Weighted search in the plane

Richa Agarwala* and David Fernández-Baca†

Abstract

We present a simple two-dimensional weighted version of Megiddo's multidimensional search technique. This speeds up algorithms for certain convex optimization problems in the plane.

Keywords: Algorithms, combinatorial problems, convex optimization.

1 Introduction

We present a two-dimensional weighted version of the multidimensional search technique of Megiddo [4, 7, 9]. The result has applications to a class of two-dimensional convex optimization problems [2, 3, 5, 6, 11]. We need to present some background before stating our result precisely.

Let $\Lambda \subseteq \mathbf{R}^d$ be convex and let $h : \mathbf{R}^d \to \mathbf{R}$ be an affine function. Define $\mathrm{sign}_\Lambda(h)$ as

$$\mathrm{sign}_\Lambda(h) = \begin{cases} 0 & \text{if } h(\lambda) = 0 \text{ for some } \lambda \in \Lambda \\ +1 & \text{if } h(\lambda) > 0 \text{ for all } \lambda \in \Lambda \\ -1 & \text{if } h(\lambda) < 0 \text{ for all } \lambda \in \Lambda \end{cases}$$

We will write sign for sign_Λ when no confusion can arise. A function h is *resolved* if $\mathrm{sign}_\Lambda(h)$ has been computed.

Suppose we have a set $\mathcal{H}$ of n d-dimensional affine functions and an oracle $\mathcal{B}$ that can resolve any h. Let S be a set on which a weight function $w : S \to \mathbf{R}^+$ has been defined. For $S' \subseteq S$ we write $w(S')$ to denote $\sum_{s \in S'} w(s)$. The following result is known [2].

Theorem 1 *For each $d \geq 0$, there exist constants $\beta(d)$ and $\alpha(d)$, $\alpha \leq 1/2$ and an algorithm* WEIGHTED-SEARCH *with the following property. Given a set $\mathcal{H}$ of affine functions and a weight function $w : \mathcal{H} \to \mathbf{R}^+$,* WEIGHTED-SEARCH *resolves every $h \in \mathcal{H}' \subseteq \mathcal{H}$, where $w(\mathcal{H}') \geq \alpha(d) \cdot w(\mathcal{H})$ by making at most $\beta(d)$ calls to $\mathcal{B}$. Furthermore, the work done by* WEIGHTED-SEARCH *in addition to the oracle calls is $O(n)$.*

*DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), Rutgers University, Piscataway, NJ 08855 (*agarwala@dimacs.rutgers.edu*). Part of this work was done while this author was at Iowa State University and was supported by the National Science Foundation under grant CCR-9211262.

†Department of Computer Science, Iowa State University, Ames, IA 50011 (*fernande@cs.iastate.edu*). Supported in part by the National Science Foundation under grant CCR-9211262.

Agarwala and Fernández-Baca [2] gave two algorithms for this problem, one of which achieved $\beta(d) = 2^d - 1$ and $\alpha(d) = 12/24^{2^{d-1}}$, the other achieved $\beta(d) = 2(60^{d-1})$ and $\alpha(d) = 1/12$. Subsequently, Agarwal, Sharir, and Toledo [1] gave an algorithm that achieved $\beta(d) = O(d^3 \log d)$ and $\alpha(d) = 1/2$. All these algorithms are quite complex and require many oracle calls, even for small d. Since oracle calls tend to be expensive, it is of interest to develop simple algorithms which require fewer oracle calls. Here we give a comparatively simple weighted search algorithm for the plane which achieves $\beta(2) = 3$ and $\alpha(2) = 1/4$ with $O(n)$ additional work.

2 The algorithm

Suppose $\mathcal{H} = \{h_1, \ldots, h_n\}$, where $h_i(x, y) = a_i x + b_i y + c_i$. If $a_i, b_i = 0$, $\mathrm{sign}(h_i)$ is simply the sign of c_i, and no oracle calls are needed. Thus, the presence of such h_i's can only help. We shall henceforth assume that $a_i \neq 0$ or $b_i \neq 0$ for $i = 1, \cdots, n$. In this case, each affine function h_i corresponds to a line $H_i \in \mathbf{R}^2$ where $H_i = \{(x, y) : h_i(x, y) = 0\}$. Computing $\mathrm{sign}(h_i)$ is thus equivalent to determining whether H_i intersects Λ and, if not, which side of H_i contains Λ. We shall find it convenient to deal interchangeably with lines and affine functions and to extend the weight function w to these lines by making $w(H_i) = w(h_i)$.

