Hitting sets when the VC-dimension is small

Guy Even* Dror Rawitz* Shimon (Moni) Shahar*

September 28, 2004

Abstract

We present an approximation algorithm for the hitting set problem when the VC-dimension of the set system is small. Our algorithm builds on Pach & Agarwal [7], and we show how it can be parallelized and extended to the minimum cost hitting set problem. The running time of the proposed algorithm is comparable with that of Brönnimann & Goodrich [2], and the approximation ratio is smaller by a constant factor.

1 Introduction

1.1 Definitions

Definition 1 *A* set system *is a pair $\mathcal{F} = (X, \mathcal{R})$, where X is a set of* elements *(or* points*) and $\mathcal{R}$ is a collection of subsets of X.*

Definition 2 *A* hitting set *of a set system $\mathcal{F} = (X, \mathcal{R})$ is a subset $H \subseteq X$ that stabs every subset in $\mathcal{R}$ (i.e., $\forall S \in \mathcal{R} \ : \ S \cap H \neq \emptyset$).*

In computational geometry set systems are often called *range spaces* and hitting sets are often called *transversals* [7].

In the hitting set problem, the goal is to find a hitting set of minimum cardinality. The fractional hitting set problem is a fractional relaxation of the hitting set problem and is defined by

*School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel. {guy, rawitz, moni}@eng.tau.ac.il

the following linear program:

$$\begin{array}{lll} \min & \sum_{u\in X} x(u) & \\ \text{s.t.} & \sum_{u\in S} x(u) \geq 1 & \forall S \in \mathcal{R} \\ & x(u) \geq 0 & \forall u \in X \end{array} \tag{LP1}$$

We denote the size of a minimum size hitting set by τ, and the optimum value of LP1 by τ^*.

Definition 3 ([9]) *Consider a set system $\mathcal{F} = (X, \mathcal{R})$. A subset $A \subseteq X$ is* shattered *by $\mathcal{F}$ if, for every subset $B \subseteq A$, there exists a set $S \in R$ such that $B = A \cap S$.*

The VC-dimension *of $\mathcal{F}$ is the size of the largest set $A \subseteq X$ that is shattered by $\mathcal{F}$. If, for every k, there exists a subset A that is shattered by $\mathcal{F}$ and $|A| \geq k$, then the VC-dimension of $\mathcal{F}$ is said to be infinite.*

We denote the VC-dimension of the set system at hand by d.

Definition 4 ([3]) *Consider a set system $\mathcal{F} = (X, \mathcal{R})$. A set $H \subseteq X$ is an ε-net of $\mathcal{F}$ if $S \cap H \neq \emptyset$ for every subset $S \in \mathcal{R}$ for which $|S| \geq \varepsilon \cdot |X|$.*

Note that an ε-net is a hitting set of the set system $(X, \mathcal{R}_\varepsilon)$, where $\mathcal{R}_\varepsilon = \{S \in \mathcal{R} \mid |S| \geq \varepsilon \cdot |X|\}$.

Given a weight function $w : X \to \mathbb{R}^{\geq 0}$, we define the weight $w(A)$ of a subset A to be the sum of the weights of the points in A. The weighted version of an ε-net is defined as follows.

Definition 5 *Consider a set system $\mathcal{F} = (X, \mathcal{R})$ and a weight function $w : X \to \mathbb{R}^{\geq 0}$. A set $H \subseteq X$ is an ε-net of $\mathcal{F}$ with respect to w if $S \cap H \neq \emptyset$ for every subset $S \in \mathcal{R}$ for which $w(S) \geq \varepsilon \cdot w(X)$.*

Since every weight function induces a probability measure $\mu : X \to [0, 1]$ by $\mu(u) \triangleq w(u)/w(X)$, probability measures over X are often used instead of weights. An algorithm that, given a set system (with weights) and $\varepsilon > 0$, outputs an ε-net is called a *net-finder*.

1.2 Previous Results

The following theorem is an improvement over Haussler & Welzl [3] that followed an earlier proof of [9] for a theorem on ε-approximations. This theorem states that random samples are likely to be net-finders.

