Optimal Slope Selection via Expanders*

Matthew J. Katz† Micha Sharir‡

Abstract

Given n points in the plane and an integer k, the slope selection problem is to find the pair of points whose connecting line has the k-th smallest slope. (In dual setting, given n lines in the plane, we want to find the vertex of their arrangement with the k-th smallest x-coordinate.) Cole et al. [6] have given an $O(n \log n)$ solution (which is optimal), using the parametric searching technique of Megiddo. We obtain another optimal (deterministic) solution that does not depend on parametric searching and uses expander graphs instead. Our solution is somewhat simpler than that of [6] and has a more explicit geometric interpretation.

1 Introduction

In this paper we consider the slope selection problem, as defined in the abstract. For convenience, we prefer to study its dual version. We thus have a collection $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ of n lines in the plane, which we assume to be in general position, meaning that no line is vertical, no two lines are parallel, no three lines are concurrent, and no two distinct pairs of lines intersect at points that have the same x-coordinate. We are also given an integer parameter $1 \leq k \leq \binom{n}{2}$, and our goal is to find the vertex of the arrangement $\mathcal{A}(\mathcal{L})$ of $\mathcal{L}$ that has the k-th smallest x-coordinate.

This problem was studied by Cole et al. [6], who developed a fairly complicated $O(n \log n)$-time solution, which is optimal. Their algorithm is based on the parametric searching technique of Megiddo [14]. In the context of slope selection, this technique is based on an 'oracle' procedure that, given any vertical line $x = a$, can quickly compute the number of vertices of $\mathcal{A}(\mathcal{L})$ that lie to the left of the line. There is a simple way to perform this task, in $O(n \log n)$ time, by sorting the lines by their intercepts with the line $x = a$, and by counting inversions between the resulting permutation and the similar permutation of the lines at $x = -\infty$. The parametric searching approach then runs a (sequential simulation of a) 'generic' parallel sorting algorithm, which attempts to sort the lines along the line $x = a^*$, where a^* is the x-coordinate of the desired k-th leftmost vertex v_k, without knowing the value of a^*. At each parallel step we obtain a batch of comparisons between pairs of lines. We compute the x-coordinates of all the intersection points between these pairs, and run a binary search among them, using the oracle itself, to obtain a vertical slab between two successive such x-coordinates which is known to contain v_k; this allows us to determine the output to each of the comparisons involved. We can then proceed to the next parallel step. When the algorithm terminates, we are left with a slab that contains only the vertex v_k and then we can easily retrieve that vertex.

Naive implementation of this algorithm results in an $O(n \log^3 n)$ solution: Assume we use a parallel sorting algorithm of depth $O(\log n)$ and $O(n)$ processors. Each parallel step requires $O(\log n)$ oracle calls, each costing $O(n \log n)$, for a total cost of $O(n \log^3 n)$. A standard trick due to Cole [5] improves this to $O(n \log^2 n)$. Roughly speaking, at each parallel step we perform only a constant number of binary search steps. This of course does not resolve all comparisons of this step, but it does resolve a large fraction of them. With this delayed evaluation, each parallel step is now a mixture of many stages of the original parallel sort. Cole shows that, if we use a sorting *network* for the generic parallel sort, and use a weighted scheme in the binary searches, where the weight of an unresolved comparison increases exponentially with the number of steps it is left unresolved, the whole procedure does terminate after $O(\log n)$ new parallel steps. Since each of these steps uses only a constant number of oracle calls, the whole running time improves to $O(n \log^2 n)$.

To recap, even to obtain only an $O(n \log^2 n)$ solution, we have to use rather heavy machinery—the AKS sorting network [1] (the only known network with the above performance characteristics), combined with Cole's technique; this already results in a fairly complicated algorithm.