Suppose $b_i = 0$ for some $i \in \{1, \cdots, n\}$. Pick a β such that $\beta \neq (-b_i/a_i)$ for $i = 1, \cdots, n$. By using a change of variables $x = x' + \beta \cdot y$ and $b_i = b_i' - \beta \cdot a_i$, we get a set of lines in which $b_i' \neq 0$ for $i = 1, \cdots, n$. This change of variables simplifies the exposition and needs to be reversed before making an oracle call. For convenience, we now drop the primes on x' and b_i'. Henceforth, $b_i \neq 0$ for $i = 1, \cdots, n$.

Since for every scalar $l \neq 0$, $\mathrm{sign}(h(x, y)) = \mathrm{sign}(l) \cdot \mathrm{sign}(h(x, y)/l)$, we can rewrite the equations of these lines so that $a_i \geq 0$. Let the *slope* α_i of H_i be the same as that of $a_i x + b_i y + c_i = 0$ with respect to $y = 0$; i.e., let $\alpha_i = (-a_i/b_i)$. Since $b_i \neq 0$ for $i = 1, \cdots, n$, the slopes are well defined. Let α^* be the weighted median of the set $\{\alpha_i\}$ where the weight of α_i is $w(H_i)$. Now we make the slopes of roughly weighted half of the lines nonnegative and weighted half nonpositive by using the change of variables $y = y' + \alpha^* \cdot x$ and $a_i = a_i' - \alpha^* \cdot b_i$. As before, this change of variables is only done to simplify the exposition and needs to be reversed before making an oracle call. We drop the primes on y' and a_i'. Recalculate the slopes of the lines after making this change in variables. All lines that originally had a slope of α^* will have 0 slope. Let $\mathcal{H}_0 = \{H_i : \alpha_i = 0\}$, $\mathcal{H}_- = \{H_i : \alpha_i < 0\}$, and $\mathcal{H}_+ = \{H_i : \alpha_i > 0\}$.

Since 0 is our new weighted median slope, $w(\mathcal{H}_-) \leq W/2$ and $w(\mathcal{H}_-) + w(\mathcal{H}_0) \geq W/2$. Therefore, $w(\mathcal{H}_-) \geq W/2 - w(\mathcal{H}_0)$. Similarly, $W/2 \geq w(\mathcal{H}_+) \geq W/2 - w(\mathcal{H}_0)$. Assume without loss of generality that $\mathcal{H}_+ = \{h_1, \cdots, h_k\}$, $\mathcal{H}_- = \{h_{k+1}, \cdots, h_m\}$, and $\mathcal{H}_0 = \{h_{m+1}, \cdots, h_n\}$. Let $W_P(c)$ $[W_N(c)]$ be the weighted median of $p_1, \cdots, p_k$ $[p_{k+1}, \cdots, p_m]$, where p_i is the point of intersection of lines H_i and $y = c$ (i.e., $p_i = (-(c \cdot b_i + c_i)/a_i, c)$) and $w(p_i) = w(H_i)$. Let W_P $[W_N]$ be the curve that we get if we plot $W_P(c)$ $[W_N(c)]$ from $c = -\infty$ to $c = \infty$ (See Figure 1).

