Good Splitters with Applications to Ray Shooting (Extended Abstract)

Reuven Bar Yehuda[*] and Sergio Fogel
Computer Science Department
Technion - IIT, Haifa 32000, Israel

1 Introduction

In this paper we study some problems related to ray shooting: Given a set of objects in the plane, preprocess them in such a way that queries about a ray can be answered efficiently. The objects that we consider are either lines or segments, and the queries that can be asked are the number of intersections, the first object hit, or the k first objects hit.

We solve all these problem using ES-trees, a data structure proposed by Matoušek and Welzl [11], which takes $O(n \log n)$ space and is built in time $O(n^{1.5} \log n)$. Although other data structures can give us similar space and query time complexities, ES-trees have a much lower preprocessing time.

ES-trees are found by repeatedly extracting long increasing subsequences from a sequence of numbers. We will show that a factor of $\log n$ can be saved in the building of an ES-tree by using a more efficient algorithm to extract increasing subsequences. In particular, we will show that after a preparation taking time $O(n \log n)$, an increasing subsequence of length $O(\sqrt{n})$ of a sequence of n numbers can be found and deleted in time $O(n)$.

We say that a data structure has complexity $(P(n), S(n), Q(n))$ if the space taken by the structure is $O(S(n))$, it can be built in $O(P(n))$ time and the time needed for answering a query is $O(Q(n))$. The complexity bounds that we achieve for ray shooting problems are as follows:

For ray-shooting among lines:

Counting $(n^{1.5}, n \log n, \sqrt{n} \log n)$.
  Previous result [11]: $(n^{1.5} \log n, n \log n, \sqrt{n} \log n)$.

First Hit $(n^{1.5}, n \log n, \sqrt{n} \log n)$
  Previous result [1] : $(n^{1.5} \log^{4.3} n, n \log^2 n, \sqrt{n} \log n)$

First k Hits for a veritcal ray $(n^{1.5}, n \log n, \sqrt{n} \log n + k \log n)$

First k Hits (general) $(n^{1.5}, n \log^2 n, \sqrt{n} \log n + k \log^2 n)$

and for ray-shooting among segments:

Counting $(n^{1.5}, n \log^2 n, \sqrt{n} \log^2 n)$

[*]The research of this author was partially supported by Technion VPR Fund— Albert Einstein Research Fund

First Hit (non-intersecting segments) $(n^{1.5}, n\log^2 n, \sqrt{n}\log^2 n)$
Previous result: [2] $(n^{1.5}\log^{4.3} n, n\log^3 n, \sqrt{n}\log^2 n)$.

First Hit (general) $(n^{1.5}\log^{4.3} n, n\alpha(n)\log^3 n, \sqrt{n\alpha(n)}\log^2 n)$
Previous result: [2] $(n^{1.5}\log^{4.3} n, n\alpha(n)\log^4 n, \sqrt{n\alpha(n)}\log^2 n)$.

First k Hits (non-intersecting segments) $(n^{1.5}, n\log^2 n, \sqrt{n}\log^2 n + k\log^2 n)$

In many cases only a limited number of queries are asked, and preprocessing time becomes significant. We show how the structure of [11] can be modified in order to improve the preprocessing by a factor of $O(n^\alpha)$ at the expense of increasing the query time by the same factor. This applies to all the algorithms above. Trading off allows us to fine tune the algorithm when we know in advance how many queries will be asked.

Many recent algorithms in computational geometry are based on spanning paths with a low stabbing number. We show that an ES-tree is in fact a compact representation of a family of $O(n)$ spanning paths in the plane of stabbing number $O(\sqrt{n})$. Similar complexity bounds for constructing families of $O(\log n)$ spanning paths of stabbing number $O(\sqrt{n})$ have been previously achieved by Edelsbrunner et al. [8] and Agarwal [1]. However, those algorithms are fairly complex, and in most cases, the number of paths does not affect the complexity of the application.

For ray shooting among planes in 3-space, we cannot use ES-trees since they give too high query times. However, using spanning paths of a low stabbing number we achieve a query time of $O(n^{2/3}\log^2 n)$ and space complexity of $O(n\log n)$. The preprocessing time is the time required for building a spanning path for a set of points in the space with stabbing number $O(n^{2/3})$.

We have omitted all proofs from this short version of the paper. They can be found in the full paper [3].

