clarkson [14], Brönnimann and Goodrich [9] 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 (see Section 2 for the definition), their algorithm also extends to set cover. Hence, if Σ has finite VC-dimension, then a hitting set of size $O(\kappa \log \kappa)$ 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 $\mathbb{R}^3$. A related technique by Even et al. [19], which exploits LP-relaxation, results in a similar approximation factor. See [16, 5, 37] for improved approximation ratios for set covers of geometric range spaces.

The algorithms by Clarkson [14] and by Brönnimann and Goodrich [9] are based on 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 [23], optimization [13], game theory [24, 26], on-line algorithms [21] and computational geometry. For example, MW algorithms are used for boosting [23], packing and covering LPs [28, 35, 39], semidefinite programming [6], and sparsest cuts for graphs [7]. In computational geometry, besides hitting-set/set-cover, they have been used for linear programming [15] and range search [38]. See the survey [8] by Arora *et al.* 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 space.

Our results. We present two simple algorithms for computing a small-size hitting set or set cover of Σ, each based on the MW method, when the VC-dimension of Σ is finite. Both algorithms 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 [19]). 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. 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 [9]. 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. Using the MW method, the algorithm computes near-optimal mixed strategies for both Alice and Bob. A near-optimal mixed strategy for Alice implies a near-optimal fractional solution to the hitting set, from which we use ε-net to obtain a hitting set of small size. We remark that the MW method has been used to solve the zero-sum game approximately [24, 26], but our algorithm is somewhat different, tailored to the fact that the goal is to compute a hitting set.

We next describe in Section 5 consequences of these algorithms for a number of geometric range spaces. Table 1 summarizes the new results for hitting-sets and set-covers. 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. In all these instances, our algorithms obtain, in near-linear time, the best known approximation ratio (within a constant factor) that can be computed by polynomial-time algorithms using the ε-net approach. 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. The local-search technique by Mustafa and Ray [33] gives a $(1 + \varepsilon)$-approximation algorithm for disks, but its running time is $O(N^{O(\varepsilon^{-2})})$. We remark that our algorithms rely on standard range searching data structures. It might be possible to improve $\log N$ 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 in $\mathbb{R}^2$ and $\mathcal{R}$ is 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 [11]. The algorithm in [11] 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 $\Sigma^\perp$.

A hitting set $\mathsf{H} \subseteq \mathsf{X}$ is called *c-approximate*, for $c \geq 1$, if $|\mathsf{H}| \leq c\kappa(\Sigma)$. Similarly, we call a set cover $\mathcal{C} \subseteq \mathcal{R}$ *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[9] (and Clarkson [14]), we briefly describe their main idea. Let k be an integer such that $k/2 < \kappa \leq k$. We initialize the weight of each point to 1 and repeat the following step until every range is $\frac{1}{2k}$-heavy. Let R be a $\frac{1}{2k}$-light range. We double the weights of all points in R. We refer to this step as a *weight-doubling* step. When the process stops, we return a $\frac{1}{2k}$-net H of Σ (with respect to the final weights). If a $\frac{1}{2k}$-net of size $O(kg(k))$ can be computed efficiently, we obtain a hitting set of size $O(\kappa g(\kappa))$. The following lemma, proved in [9], is the key to the performance of the algorithm. [1]

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.*

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

Ranges	Approximation Ratio		Running Time
	Hitting Set	Set Cover	
Rectangles ($d = 2, 3$)	$O(\log\log \text{OPT})$	$O(\log \text{OPT})$	$O(N \log^{d+1} N)$
Halfspaces ($d = 3$)	$O(1)$	$O(1)$	$O(N \log^3 N)$
Disks	$O(1)$	$O(1)$	$O(N \log^3 N)$
Fat triangles	$O(\log\log \text{OPT})$	$O(\log(\log^* \text{OPT}))$	$O(N \log^5 N)$
Locally fat objects	NA	$O(\log^* \text{OPT})$	$O(N \log^4 N)$
Pseudo-disks	NA	$O(1)$	$O(N \log^4 N)$

Table 1: Summary of results for different geometric range spaces.

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 $k/2 < \kappa \leq k$. Brönnimann-Goodrich algorithm computes a $\frac{1}{2k}$-net in each step to check whether there is a light range; see the original paper for details.