Theorem 6 ([1, 4]) *Let $\mathcal{F} = (X, R)$ denote a set system with weights $w(u)$. For every $\varepsilon \leq 1/2$, a random sample of X according to the probability distribution $w(u)/w(X)$ is likely to be an ε-net with respect to $w(u)$ if the sample contains $O(\frac{d}{\varepsilon}\log\frac{1}{\varepsilon})$ elements.*

Corollary 7 ([7, 6]) $\tau = O(\tau^* d \log \tau^*)$.

The algorithmic nature of Coro. 7 is not widely known and many refer to the algorithm of Brönnimann & Goodrich [2] instead. The algorithm in [2] finds a hitting set whose size is bounded by $O(d\tau \log \tau)$. In special cases, where there exist small ε-nets of size $O(\frac{d}{\varepsilon})$, $O(d)$-approximation algorithms are obtained.

The algorithm of Brönnimann & Goodrich [2] uses two black boxes (oracles): a net-finder and a verifier. The verifier is a separation procedure that checks whether a given subset $H \subseteq X$ is a hitting set. If it is not, then the verifier outputs a witness $S \in \mathcal{R}$ that is not stabbed by H.

We now briefly describe the algorithm of Brönnimann & Goodrich. The algorithm performs a doubling search until the value of τ is reached. For each guess τ' of τ, the algorithm tries to compute a weight function $w(u)$ with the following property: $w(S) \geq \varepsilon \cdot w(X)$, for every $S \in \mathcal{R}$ (where $\varepsilon = 1/\tau'$). This is done as follows. One starts with unit weights and iterates as follows: (i) Invoke the net-finder that returns an ε-net H. (ii) Verify whether the ε-net is a hitting set. (iii) If H is not a hitting set, the weights of the elements in the witness set are doubled. A bound on the number of iterations needed to compute such a weight function is given if $\tau' \geq \tau$. Hence, if the number of iterations exceeds this bound, then we know that the guess τ' is too small, and the guess τ' it doubled. The doubling search stops if the guess is not too small, in which case a hitting set is obtained, the size of which is guaranteed to be $O(d\tau \log \tau)$.

1.3 Our Results

We present an algorithm for computing hitting sets of set systems with small VC dimension. This algorithm is achieves the bound of Coro. 7. As in [2], a constant approximation ratio is obtained if the net-finder returns an ε-net of size $O(\frac{d}{\varepsilon})$.

The algorithm requires (i) approximately solving a linear program of an instance of the fractional hitting set problem, and (ii) invoking a net-finder exactly once. The running time of the algorithm is not bigger than the running time of the algorithm of Brönnimann & Goodrich. In fact, if the net-finder is slow, then our algorithm is faster since we invoke the net-finder only once. The approximation ratio of our algorithm is smaller by a factor of 4, which is of interest when the algorithm gives an $O(d)$-approximation (i.e., when the net-finder returns an ε-net of size $O(\frac{d}{\varepsilon})$).

Another advantage of our algorithm is that it can be parallelized if the net-finder can be parallelized. This is the case with random samples and with other constructions (e.g., range spaces of a finite set of points in the Euclidean plane and sets induced by half-spaces).

Finally, we deal with hitting sets of set systems where each element has a cost associated with it. The goal here is to find (or approximate) a hitting set with minimum cost. We extend our result to approximating minimum cost hitting sets if the net-finder is a random sample of elements.

We speculate that many used the algorithm of Brönnimann & Goodrich instead of Corollary 7 since the algorithm of Brönnimann & Goodrich is easy to implement and it avoids linear programming. These advantages are not lost in our algorithm. In fact, the fractional hitting set LP can be approximately solved by algorithms that are as easy to implement and analyze (e.g., [10, 11]).

2 The Algorithm

A normalized LP. The algorithm we propose for computing a hitting set directly finds a distribution μ for which every ε-net is a hitting set by solving the following linear program.

$$\begin{array}{lll} \max & \varepsilon & \\ \text{s.t.} & \mu(S) \geq \varepsilon & \forall S \in \mathcal{R} \\ & \sum_{u \in X} \mu(u) = 1 & \\ & \varepsilon, \mu(u) \geq 0 & \forall u \in X \end{array} \tag{LP2}$$

Note that LP2 is a linear program in which the polytope P of feasible weights μ is simply the set of probability distributions over X. The objective is to maximize the value $\min_{S \in \mathcal{R}} \mu(S)$.