The final step of improvement is even more compli-

*Work on this paper has been supported by a grant from the Fund for Basic Research administered by the Israeli Academy of Sciences. Work by the second author has also been supported by NSF Grant CCR-91-22103, and by grants from the U.S.-Israeli Binational Science Foundation, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

†School of Mathematical Sciences, Tel Aviv University

‡School of Mathematical Sciences, Tel Aviv University, and Courant Institute of Mathematical Sciences, New York University

cated, because we can no longer afford to use the oracle too many times—it is too expensive. Instead, one needs to work with an approximate oracle, that counts the number of desired permutation inversions only approximately, with an error that keeps getting smaller and smaller as we get closer to the desired vertex v_k. Cole et al. give a fairly complicated technique for doing just that, thus obtaining their optimal solution.

A close inspection of the approximate counting scheme of [6] reveals that it is rather independent of the parametric search itself. More precisely, it maintains, along each vertical line λ bounding the current slab, a partition of the set of intersection points of the lines with λ into blocks of some fixed size q; the blocks are ordered along λ but the lines within a block may not be ordered. Using such a partition, Cole et al. manage to count, in linear time, the number of vertices of $\mathcal{A}(\mathcal{L})$ to the left of λ, with an additive error of $O(nq)$.

The scheme uses three main procedures: (i) Generate a similar partitioning along a given line lying within the slab (at which we want to make an oracle call); (ii) Refine the granularity of the partition (halve each block into two sub-blocks), so as to roughly halve the error in the inversions counting; (iii) Approximately count the number of vertices of $\mathcal{A}(\mathcal{L})$ within a slab bounded by two partitioned borders. Overall the procedure performs $O(\log n)$ refinements, and $O(\log n)$ partitionings and countings (because the technique of [6] generates only $O(\log n)$ oracle calls). Each partitioning, counting and refinement takes only linear time.

The important observation is that if we replace the parametric searching of [6] by any other kind of searching over x-coordinates, so that it generates only $O(\log n)$ oracle calls, we can replace the exact counting oracle by the above scheme, and obtain a procedure whose cost is only $O(n \log n)$. (We assume here that the residual cost of the search, excluding oracle calls, is only $O(n \log n)$; this is indeed the case in [6] and in our approach.)

Since the appearance of [6], several attempts have been made to obtain simpler solutions. Last year, Chazelle et al. [4] have proposed an alternative approach to parametric searching in geometric optimization. They have applied it only to the slope selection problem and obtained an $O(n \log^2 n)$ (deterministic) solution. Their technique is based on epsilon-nets and on recent related partitioning techniques (called *cuttings*). This solution is suboptimal, but it is conceptually simpler than [6]. Still, it requires the computation of ϵ-nets and of cuttings for arrangements of lines, which is by no means an easy task. By the observation just made, the running time of this algorithm can be improved to $O(n \log n)$, by replacing the oracle calls by calls to the approximating oracle reviewed above (this observation was missed in [4]).

Another alternative approach has recently been proposed by Matoušek [12] and by Dillencourt et al. [7]. This approach is based on randomization; it generates a small number of random vertices of the arrangement, and runs the usual binary search to locate a^* among these vertices. Then it generates a new set of random vertices, all within the restricted slab where v_k is now known to lie, runs a second binary search, and continues in this manner until v_k is found. The algorithms of [7, 12] have expected running time $O(n \log n)$, although they use a fairly large amount of random bits.

In this paper we propose another approach to the slope selection problem. Our approach is based on *expander graphs* that are constructed on certain subsets of the given lines. Since expanders can be constructed in an explicit deterministic manner (see e.g. [3] and below), our technique is deterministic. It is conceptually simpler than the other deterministic techniques, does not require parallelization, and has a very clear geometric interpretation. Its analysis relies on a simple and known property of (certain kinds of) expanders, which states, loosely, that for any pair of sufficiently large sets of vertices the expander contains sufficiently many edges between them. There are only a few applications of expanders to geometric problems, such as the recent application of Ajtai and Megiddo of expanders to parallel linear programming [2] (our solution exploits some ideas developed in that paper). We hope that our study will lead to further geometric applications of expanders. We remark that a different approach to parametric searching via expanders has recently been developed by the authors [9]. It applies to more general problems in geometric optimization, but for the slope selection problem it only yields an $O(n \log^3 n)$ algorithm.