3. HITTING SET IN ROUNDS

As in [9], suppose we have an integer k such that $\kappa \in (k/2, k]$; we perform an exponential search on k. The algorithm is similar to [9] except that it works in rounds. 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 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 in the current round as the 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 stops within $2\log(n/2k) + 1$ rounds because otherwise it would have performed more than μ_k weight-doubling steps.

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 $1 + \frac{1}{2k}$,

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

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

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

Hence, 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. $\square$

By Lemma 3.1, if an ε-net of Σ of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ can be computed, then we obtain a hitting set of size $O(\kappa g(\kappa))$.

Running time. We now analyze the running time of the algorithm. 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 $\mathcal{R} = \{R \mid R \cap \mathsf{N}_0 = \varnothing\}$. So we assume that 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. There are three nontrivial steps in the above algorithm:

(i) Compute an ε-net of Σ. Suppose an ε-net of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ can be computed in time $\varphi(N)$.

(ii) Check whether a range is ε-light. We assume that we have a range-counting data structure that, given a range R, returns YES if R is ε-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 $\tau(n)$ also accounts for the time spent in building the data structure.

(iii) Report all points of X that lie in a range $R \in \mathcal{R}$. We assume that we have a range reporting data structure that takes $\tau(n) + s$ time to report all points of a range, where s is the output size.

Using (ii) and (iii), we can process each range R in $\mathcal{R}$ as follows. First, using (ii) we check in $\tau(n)$ time whether R is $\frac{1}{2k}$-light. If the answer is YES, we use (iii) 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 in (ii). 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 an ε-net of Σ of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ can be computed in $\varphi(N)$ time and range queries in Σ can be answered in $\tau(N)$ time. 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 an ε-net of $\Sigma^\perp$ of size $O(\frac{1}{\varepsilon} g(\frac{1}{\varepsilon}))$ can be computed in $\varphi(N)$ time and range queries in $\Sigma^\perp$ can be answered in $\tau(N)$ time. Then an $O(g(\chi))$-approximate set cover of Σ can be computed in $O((N\tau(N)\log N + \varphi(N))\log \kappa)$ time.*

Remark. In the above algorithm, we can also use an approximate range-counting data structure that returns $\hat{w}(R) \in [w(R), 2w(R))$. We then answer YES if $\hat{w}(R) < \varepsilon w(\mathsf{X})$ and NO otherwise. If the answer is NO, then range R must be $\frac{\varepsilon}{2}$-heavy. It can be checked that such a data structure suffices for our purpose — it affects the size of the hitting set by a factor of at most 2.

4. HITTING SET AS 2-PLAYER GAME

We now describe the second algorithm for computing a hitting set. As earlier, suppose we have an integer k such that $k/2 < \kappa \le k$; we perform a backward exponential search for k. The algorithm now maintains weights on both X and $\mathcal{R}$, and it also works in rounds. For $i \ge 1$, let $\pi^i : \mathsf{X} \to \mathbb{R}_{\ge 0}$ and $\omega^i : \mathcal{R} \to \mathbb{R}_{\ge 0}$ 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., $\Pi^i = \left\langle \frac{\pi^i(x_j)}{\pi^i(\mathsf{X})} \mid 1 \le j \le n \right\rangle$ and $\Omega^i = \left\langle \frac{\omega^i(R_j)}{\omega^i(\mathcal{R})} \mid 1 \le j \le m \right\rangle$.

We set $\mu := \frac{2}{\ln 2} k \ln(m^2 n)$. 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 $\mathsf{Y} = \langle \bar{x}_1, \ldots, \bar{x}_\mu \rangle$ be the multi-set of points chosen by the algorithm. Let $\tilde{\Pi}$ be the distribution on X induced by Y, i.e., if a point $x \in \mathsf{X}$ appears μ_x times in Y, 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. [2]

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: in each round, 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

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

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 [32],

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

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 $\min_{R \in \mathcal{R}} I(\Pi_{\mathsf{H}^*}, R) \ge 1/\kappa$ because $R \cap \mathsf{H}^* \ne \varnothing$ for all $R \in \mathcal{R}$. Hence $\lambda^* \ge 1/\kappa \ge 1/k$. 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 $\min_{R \in \mathcal{R}} I(\Pi^*, R) = \lambda^* \ge 1/k$. We show that $\tilde{\Pi}$ is an approximation of Π^* in the sense that $\min_{R \in \mathcal{R}} I(\tilde{\Pi}, R) \ge \frac{1}{8k}$, and thus a $\frac{1}{8k}$-net of Σ under $\tilde{\Pi}$ is a hitting set of Σ.