Observation 8 *The linear programs LP1 and LP2 are equivalent.*

Proof: Consider the following substitution: $\varepsilon \triangleq 1/x(X)$ and $\mu(u) \triangleq x(u) \cdot \varepsilon$. Thus, $\tau^* = 1/\varepsilon^*$, where ε^* is the maximum value of LP2. □

Indeed, LP2 was previously used for solving LPs of covering problems, c.f. [8, 10]. This equivalence can be performed not only in linear time but also in parallel.

The algorithm. The algorithm consists of the following two steps: (i) Solve LP2 to obtain μ^* and ε^*. (ii) Invoke a net-finder for the set-system $\mathcal{F} = (X, \mathcal{R})$ with respect to the weights μ^* and the parameter ε^*. Let H denote the ε^*-net computed by the net-finder. Since $\mu(S) \geq \varepsilon^*$, for every $S \in \mathcal{R}$, it follows that H is a hitting set.

The size of H is bounded simply by the size of the ε^*-net computed by the net-finder. A net-finder based on Theorem 6 returns a hitting set that meets the bound in Coro. 7. Moreover, if the net-finder computes an ε^*-net of size $O(\frac{d}{\varepsilon^*})$, then an $O(d)$-approximation algorithm is obtained.

Compare this with Brönnimann & Goodrich [2] in which the net-finder is given a parameter ε that can be as small as $\frac{1}{4\tau} \leq \frac{\varepsilon^*}{4}$. Hence, our approximation ratio is smaller by a factor of 4. (To be precise, our approximation ratio is smaller by an additional factor of $\tau/\tau^* \cdot (1+\delta)^{-1}$, where $(1+\delta)$ denotes the approximation ratio of the LP solver.) This reduced approximation ratio is mainly interesting in the case of $O(1)$-approximation algorithms.

Implementation and running time. Instead of solving LP2 by general linear programming techniques, it is possible to apply polynomial time approximation schemes (PTASs) that were developed for solving covering problems [8, 5, 10]. In Young's algorithm [10], for example, LP2 can be solved within a factor of $(1+\delta)$ by performing $\frac{\tau^* \ln(m)}{\delta^2}$ iterations. In each iteration, every subset $S \in \mathcal{R}$ is given a cost $y(S)$, and the most time consuming computations are: (i) find an element $u \in X$ that maximizes $\sum_{\{S \mid u \in S\}} y(S)$, and (ii) update the costs of the subsets stabbed by u. Each iteration can be easily computed in $O(mn)$ time.

We now compare the running time required for solving LP2 and one invocation of the net-finder with the running time of the algorithm of [2]. In [2], $O(\tau \log \frac{n}{\tau})$ invocations of the net-finder and the verifier are required. The running time of the verifier is comparable to the running time of one iteration of Young's algorithm. Hence, our suggestion for computing a hitting set is not slower than that of [2]. In fact, if the net-finder is slow (e.g., not based on random sampling), then the fact that we require only one invocation of the net-finder may become significant.

Parallelization. The main open question that is stated by Brönnimann & Goodrich in [2] is whether the algorithm can be parallelized. Recall that in their algorithm there are $O(\tau \log \frac{n}{\tau})$ iterations; each iteration invokes the net-finder, the verifier, and updates the weights of points in the witness set (if the verifier finds one). Our algorithm (approximately) solves LP2 and then invokes the net-finder exactly once.

The linear program LP2 can be approximately solved in parallel using the algorithms of [5, 11]. The running time of Young's algorithm [11] is polylogarithmic in $|X|$ and $|\mathcal{R}|$.

Since parallel algorithms for solving LP2 are known, we are left with the task of parallelizing the net-finder. As in the algorithm of Brönnimann & Goodrich, the net-finder is a black-box, so it is impossible to deal with parallelization of general net-finders. Hence, we address specific cases. The simplest case is the case of random selections. Namely, if the net-finder simply randomly selects elements (as suggested in Theorem 6), then the random selection can be obviously done in parallel.