The paper is organized as follows. In Section 2 we review expander graphs and their basic properties. In Section 3 we describe applications of expanders to arrangements of lines in the plane, obtaining a decomposition theorem that extends and somewhat improves similar results of [2] and may be interesting for its own sake. In Section 4 we present our slope selection algorithm, and conclude the paper in Section 5 with a brief discussion of our results and some open problems.

2 Expanders and Their Properties

Definition 2.1 *A graph $G = (V, E)$ is an (n, d, c) expander if it has n vertices, its degree is d, and for every set of vertices $W \subset V$ of cardinality $|W| \leq n/2$, $|N(W)| \geq c|W|$, where $N(W)$ is the set of vertices in $V \setminus W$ that are connected to W by an edge of G.*

The following property is proved in [3, Chap. 9, Corollary 2.2]:

Lemma 2.2 *If G is a d-regular graph with n vertices and λ is the second largest eigenvalue of the adjacency matrix of G, then G is an (n, d, c) expander with $c = (d - \lambda)/2d$.*

Thus, if λ is much smaller than d (which is clearly the largest eigenvalue of the adjacency matrix of G), then G is a good expander.

Lubotzky, Phillips and Sarnak [10] (and independently Margulis [11]) have given an explicit description of a d-regular graph G with n vertices for which $\lambda \leq 2\sqrt{d-1}$, for any $d = p + 1$ and $n = q + 1$, where p and q are primes congruent to 1 modulo 4. These graphs actually have the stronger property that all their eigenvalues (except d) have absolute value at most $2\sqrt{d-1}$. We will refer to such graphs as LPS-expanders.

From the description in [10] it follows that whenever d is a constant, an LPS-expander of degree d with n vertices can be constructed deterministically in $O(n)$ time.

The following lemma, which is the main property of LPS-expanders that we will need in this paper, is proved in [3, Chap. 9, Corollary 2.5]:

Lemma 2.3 *Let $G = (V, E)$ be a d-regular graph with n vertices. Assume the absolute value of all its eigenvalues but the largest is at most λ. Then, for every two sets of vertices, A and B, of respective cardinalities a and b, we have $|e(A, B) - abd/n| \leq \lambda\sqrt{ab}$, where $e(A, B)$ is the number of edges of G connecting a vertex of A with a vertex of B.*

Corollary 2.4 *If G is an LPS-expander of degree d with n vertices, and A and B are two sets of vertices of respective cardinalities a and b, such that $ab \geq 9n^2/d$, then $e(A, B) \geq 3n$.*

Proof. The previous lemma, and the fact that $\lambda < 2\sqrt{d}$, imply that $e(A, B) \geq abd/n - 2\sqrt{abd} \geq 3n$, as is easily verified. □

3 Arrangements of Lines and Expanders

In this section we derive several geometric results that relate expanders to arrangements of lines in the plane. Let $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ be a set of n lines in the plane in general position (as defined in the Introduction), and let $\mathcal{A}(\mathcal{L})$ denote the *arrangement* of $\mathcal{L}$, that is, the planar subdivision formed by drawing the lines in $\mathcal{L}$ (see [8] for more details concerning arrangements).

A basic step in our algorithms is to construct expanders whose vertex sets are certain subsets of $\mathcal{L}$. In this section we derive several properties possessed by such expanders. Let r be a sufficiently large constant integer, and let d be the smallest integer such that $d = p + 1$, where p is a prime congruent to 1 modulo 4, and $d \geq 9r^2$. Let G be an LPS-expander of degree d whose vertex set is $\mathcal{L}$.