Observation 1 *W_P $[W_N]$ is a monotonically increasing [decreasing] function.*

Let $p^* = (a^*, b^*)$ be the point where W_P intersects W_N. Let $H_P^* \in \mathcal{H}_+$ and $H_N^* \in \mathcal{H}_-$ be the lines whose intersection point is p^*. It is easy to see that if we resolve the lines

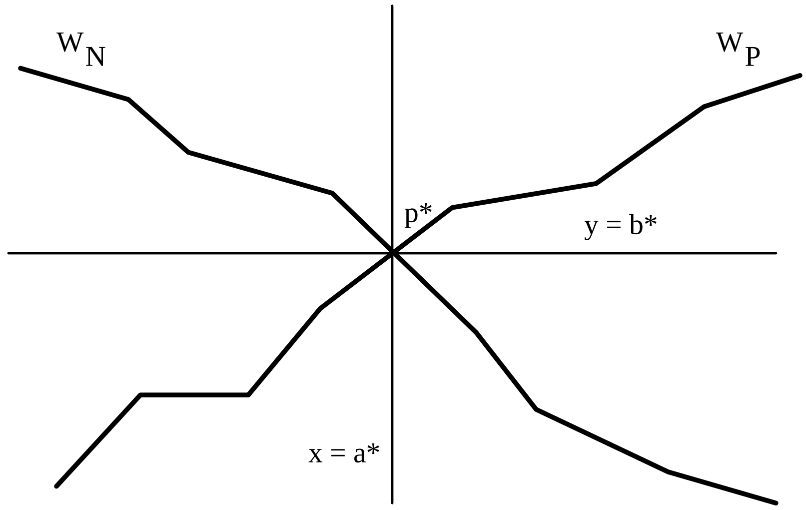

Figure 1: W_P and W_N.

$y = b^*$ and $x = a^*$, which requires two oracle calls, then either we resolve $\mathcal{H}' \subseteq \mathcal{H}_+$ such that $w(\mathcal{H}') \geq w(\mathcal{H}_+)/2$ or we resolve $\mathcal{H}' \subseteq \mathcal{H}_-$ such that $w(\mathcal{H}') \geq w(\mathcal{H}_-)/2$. (For example, if Λ lies in the quadrant described by $y < b^*$ and $x > a^*$, then we resolve $\{h_i \in \mathcal{H}_+ : -(b^* \cdot b_i + c_i)/a_i \leq a^*\}$; see [9] for details.) In either case, we resolve $\mathcal{H}' \subseteq \mathcal{H}$ such that $w(\mathcal{H}') \geq (1/2)(W/2 - w(\mathcal{H}_0))$. Furthermore, p^* can be found in $O(m)$ time by adapting a technique due to Megiddo [10] as described in the next section.

Now resolve line $y = c^*$, which requires one oracle call, where c^* is the weighted median of $\{p_i : p_i = -c_i/b_i, h_i \in \mathcal{H}_0\}$ and $w(p_i) = w(h_i)$. This resolves lines with total weight at least $w(\mathcal{H}_0)/2$. Hence, when the search is restricted to Euclidean plane, we have $\beta(2) = 3$ and the weight resolved is at least $(1/2)(W/2 - w(\mathcal{H}_0)) + w(\mathcal{H}_0)/2 = W/4$ giving us $\alpha(2) = 1/4$ with $O(n)$ additional work.

3 Finding p^*

We rely on an idea introduced by Megiddo [10] who gave an algorithm for the case where W_N and W_P correspond to unweighted medians. Like Megiddo, we use a line query algorithm which finds the position of the corresponding p^* with respect to any straight line in $O(m)$ time. In Section 3.1, we adapt Megiddo's line query algorithm to handle the weighted case. We solve our problem by invoking the line query algorithm repeatedly; each step either finds p^* or it discards at least $1/8$ of the lines as candidates for H_P^* and H_N^*. Once we find H_P^* and H_N^*, p^* is then just their intersection point. A summary of the steps required for finding H_P^* and H_N^* is presented in Section 3.2.

3.1 The line query

Suppose L is a vertical line given by $x = c$. The analysis for the case where L is horizontal is analogous. L intersects W_P and W_N exactly once. Let $P = (c, b_P)$ be the point where W_P intersects L and $N = (c, b_N)$ be the point where W_N intersects L. N [P] is the weighted median of the y values of the intersection points of lines in $\mathcal{H}_-$ [$\mathcal{H}_+$] with L. It is easy to see that if $b_N < b_P$ [$b_N > b_P$] then $a^* < c$ [$a^* > c$]. Furthermore, if $b_N = b_P$ we have $p^* = (c, b_N)$.