[2]In the exponential search, we run the above algorithm $O(\log n)$ times for a fixed k. If the algorithm returns a hitting set H in one of the iterations, we remember H, halve the value of k, and continue. Otherwise, we stop and return the smallest hitting set computed.

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

LEMMA 4.1. *For every $x \in \mathsf{X}$, $\sum_{t=1}^{\mu} I(\Pi^t, \bar{R}_t) \ge -\ln n + \ln 2 \sum_{t=1}^{\mu} I(x, \bar{R}_t)$.*

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

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

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

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

The lemma follows from (2) and (3). $\square$

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

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

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

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

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

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 Y is close to Π^*. For a range $R \in \mathcal{R}$, let $\lambda_R = \frac{1}{\mu}\sum_{t=1}^{\mu} I(\bar{x}_t, R) = I(\tilde{\Pi}, R)$ 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

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

Therefore,

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

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. □

By Lemma 4.5, 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. Now each range in $\mathcal{R}$ has at most n/k points of X. 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 $\mathsf{X} = \mathsf{X} \setminus \mathsf{N}_1$, $\mathcal{R} = \{R \mid R \cap \mathsf{N}_1 = \varnothing\}$. Each point of X lies in at most m/k ranges, and each range 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. Suppose an ε-net of Σ of size $O(\frac{1}{\varepsilon}g(\frac{1}{\varepsilon}))$ and the set N_1 can be computed in $\varphi(N)$ time.

We assume that we have (static) range reporting data structures for both Σ and its dual range space $\Sigma^{\perp}$, i.e., given a range R, we can report all points in R, and given a point x, we can report all ranges that contain x. Suppose each such query can be answered in time $\tau(n) + s$, where s is the output size and $\tau(n) = \Omega(\log n)$.

Finally, we build a balanced binary tree on X and another one on $\mathcal{R}$ so that the weights of X and $\mathcal{R}$ can be updated in $O(\log n)$ and $O(\log m)$ time, respectively, and 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 and update the weights of these points 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 and update their weights 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. Finally, we run it $O(\log N)$ times to perform backward exponential search on the value of k.

We remark that, 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 tree. 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* $|\mathsf{X}| + |\mathcal{R}| = N$, *for which an* ε*-net of* Σ *or* $\Sigma^{\perp}$ *of size* $O(\frac{1}{\varepsilon}g(\frac{1}{\varepsilon}))$ *can be computed in* $\varphi(N)$ *time. Let* $\tau(N) = \Omega(\log N)$ *be the range reporting query time for both* Σ *and* $\Sigma^{\perp}$*. Then an* $O(g(\text{OPT}))$*-approximate hitting set or set cover of* Σ *can be computed in expected time* $O((\varphi(N) + N\log(N)\tau(N))\log N)$*, with high probability, where* OPT *denotes the size of the optimal hitting set or set cover.*

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 the preprocessing step is a little more involved.

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 fast range searching data structures exist. 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. Let $\Sigma = (\mathsf{X}, \mathcal{R})$ be the range space induced by X and $\mathcal{R}$. Set $N = |\mathsf{X}| + |\mathcal{R}|$. We mainly describe the implementations of the second algorithm, which gives 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 rectangles in $\mathbb{R}^d$ for $d = 2, 3$. Aronov *et al.* [5] 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, so $g(1/\varepsilon) = \log\log(1/\varepsilon)$ and $\varphi(n) = O(n\log^{d-1} n)$. For simplicity, we describe the required primitives for $d = 2$ and then mention the bounds for $d = 3$.