(Technically speaking, the size of the vertex set of such a graph has to be of the form $q + 1$, where q is a prime congruent to 1 modulo 4; thus we let n' be the smallest integer of this form which is greater than or equal to n, construct an LPS graph G' on a vertex set V' of size n', map V' onto $\mathcal{L}$ by mapping the i-th element of V' to $\ell_{i(\mathrm{mod}n)+1}$, and let G be the graph induced by that mapping. It is easily seen that, up to (small) constants of proportionality, the resulting graph obeys properties similar to those stated in Section 2. For convenience, and with no real loss of generality, we will assume that G behaves exactly as asserted in Section 2, and ignore this technical issue in what follows.)

With each edge e of G we associate the point of intersection between the two lines connected by e (which is a vertex of $\mathcal{A}(\mathcal{L})$). This yields a collection of $O(nd)$ vertices of $\mathcal{A}(\mathcal{L})$, to which we refer as *expander points*. We partition the plane into $O(d)$ vertical slabs, each containing at most n expander points, and enumerate the slabs in increasing left-to-right order.

Let $\sigma : \; u \leq x \leq v$ be one of these slabs. Let U_k (resp. V_k) denote the set of the k lowest lines along the left (resp. right) border of the slab.

Proposition 3.1 *(cf. also [2]) For each $1 \leq k \leq n$, the size of the symmetric difference $(U_k \setminus V_k) \cup (V_k \setminus U_k)$ is at most $2n/r$.*

Proof. Suppose to the contrary that $|(U_k \setminus V_k) \cup (V_k \setminus U_k)| > 2n/r$. Since $|U_k \setminus V_k| = |V_k \setminus U_k|$, the size of both sets must be greater than n/r. It is easy to check that every vertex of $\mathcal{A}(\mathcal{L})$ formed by a line in $U_k \setminus V_k$ and by a line in $V_k \setminus U_k$ must lie in the slab σ. By Corollary 2.4, at least $3n$ edges of G connect between the sets $U_k \setminus V_k$ and $V_k \setminus U_k$. Thus σ contains at least $3n$ expander points, contrary to the construction of the slabs. □

Lemma 3.2 *The slab σ can be partitioned into r trapezoids, all incident to the left and right boundaries of σ, so that each trapezoid is intersected by at most $3n/r$ lines. The partitioning can be performed in $O(n \log r)$ time, and also yields, within the same time bound, the subsets of the lines intersecting each trapezoid.*

Proof. We partition $\mathcal{L}$ into r subsets, $L_1, \ldots, L_r$, so that each set contains n/r lines, and, for any $i < r$, any line in L_i lies at $x = u$ below any line in L_{i+1}. This

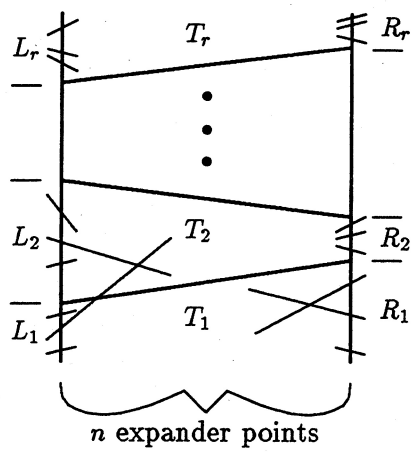

Figure 1: The trapezoidal partitioning of Lemma 3.2

partitioning can be done in time $O(n \log r)$, by repeated median finding. Similarly, we partition $\mathcal{L}$ into r equal subsets, $R_1, \ldots, R_r$, according to the permutation of the intercepts of lines of $\mathcal{L}$ along the vertical line $x = v$. For each $i < r$ we draw a line segment s_i connecting a point on the left border of σ, which separates the intercepts of the lines of L_i with that border from the intercepts of the lines of L_{i+1}, with a similar point on the right border separating between the intercepts of R_i and of R_{i+1}. This yields a partition of σ into r trapezoids, $T_1, \ldots, T_r$, ordered in increasing y-order (the topmost and bottommost of these regions are unbounded, but we will refer to them also as trapezoids). Each trapezoid has two left and right vertical edges (contained in the border lines of σ), and a top edge and a bottom edge (one of which may be missing). See Figure 1 for an illustration.