2 Extracting Increasing Subsequences

Let $A = (a_1, a_2 \ldots a_n)$ be a sequence of n real numbers. Let $A' = (a_{i_1}, a_{i_2} \ldots a_{i_k})$. A' is a subsequence of A if $i_1 < i_2 \ldots < i_k$. A' is called k-increasing if $a_{i_1} \leq a_{i_2} \leq \ldots \leq a_{i_k}$ and k-decreasing if $a_{i_1} \geq a_{i_2} \geq \ldots \geq a_{i_k}$.

Theorem 1 *After $O(n\log n)$ preprocessing, a k-increasing subsequence can be found and deleted from a data structure (and the data structure updated) in time $O(n + k^2)$. Alternatively, this can be done in amortized time $O(k^2\log n)$.*

Proof: Omitted.

Corollary 2 *Any sequence of n real numbers can be partitioned into $2\lfloor\sqrt{n}\rfloor$ monotone subsequences in time $O(n^{1.5})$. The subsequences can be all chosen to be small (of length not exceeding $\lceil\sqrt{n}\rceil$).*

Proof: Omitted.

3 Efficiently Building Good Splitters

Matoušek and Welzl [11] have proposed a data structure called ES-Tree which takes $O(n\log n)$ space and allows to answer halfplane range queries in time $O(\sqrt{n}\log n)$. By a result of Chazelle [4], this bound is nearly optimal. Even though a similar result was known before, the interesting feature of ES-trees is the fact that they can be built efficiently. The algorithm shown in [11] for building an ES-tree works in time $O(n^{1.5}\log n)$. We will show how to improve this time to $O(n^{1.5})$ using the algorithms of the previous section.

We now give a brief description of ES-trees. The reader is referred to [11] for a complete description. At the root of the tree we will store a vertical line b, and two set systems[1] of lines $\mathcal{T}_L$ and $\mathcal{T}_R$, such that the sets in $\mathcal{T}_L$ (and those in $\mathcal{T}_R$) are disjoint, and such that for each $L \in \mathcal{T}_L$ no two lines in L intersect to the left of b (and no two sets in $R \in \mathcal{T}_R$ intersect to the right of R). The idea is that if a point is to the right of b, then it is easy to determine the number of lines above it in any set $L \in \mathcal{T}_L$ by doing a binary search. Other queries related to ray shooting will also be simplified in a set $L \in \mathcal{T}_L$.

We want to keep the number of sets in $\mathcal{T}_L$ (and $\mathcal{T}_R$) small in order to do few searches. Not all lines will participate in $\mathcal{T}_L$ (resp. $\mathcal{T}_R$), so we will keep pointers to recursively defined structures for the lines not participating in $\mathcal{T}_L$ and for those not participating in $\mathcal{T}_L$. We will want $\mathcal{T}_L$ and $\mathcal{T}_R$ to be roughly balanced, and to cover among them all the lines of H in order to keep the depth of the tree low. A triple $(\mathcal{T}_L, \mathcal{T}_R, b)$ fulfilling all those requirements is called a *good splitter*. When given a query point p, if the point is to the right (resp. left) of b, we will find the number of lines above p (or answer any other query) in each $L_i \in \mathcal{T}_L$ (resp. $R_i \in \mathcal{T}_L$), and apply the algorithm recursively for the structure of the lines not in $\mathcal{T}_L$ (resp $\mathcal{T}_R$). We call this recursive application a 'visit' of the ES-Tree.

In [11], a good splitter is found using the algorithm SPLIT. The bottleneck of that algorithm is repeatedly extracting k-increasing and k-decreasing subsequences. Using theorem 1, algorithm SPLIT can be implemented in time $O(n \log^2 n + nk)$. This means that an ES-tree can be found in time $O(n^{1.5})$.

Theorem 3 *Counting the number of lines above a query point can be done using $O(n \log n)$ space, $O(n^{1.5})$ preprocessing, and query time $O(\sqrt{n} \log n)$.*

Counting the number of lines above a query point is the dual of the halfplane counting problem. Therefore the halfplane counting problem can be solved with complexity $(n^{1.5}, n \log n, \sqrt{n} \log n)$.