Now consider the case where L has positive slope; the analysis for the case where L has negative slope is analogous. Line L will intersect with W_N exactly once, at a point $N = (a, b)$ that is the weighted median of the x values of the intersection points of lines in $\mathcal{H}_-$ with L. Thus N can be found in $O(m - k)$ time. Let P be the weighted median of the y values of the intersection points of lines in $\mathcal{H}_+$ and the line $x = a$. This takes $O(k)$ time. As before, the relative position of N and P along with the fact that N lies on W_N and P lies on W_P is sufficient to determine the relative position of L and p^*.

3.2 Finding H_P^* and H_N^*

Let R be the set of candidate lines for H_P^* and H_N^* at the beginning of the current iteration. At the start of the algorithm $R = \mathcal{H}_+ \cup \mathcal{H}_-$. Each iteration either finds p^* or discards $1/8$ of the candidate lines as follows:

Step 1. Find the median slope s of the lines in R.

Step 2. Partition R into sets G and S such that the lines in G [S] have slopes greater [smaller] than s.

Step 3. Form disjoint pairs of lines so that each pair has one line from G and another from S.

Step 4. Compute the x values of the intersection points of the paired lines. Let c be the median of these values.

Step 5. Do a line query on line $x = c$. It either finds p^* in which case we are done, or it determines that $a^* < c$, or $a^* > c$. Suppose $a^* < c$. The case for $a^* > c$ is analogous.

Step 5.1 Consider the pairs of lines whose intersection points have x values greater than or equal to c. There are at least $|R|/4$ lines in these pairs.

Step 5.2 Let L be the line with slope s and y intercept equal to the median of the set $\{y - sx\}$ where (x, y) runs over the intersection points still under consideration. Hence L divides the intersection points under consideration into two equal sets.

Step 5.3 Do a line query on L. Lines L and $x = c$ identify a quadrant Q in which p^* lies and the quadrant opposite to Q has $|R|/8$ intersection points. With respect to each intersection point, there is a line which does not cross Q and thus can not be H_P^* or H_N^*. Hence these lines can be discarded and we have a similar problem with $7|R|/8$ lines.

Acknowledgement

We thank Naoki Katoh for drawing our attention to [10].

References

[1] P.K. Agarwal, M. Sharir, and S. Toledo. An efficient multidimensional searching technique and its applications, Technical Report CS-1993-20, Duke University, July 1993.

[2] R. Agarwala and D. Fernández-Baca. Weighted multidimensional search and its application to convex optimization, DIMACS Technical Report 92-51, November, 1992. To appear in *SIAM J. Comput.*.

[3] Y. P. Aneja and S. N. Kabadi. Polynomial algorithms for lagrangean relaxations in combinatorial problems, Manuscript.

[4] K.L. Clarkson. Linear programming in $O(n \times 3^{d^2})$ time, *Information Processing Letters* 22 (1986), 21–24.

[5] E. Cohen. Combinatorial algorithms for optimization problems, Report No. STAN-CS-91-1366, Department of Computer Science, Stanford University, Stanford, CA 94305.

[6] E. Cohen and N. Megiddo. Maximizing concave functions in fixed dimension, in : P. M. Pardalos, editor, *Complexity in Numerical Optimization* (1993), 74–87.

[7] M. E. Dyer. On a multidimensional search technique and its application to the Euclidean one–centre problem, *SIAM J. Comput.* 15 (3) (1986), 725–738.

[8] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems, *Management Science* 27 (1) (1981), 1–18.

[9] N. Megiddo. Linear programming in linear time when the dimension is fixed, *J. Assoc. Comput. Mach.*, 31 (1984), 114–127.

[10] N. Megiddo. Partitioning with two lines in the plane, *J. Algorithms*, 6 (1985), 430-433.

[11] C.H. Norton, S.A. Plotkin, and É. Tardos. Using separation algorithms in fixed dimension, *J. Algorithms* 13 (1992), 79–98.