We claim that each trapezoid T_i is intersected by at most $3n/r$ lines. Indeed, the left edge of T_i is crossed by n/r lines, by construction, and the same holds for the right edge. We claim that at most $2n/r$ lines cross the bottom edge s_{i-1}. This follows by observing that each such line must belong to $(U_k \setminus V_k) \cup (V_k \setminus U_k)$, where $k = (i-1)n/r$, so the claim follows from Proposition 3.1. An identical argument shows that at most $2n/r$ lines cross the top edge s_i. We thus obtain at most $6n/r$ crossings between the lines of $\mathcal{L}$ and the boundary of T_i, and since each line that crosses T_i has two such boundary crossings, the claim follows.

Concerning the time bound asserted in the lemma, we already argued that the partitioning of σ into these trapezoids can be done in time $O(n \log r)$. Next, for each line $\ell \in \mathcal{L}$, we locate the trapezoid T_l whose left edge is crossed by ℓ and the trapezoid T_r whose right edge is crossed by ℓ. Then ℓ crosses exactly those trapezoids that lie between T_l and T_r, and we add ℓ to the lists of these trapezoids. Clearly, all this can be easily done in time $O(n \log r)$. □

By estimating the number of vertices of $\mathcal{A}(\mathcal{L})$ within each trapezoid, and multiplying by r, we easily obtain:

Corollary 3.3 *The number of vertices of $\mathcal{A}(\mathcal{L})$ within such a slab is less than $9n^2/2r$.*

The following partitioning theorem is obtained by applying Lemma 3.2 to each of the $O(r^2)$ vertical slabs.

Theorem 3.4 *The plane can be partitioned into $O(r^3)$ 'vertical trapezoids' (each having two vertical edges), so that each trapezoid is intersected by at most $3n/r$ lines. The partitioning can be performed in $O(nr^2 \log r)$ time, and also yields, within the same time bound, the subsets of the lines intersecting each trapezoid.*

4 An Optimal Slope Selection Algorithm

In this section we present our slope selection algorithm. We do it in two stages: First we describe a simple version that runs in time $O(n \log^2 n)$ time, and then we plug into it the approximate counting scheme of [6], reviewed in the Introduction, to obtain the optimal solution. Our $O(n \log^2 n)$ version is similar in spirit to the algorithm of Chazelle et al. [4], with the main difference that we use expanders, instead of the ϵ-nets used in [4], to compute certain partitionings that are required by the algorithm.

Let $\mathcal{L}$ and v_k be as in the Introduction. We 'zoom in' on v_k in $O(\log n)$ stages. At the beginning of the $(j+1)$-st stage, $j \geq 0$, we will have already found a vertical slab $\sigma_j = [u_j, v_j]$ which contains only $O(n^2 \rho^j)$ vertices of $\mathcal{A}(\mathcal{L})$, including v_k, for some constant $\rho < 1$. We will have also found a (partial) partitioning $\mathcal{T}^{(j)}$ of σ_j into t_j 'vertical trapezoids' (each having two vertical edges), $T_1^{(j)}, \ldots, T_{t_j}^{(j)}$, and t_j corresponding *non empty* subsets of $\mathcal{L}$, denoted $\mathcal{L}_1^{(j)}, \ldots, \mathcal{L}_{t_j}^{(j)}$, where $\mathcal{L}_i^{(j)}$ is the subset of lines of $\mathcal{L}$ that cross $T_i^{(j)}$ (we call it the *crossing list* of $T_i^{(j)}$), such that the following properties hold, for some constant parameter r:

(i) $\bigcup_i \mathcal{L}_i^{(j)} = \mathcal{L}$;

(ii) $|\mathcal{L}_i^{(j)}| \leq 3^j n/r^j$, for $i = 1, \ldots, t_j$;

(iii) $\sum_{i=1}^{t_j} |\mathcal{L}_i^{(j)}| \leq 3n$; and

(iv) for every vertex p of $\mathcal{A}(\mathcal{L})$ that lies in σ_j, there exists an index $1 \leq i \leq t_j$ such that the two lines meeting at p belong to $\mathcal{L}_i^{(j)}$.