We provide another example in which the net-finder can be parallelized. Let X denote a finite set of n points in the Euclidean plane. Let $\mathcal{R}$ denote the collection of subsets of X obtained by intersecting X with open half-planes. Komlós et al. [4] present a construction for ε-nets of size $O(\frac{1}{\varepsilon})$ for this range space. The construction (for $\varepsilon < 2/3$) picks a minimal subset N of points lying on the convex of X such that N is an ε-net. They prove that such an ε-net contains at most $\lceil \frac{2}{\varepsilon} \rceil - 1$ points. This construction can be easily parallelized.

3 Minimum Cost Hitting Set

In the minimum cost hitting set problem (min-cost HS), each element $u \in X$ is given a cost $c(u)$. The cost of a subset $A \subseteq X$ is the sum of the costs of the elements in A. We denote the cost of A by $c(A)$. The goal is to find a hitting set of minimum cost. We assume that $c(u) \geq 1$, for every $u \in X$. (This assumption is important because the approximation ratio is $O(d \cdot \log \tau')$, where τ' is the value of a fractional min-cost HS.) The following claim is proved in this section.

Claim 9 *There exists a randomized polynomial time algorithm that computes a hitting set with probability greater than* $1/2$. *The expected cost of the hitting set is* $O(d\tau' \log \tau')$.

Proof: The fractional relaxation, LP1', of the min-cost HS is identical to LP1, except that the objective function is $\min \sum_{u\in X} c(u)x(u)$.

$$\begin{array}{lll} \min & \sum_{u\in X} c(u)\cdot x(u) & \\ \text{s.t.} & \sum_{u\in S} x(u) \geq 1 & \forall S \in \mathcal{R} \\ & x(u) \geq 0 & \forall u \in X \end{array} \tag{LP1'}$$

Let τ' denote the optimal value of a solution of LP1'. Obviously, τ' is a lower bound on the cost of a minimum cost HS. The linear program LP2' that is analogous to LP2 is obtained by replacing the constraint $\sum_{u\in X} \mu(u) = 1$ by the constraint $\sum_{u\in X} c(u)\mu(u) = 1$.

$$\begin{array}{lll} \max & \varepsilon & \\ \text{s.t.} & \mu(S) \geq \varepsilon & \forall S \in \mathcal{R} \\ & \sum_{u\in X} c(u)\cdot \mu(u) = 1 & \\ & \varepsilon, \mu(u) \geq 0 & \forall u \in X \end{array} \tag{LP2'}$$

Observation 10 *Let μ', ε' denote an optimal solution to LP2'. Then, $\tau' = 1/\varepsilon'$.*

The algorithm first (approximately) solves LP2'. Note that $\mu'(X) \leq 1$ since $\sum_{u\in X} c(u)\cdot\mu(u) = 1$ and $c(u) \geq 1$. We add a dummy element z with zero cost and $\mu'(z) = 1 - \mu'(X)$. The algorithm now finds an ε'-net with respect to the weights μ' by randomly selecting elements according to Theorem 6.

Observation 11 *Every ε'-net with respect to the weights μ' is a hitting set.*

Let H denote a random sample of points from $X \cup \{z\}$ according to μ'. By Theorem 6, if $|H| = O(\varepsilon' d \log \frac{1}{\varepsilon'})$, then H is likely to be a hitting set.

Observation 12 *The expected cost of H is $|H|$.*

Proof: Since $\sum_{u\in X\cup\{z\}} c(u)\mu'(u) = 1$, it follows that if we select an element $u \in X \cup \{z\}$ with probability $\mu'(u)$, then the expected cost of the selected element is 1. □

We conclude that the randomized algorithm computes a hitting set whose expected cost is as required, and the claim follows. □

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[2] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14:463–479, 1995.

[3] D. Haussler and E. Welzl. ϵ-Nets and Simplex Range Queries. *Discrete Computational Geometry*, 2:127–151, 1987.

[4] J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for epsilon-nets. *Discrete and Computational Geometry*, 7:163–173, 1992.

[5] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *ACM Symposium on Theory of Computing*, pages 448–457, 1993.

[6] J. Matoušek. *Lectures on Discrete Geometry.* Springer-Verlag New York, Inc., 2002.

[7] J. Pach and P. K. Agarwal. *Combinatorial Geometry.* J. Wiley, New York, 1995.

[8] S. A. Plotkin, D. B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20:257–301, 1995.

[9] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

[10] N. E. Young. Randomized rounding without solving the linear program. In *6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 170–178, 1995.

[11] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.