Using the dynamic range-tree data structure [18], a range counting query can be performed in $O(\log^2 n)$ time and the weight of a point can also be updated in $O(\log^2 n)$ time after $O(n\log n)$ preprocessing. Furthermore, using a static range tree, a range reporting query can be answered in $O(\log n + s)$ time, where s is the output size, and a range emptiness query takes $O(\log n)$ time. Hence, $\tau(n) = O(\log^2 n)$ for the algorithm in Section 3. Plugging these bounds of $\tau(n)$, $g(1/\varepsilon)$ and $\varphi(n)$ in Theorem 3.2, a hitting set of Σ, where $\mathcal{R}$ is a set of rectangles in $\mathbb{R}^2$, can be computed in $O(N\log^3 N\log\kappa)$ expected time.

For the second algorithm, we do not need dynamic range counting, but we need a range-reporting data structure for $\Sigma^\perp$, i.e., report all rectangles of $\mathcal{R}$ that contain a query point. This can also be done in $O(\log m + s)$ time, where s is the output size, after $O(m\log m)$ preprocessing. Hence $\tau(n) = O(\log n)$ for the second algorithm.

Finally, we also need an algorithm 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 in $O(N\log N)$ time by a simple sweep-line algorithm — we sweep a vertical line $\ell(x)$ from $x = -\infty$ to $x = +\infty$ and maintain a segment tree on the set $\mathcal{I}(x) = \{\ell(x) \cap R \mid R \in \mathcal{R}\}$ of intervals. When ℓ encounter a point $p \in \mathsf{X}$, we check in $O(\log m)$ time whether p lies in more than m/k intervals of $\mathcal{I}$. If so, we add p to N_1 and delete all rectangles of $\mathcal{R}$ that contain p. Obviously at most k points are reported, and the time spent is $O(N\log N)$.

Putting everything together $\tau(N) = O(\log N)$, $\varphi(N) = O(N\log N)$, and $g(1/\varepsilon) = \log\log(1/\varepsilon)$, so by Theorem 4.6, the expected time of the algorithm is $O(N\log^3 N)$. For $d = 3$, the sweep line and ε-net take $O(N\log^2 N)$ time and $\tau(N) = O(\log^2 N)$. We omit the details from this abstract. Hence, the expected running time of the second algorithm for $d = 3$ is $O(N\log^4 N)$. We thus 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 $O(N\log^{d+1} N)$ expected time, with high probability.*

For the set cover, we compute an ε-net of $\Sigma^\perp$ simply by choosing a random sample of size $O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$. By maintaining a range tree, the set $\mathsf{N}_1 \subseteq \mathcal{R}$ for the second algorithm can be computed in $O(N\log^d N)$ time. 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 $O(N\log^{d+1} N)$ expected time, with high probability.*

Halfspaces in 3D. Let X be a set of n points in $\mathbb{R}^3$ and $\mathcal{R}$ a set of m halfspaces in $\mathbb{R}^3$. An ε-net of $\Sigma = (\mathsf{X}, \mathcal{R})$ of size $O(1/\varepsilon)$ can be computed in $O(n\log n)$ expected time, and by duality transform, an ε-net of $\Sigma^\perp$ can be computed in $O(m\log m)$ expected time. Using the data structure by Afshani and Chan [1], a halfspace range-reporting query can be answered in $O(\log n + s)$ time, after $O(n\log n)$ preprocessing. A range-emptiness query takes $O(\log n)$ time. Hence $\tau(n) = O(\log n)$ for the second algorithm.

Finally, computing N_1 is more involved in this case. We basically use the approach described in Agarwal *et al.* [3, Section 3]. The algorithms works in $\lceil\log\kappa\rceil$ 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 $\mathcal{R}_i$. Since the maximum depth of R_i is $m/2^i$, $|\Xi_i| = O(2^i)$. 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 $\mathcal{R}_{i+1} = \mathcal{R}_i \setminus \bar{\mathcal{R}}_i$. Closely following the algorithm described in [3], Ξ_i, P_i and $\bar{\mathcal{R}}_i$ can be computed in $O(|\mathcal{R}_i|\log r_i) = O(|\mathcal{R}_i| i)$ 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 $i \leq \log\kappa$ is $O((m + n)\log^2\kappa)$, and $|\mathsf{N}_1| = O(\kappa)$. Hence $\varphi(N) = O(N\log^2 N)$. Putting everything together, 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 $O(N\log^3 N)$ expected time, with high probability.*

By a standard lifting transform, 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 $O(N\log^3 N)$ expected time, with high probability.*