Property (iv) is a consequence of the definition of the sets $\mathcal{L}_i^{(j)}$ in terms of a partitioning of σ_j into trapezoids (the partitioning does not necessarily cover all of σ_j, but it covers all the vertices of $\mathcal{A}(\mathcal{L})$ within σ_j, as will follow from the construction given below).

Note that properties (ii) and (iii) imply that the number of vertices of $\mathcal{A}(\mathcal{L})$ within σ_j is at most

$$\sum_{i=1}^{t_j} \binom{|\mathcal{L}_i^{(j)}|}{2} \leq \frac{1}{2} \max_i |\mathcal{L}_i^{(j)}| \cdot \sum_{i=1}^{t_j} |\mathcal{L}_i^{(j)}| \leq \frac{3}{2} \cdot \frac{3^j n^2}{r^j} .$$

We initialize the algorithm by setting $\sigma_0 = (-\infty, +\infty)$, $t_0 = 1$ (so the partitioning of σ_0 consists just of itself), and thus $\mathcal{L}_1^{(0)} = \mathcal{L}$. Properties (i)–(iv) clearly hold at this stage.

We now describe the j-th stage. For each $i = 1, \ldots, t_{j-1}$, let $G_i^{(j-1)}$ be an LPS-expander of degree $d \approx 9r^2$ over the vertex set $\mathcal{L}_i^{(j-1)}$; put $m_i = |\mathcal{L}_i^{(j-1)}|$, for $i = 1, \ldots, t_{j-1}$. The number of edges in $G_i^{(j-1)}$ is $O(m_i r^2)$. Thus, by property (iii), the total number of edges of all expanders constructed in this stage is $O(nr^2)$.

First, here is a brief outline of the j-th stage. We first refine the partitioning $\mathcal{T}^{(j-1)}$ of σ_{j-1} by cutting each of its trapezoids into a constant number of smaller (vertical) trapezoids that are each crossed by at most $3^j n/r^j$ lines of $\mathcal{L}$. Let $\overline{\mathcal{T}^{(j)}}$ denote the resulting finer partitioning of σ_{j-1}, after removing all new trapezoids whose crossing lists are empty. Obviously, properties (i), (ii), and (iv) now hold for the collection of crossing lists of the trapezoids of $\overline{\mathcal{T}^{(j)}}$ (with j instead of $j-1$), but property (iii) may fail to hold—the sum of the sizes of the crossing lists is still linear, since each trapezoid of $\mathcal{T}^{(j-1)}$ was divided into a constant number of smaller trapezoids, but the constant may be larger than 3. We enforce property (iii) by narrowing the slab associated with $\overline{\mathcal{T}^{(j)}}$ (i.e., σ_{j-1}), and restricting $\overline{\mathcal{T}^{(j)}}$ to the resulting narrower slab, which we denote by σ_j, to obtain $\mathcal{T}^{(j)}$.

We next describe in more detail the two main components of the j-th stage, i.e., constructing $\overline{\mathcal{T}^{(j)}}$, and computing σ_j.