4 Applications of ES-trees

Let S be a set of points in the plane, and P a spanning (polygonal) path on S. We say that a line ℓ stabs P if it cuts one or more edges of P. The stabbing number of P is the maximum number of segments of P cut by any line. In [6] it is proved that a spanning path with stabbing number $O(\sqrt{n})$ always exists, and can be computed in polynomial time. A family $P_1, \ldots, P_m$ of spanning paths of S has stabbing number s if for every straight line ℓ there is a path P_i that is intersected by ℓ at most s times. Edelsbrunner et al. [8] have shown how to build a family of $O(\log n)$ spanning paths with stabbing number $O(\sqrt{n} \log^2 n)$. Their algorithm is randomized and runs in time $O(n^{1.5} \log^2 n)$. Agarwal [1] later improved this to a deterministic algorithm that builds a family of $O(\log n)$ spanning paths of stabbing number $O(\sqrt{n})$ in time $O(n^{1.5} \log^{4.3} n)$.

We will show that the dual of an ES-tree is a family of $O(n)$ paths with stabbing number $O(\sqrt{n})$, taking space $O(n \log n)$. Moreover, given a line ℓ, we can compute in $O(\log n)$ time a path intersected by ℓ $O(\sqrt{n})$ times.

When we visit an ES-tree, we search $O(\sqrt{n})$ ordered lists of lines; call those lists $S_1 \ldots S_m$. The first lines of each S_i are below p and all the subsequent ones are above it, or vice versa. Let ℓ be the dual of p, and $\bar{S}_i$ the ordered list of the duals of the lines of S_i. Since the dual transformation preserves the 'above' relation, the first points of each $\bar{S}_i$ are below ℓ, and the last ones are above it. In other words, ℓ stabs the polygonal path defined by the points of $\bar{S}_i$ only once.

For a set of points P, we will build an ES-tree for the duals of the points in P. When receiving a query line ℓ, we will also dualize it and travel with it along the ES-tree. If $S_1 \ldots S_m$ are the lists of lines visited and we add

[1] A set system is a set of sets. If S is a set system, $|S|$ is its cardinality, and $||S||$ is the cardinality of the union of the sets in S.

dummy segments by joining together all the $\bar{S}_i$s, we have ordered the input points into a path such that ℓ stabs it $O(\sqrt{n})$ times. Of course, for each different path in the ES-tree there is a different spanning path, but we can compute the spanning path as we travel along the tree.

We can use ES-trees in order to solve problems which have solutions based on spanning paths with low stabbing number. The idea is that ray shooting from 'outside' the lines or the segments is easier, and can be accomplished in logarithmic or polylogarthmic time. For ray shooting among lines, shooting from outside will mean shooting from a point of an unbounded face of the arrangement of the lines. In segments it will mean that the line containing the input ray is below all the left endpoints of the segments.

For each input ray we will partition the input into $O(\sqrt{n})$ subproblems, in such a way that the query ray is outside the lines or segments in the subproblems.

The reader is referred to the full paper for a complete description of the data structures used for each of the problems.

References

[1] Agarwal, P. K. A Deterministic Algorithm for Partitioning of Lines and its Applications. In *5th Ann. ACM Conf. On Computational Geometry.*, pp. 11–22, 1989.

[2] Agarwal, P. K. Ray Shooting and Other Applications of Spanning Trees with Low Stabbing Number. In *5th Ann. ACM Conf. On Computational Geometry.*, pp. 315–325, 1989.

[3] Bar Yehuda, R. and S. Fogel. *Better Constructions and New Applications of Good Splitters.* 1990 (in preparation).

[4] Chazelle, B. Polytope Range Searching and Integral Geometry. In *Proc. 28th Ann. IEEE Symp. Found. Comp. Sci.*, pp. 1–10, 1987.

[5] Chazelle, B. and L. Guibas. Visibility and Intersection Problems in Plane Geometry. In *5th Ann. ACM Conf. On Computational Geometry.*, pp. 135–146, 1989.

[6] Chazelle, B. and E. Welzl. Quasi-Optimal Range Searching in Spaces of Finite VC-Dimension. *Discrete and Computational Geometry*, 4:467–489, 1989.

[7] Edelsbrunner, H. *Algorithms in Combinatorial Geometry. EATCS Monographs in Theoretical Computer Science*, Springer Verlag, Berlin, 1987.

[8] Edelsbrunner, H., L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly Representing Arrangements of Lines or Segments. *Discrete and Computational Geometry*, 4:433–466, 1989.

[9] Edelsbrunner,H. and E. Welzl. Halfplanar Range Search in Linear Space and $O(n^{0.695})$ Query Time. *Information Processing Letters*, 23:289–293, 1986.

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

[11] Matoušek, J. and E. Welzl. Good Splitters for Counting Points in Triangles. In *5th Ann. ACM Conf. On Computational Geometry.*, pp. 124–130, 1989.