Fat objects. 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. The algorithm described in [5] can be adapted to compute an ε-net of Σ of size $O((1/\varepsilon)\log\log(1/\varepsilon))$ in $O(n\log^3 n)$ expected time, and their another algorithm can construct an ε-net of $\Sigma^\perp$ of size $O((1/\varepsilon)\log(\log^* 1/\varepsilon))$ (using the improved bounds on the union of fat triangles [4]) in $O(m\log m)$ expected time. A range reporting query for both Σ and $\Sigma^\perp$ can be answered in $O(\log^3 N + s)$ time, where s is the output size. The set N_1 for both Σ and $\Sigma^\perp$ can be computed in $O(N\log^3 N)$ time. Omitting further details, including the computation of N_1, we conclude the following.

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

The range-searching data structure for $\Sigma^{\perp}$ works even if $\mathcal{R}$ is a set of locally fat objects of constant description complexity. The size of an ε-net in this case is $O((1/\varepsilon)\log^*(1/\varepsilon))$ [5, 4]. Hence, we obtain the following.

THEOREM 5.6. *Let* X *be a set of points in* $\mathbb{R}^2$ *and* $\mathcal{R}$ *a set of locally fat objects of constant description complexity in* $\mathbb{R}^2$*, with* $|\mathsf{X}| + |\mathcal{R}| = N$*. Then an* $O(\log^* \chi)$*-approximate set cover of* $(\mathsf{X}, \mathcal{R})$ *can be constructed in* $O(N \log^4 N)$ *expected time, with high probability.*

The algorithm for locally fat objects can be applied to obtain an $O(1)$-approximate set cover for the case when $\mathcal{R}$ a set of convex pseudo-disks in $\mathbb{R}^2$, each of constant description complexity.

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 $\delta/2 < k \le \delta$, 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 $\mathsf{E}[\max I(x, \tilde{\Omega})] \le 3/k$. Let y_i be the variable corresponding to the range $R_i \in \mathcal{R}$. Then by setting $y_i = \frac{k}{3}\Pr(R_i \sim \tilde{\Omega})$, we obtain a fractional solution for the independent-set problem with objective value $\sum y_i = k/3$. Next, we convert the fractional solution into an integral solution using the approach of Chan and Har-Peled [11]: 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 [11] 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 it takes $O(N \log^3 N)$ time. Finally, using a dynamic disk-emptiness data structure, the rounding step can be performed in $O(N \log^2 N)$ time [2]. Omitting the details, 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* $O(N \log^3 N)$ *expected time, with high probability.*

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 also extends 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 multi-cover [12] of range spaces?

(ii) 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 [10].

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, *Contemp. Math.*, 223 (1999), 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] B. Aronov, M. de Berg, E. Ezra, and M. Sharir, Improved bounds for the union of locally fat objects in the plane, *SIAM J. Comput.*, (2013).

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

[6] 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.

[7] 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.

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

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

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

[11] 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.

[12] 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.

[13] 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.

[14] K. Clarkson, Algorithms for polytope covering and approximation, in: *Algorithms and Data Structures* (F. Dehne, J.-R. Sack, N. Santoro, and S. Whitesides, eds.), *Lecture Notes in Computer Science*, Vol. 709, Springer, 1993, pp. 246–252.

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

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

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

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

[19] G. Even, D. Rawitz, and S. Shahar, Hitting sets when the vc-dimension is small, *Inf. Process. Lett.*, 95 (2005), 358–362.

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

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

[22] 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.

[23] 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.

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

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1990.

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

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

[28] 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.

[29] 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.

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

[31] J. Matoušek, R. Seidel, and E. Welzl, How to net a lot with little: Small ε-nets for disks and halfspaces, *Proc. 6th Annual Symp. Comput. Geom.*, 1990, pp. 16–22.

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

[33] N. H. Mustafa and S. Ray, PTAS for geometric hitting set problems via local search, *Proc. 25th Annual Symp. Comput. Geom.*, 2009, pp. 17–22.

[34] J. Pach and G. Tardos, Tight lower bounds for the size of epsilon-nets, *Proc. 27th Annual Symp. Comput. Geom.*, 2011, pp. 458–463.

[35] 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.

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

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

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

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