To obtain a new partitioning $\overline{\mathcal{T}^{(j)}}$ from $\mathcal{T}^{(j-1)}$, let $T_i^{(j-1)}$ be a trapezoid of $\mathcal{T}^{(j-1)}$ ($T_i^{(j-1)}$ is either a vertical trapezoid or a degenerate vertical trapezoid, i.e., a 'vertical triangle' with one vertical side). Let $P_i^{(j-1)}$ denote the set of expander points of $G_i^{(j-1)}$ that fall within the vertical slab determined by the two vertical edges (or, in case of a triangle, one vertical edge and the opposite vertex) of $T_i^{(j-1)}$. We partition this slab into $O(r^3)$ vertical trapezoids (namely, $O(r^2)$ vertical sub-slabs $\times$ r trapezoids per slab), as in Theorem 3.4. The number of lines of $\mathcal{L}_i^{(j-1)}$ (and thus of $\mathcal{L}$) crossing each of the resulting trapezoids is at most $3|\mathcal{L}_i^{(j-1)}|/r \leq 3^j n/r^j$. The contribution of $T_i^{(j-1)}$ to the partitioning $\overline{\mathcal{T}^{(j)}}$ consists of the portions of the resulting trapezoids that are contained within $T_i^{(j-1)}$. Each of these portions is then cut (if necessary) into at most three vertical trapezoids or triangles (by adding at most two vertical edges from appropriate points on the top or bottom edges of $T_i^{(j-1)}$), and only those trapezoids with non empty crossing lists are retained. This portion of the j-th stage takes only linear time.

As to the second component, modifying the terminology of [4], define a *critical point* to be an intersection point between a line of $\mathcal{L}$ and an edge of some trapezoid of $\overline{\mathcal{T}^{(j)}}$. The number of critical points is clearly linear, since it is at most twice the sum of the sizes of the crossing lists of the trapezoids in $\overline{\mathcal{T}^{(j)}}$. Moreover, it is trivial to obtain the critical points from all the crossing lists, in linear time. Divide σ_{j-1} into a constant number of vertical sub-slabs, each containing at most n critical points. (Note that along any vertical line in the interior of σ_{j-1} the number of critical points is at most n.) Now use the inversion-counting oracle (or its approximating counterpart—see below), to perform a binary search among these sub-slabs, to determine which of them contains v_k. Denote that slab by σ_j, and restrict $\overline{\mathcal{T}^{(j)}}$ to σ_j to obtain $\mathcal{T}^{(j)}$. Note that this last step does not increase the number of trapezoids; on the contrary, some of the crossing lists of the shrunken trapezoids may now be empty, in which case we remove these empty trapezoids from $\mathcal{T}^{(j)}$. Note that the number of critical points on the borders of σ_j is exactly $2n$. Thus the total number of critical points in σ_j is at most $3n$ which implies property (iii). The cost of this part of the j-th stage is dominated by the cost of the oracle calls, which is $O(n \log n)$ (if we use the exact oracle, since there are only a constant number of calls in each stage).

Thus after $t = O(\log n)$ stages we are left with a slab σ_t containing v_k, with an associated partial partitioning of σ_t into trapezoids, each crossed by only a constant number of lines. We can thus afford to compute, by brute force, all intersection points within each crossing list. The union U of these collections of intersection points, whose size is only $O(n)$ (by Property (iii)), gives us all the vertices of $\mathcal{A}(\mathcal{L})$ within σ_t. Moreover, we know from the oracle calls how many vertices of $\mathcal{A}(\mathcal{L})$ lie to the left of σ_t, so we know the rank of v_k within U, and we can thus find v_k using a final linear-time selection algorithm. Since there are $O(\log n)$ stages and each takes $O(n \log n)$ time, the entire algorithm takes $O(n \log^2 n)$ time.

As observed in the Introduction, since our algorithm generates only $O(\log n)$ oracle calls, we can plug into it the approximate counting scheme of [6], as described in

the Introduction, to obtain an optimal $O(n \log n)$ solution.

5 Conclusion

In this paper we have presented a new optimal slope selection algorithm, which is based on expander graphs. The method is deterministic, avoids the use of parametric searching (and thus also the need to use complex parallel and generic sorting schemes), and has a clear and explicit geometric interpretation. Since all known optimal deterministic solutions to this problem require the use of the approximate counting scheme of [6], it is best to compare the various approaches in terms of their $O(n \log^2 n)$ versions, where this scheme is not used. We feel that our solution is the simplest and easiest to implement—the algorithm is straightforward, and requires no special techniques or data structures. The most involved part of the algorithm is a median-finder, which is needed for partitioning the current slab into sub-slabs, and for decomposing a slab into trapezoids.

The slope selection problem has been identified as one of the simplest problems in computational geometry that requires the use of parametric searching, which is one of the reasons for the attention that this problem has received. Our algorithm shows that parametric searching can be replaced by an 'expander-guided' search, resulting in a considerably simplified approach.

Among the open problems that this paper raises, we would like to mention that of finding a simpler approximate counting scheme (for the number of vertices of an arrangement of lines to the left of a query vertical line) than the one given in [6]. Another direction for further research is to find other applications and extensions of Theorem 3.4. Essentially, the partitioning described in Section 3 is a $1/r$-cutting of $\mathcal{A}(\mathcal{L})$ (in the sense of [13]), but the number of its trapezoids is $O(r^3)$ rather than the optimal number $O(r^2)$. Since we need only the subset of trapezoids within a single vertical slab, this does not cause any problems for our algorithm (the same applies to the algorithm of [2]). Can expanders be used to obtain cuttings of smaller size, by algorithms that are simpler than those currently known for that problem? We remark that Theorem 3.4 can be extended to higher dimensions, to yield $1/r$-cuttings, unfortunately of suboptimal size, for arrangements of hyperplanes in d dimensions. As an application, we can extend our technique to solve the 'slope-selection' problem in higher dimensions: Given an arrangement of n hyperplanes in d-space, and a parameter k, find the vertex of the arrangement with the k-th smallest x_1-coordinate. The running time of the resulting algorithm is close to $O(n^{d-1})$.

Acknowledgement

We thank Noga Alon for several helpful discussions concerning expanders and their applications.

References

[1] M. Ajtai, J. Komlós and E. Szemerédi, An $O(n \log n)$ sorting network, *Combinatorica* 3 (1983), 1–19.

[2] M. Ajtai and N. Megiddo, A deterministic Poly(log log n)-time n-processor algorithm for linear programming in fixed dimension, *Proc. 24th ACM Symp. on Theory of Computing*, 1992, 327–338.

[3] N. Alon and J. Spencer, *The Probabilistic Method*, Wiley-Interscience. New York, 1992.

[4] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Diameter, width, closest line-pair, and parametric searching *Proc. 8th ACM Symp. on Computational Geometry*, 1992, 120–129.

[5] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 34 (1987), 200–208.

[6] R. Cole, J. Salowe, W. Steiger and E. Szemerédi, Optimal slope selection, *SIAM J. Computing* 18 (1989), 792–810.

[7] M. Dillencourt, D. Mount and N. Netanyahu, A randomized algorithm for slope selection, *Int. J. Comput. Geom. and Appls* 2 (1992), 1–27.

[8] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[9] M. Katz and M. Sharir, An expander-based approach to geometric optimization, *Proc. 9th ACM Symp. on Computational Geometry*, 1993, 198–207.

[10] A. Lubotzky, R. Phillips and P. Sarnak, Explicit expanders and the Ramanujan conjectures, *Proc. 18th ACM Symp. on Theory of Computing*, 1986, 240–246. See also: Ramanujan graphs, *Combinatorica* 8 (1988), 261–277.

[11] G.A. Margulis, Explicit group-theoretical constructions of combinatorial schemes and their applications to the design of expanders and superconcentrators, *Problemy Peredachi Informatsii* 24 (1988), 51–60 (in Russian). English translation in *Problems of Information Transmission* 24, 39–46.

[12] J. Matoušek, Randomized optimal algorithm for slope selection, *Inf. Proc. Letters* 39 (1991), 183–187.

[13] J. Matoušek, Cutting hyperplane arrangements, *Discrete Comput. Geom.* 6 (1991), 385–406.

[14] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* 30 (1983), 852–865.