On the Complexity of Linear Search Tree Programs for Searching (extended abstract)

David Dobkin and Richard J. Lipton

Research Report #96

Department of Computer Science
Yale University
New Haven, Connecticut 06520

This research was supported in part by the Office of Naval Research under grants N00014-75-C-0450 and N00014-75-C-0752.

One of the most fundamental operations performed on a computer is searching. Generally a set is given in order and probes are then made to determine whether a new element belongs to the set or where it must be inserted in the set to preserve the set's order. Knuth [10] provides a detailed account of known methods of searching and poses an open problem that has inspired the current research direction. The problem involves searching sets that do not have a natural ordering. Such a set might consist of a set of geometric objects in Euclidean space, and we may wish to ask on which objects a new point lies or where it might occur in the partition of space introduced by such objects. Problems of this type arise naturally in such diverse areas as numerical analysis, artificial intelligence, information retrieval, coding theory, and operations research. In previous papers [3,4] we have considered such problems in attempts to derive upper bounds on their complexity and to find reasonable models in which to prove lower bounds on their complexity. In the current abstract, we report on research that extends many of these results. Our principal focus here will be on problems of a form similar to that of the knapsack problem as studied elsewhere [2,4]. We will show that any algorithm for solving the n-dimensional knapsack problem requires at least $\frac{1}{2} n^2$ operations for almost all inputs using the model of computation that is actually used to solve such problems. Furthermore, this bound holds even if n, the dimension of the problem, is given in advance and unlimited preprocessing is allowed to derive the algorithm before inputs are

received. The knapsack problem is characterized as a problem of searching the partition of n-dimensional Euclidean space by a particular set of hyperplanes. We then generalize the problem by considering an arbitrary set of hyperplanes in Euclidean space. In this case, even the results for $n = 2$ on the complexity of searching a set of lines in the plane are surprising. It appears reasonable to conjecture here that all sets of lines in general position are of approximately equal complexity with regard to searching. This is, however, not the case and we are able to show that a hierarchy exists with respect to searching complexity with an exponential gap between the least and the most complex sets of lines.

In the next section we set forth our notation and model problems. Following this a study of the general problem in n dimensions and its relation to the knapsack problem is presented resulting in a lower bound of $\frac{1}{2} n^2$ on this problem. Methods of extending this bound further are also mentioned. From this section is also developed a general statement of the problem of geometric search of geometric objects in n-dimensional Euclidean space. This statement suggests interesting problems that must be tackled in order to derive faster upper bounds on knapsack-type searching problems. The final section is devoted to a study of some results that occur when the 2-dimensional version of this problem is considered.

Our basic searching problem will be presented in terms of a set of hyperplanes $H_1,\ldots,H_m$ in n-dimensional Euclidean space E^n. Given such a set, which partitions this space into many regions, we shall probe the set in order

to determine whether a new point lies on one of the original hyperplanes or in which region of space it lies. This is a very common problem type in pattern recognition [11], coding theory [1], numerical analysis [6], and operations research [3,4,7,14]. In different situations, we will allow as probes queries of varying types. The most widely used types will be the case where a query is restricted to a concern for the location of a given point with respect to one of the original hyperplanes or an arbitrary hyperplane in E^n. As previously observed [2], we may simulate a query concerning the location of a given point with respect to any hypersurface of degree p by p linear queries. Thus lower bounds obtained here will apply to a wider class of algorithms. With this model, our algorithms can be considered as tree programs allowing statements of the form

$$L_i: \text{ if } f(x) \ R \ 0 \text{ then go to } L_j \text{ else go to } L_k$$

where f(x) is a linear, affine function of the input and R is one of the relations $\{<,=,>\}$. In this case, we shall refer to such algorithms as linear tree programs: The linear tree program of minimum depth will be used as a measure of the complexity of our operations. In some applications, we will wish to consider restricted linear tree programs where f(x) is required to define one of the original objects in our set. In this case, we will refer to the restricted linear tree program complexity of our operations. Previously, programs of these general types have been considered [2,4,5,12,13,16,18].

To conclude our preliminaries, we wish to introduce the knapsack problem in n dimensions (KS_n) as a problem in the searching of hyperplanes in E^n. This problem is generally stated [8] as: "Given an n-tuple $(x_1,\ldots,x_n)$ of

integers and another integer b, does any subset of the original set sum to b?" We might then view this as a searching problem by defining a set of hyperplanes corresponding to all possible packings of the knapsack. That is, to test whether $I \subseteq \{1,\ldots,n\}$ corresponds to a perfect packing, i.e. whether

$$\sum_{i\in I} x_i = b$$

is equivalent to determining whether the point

$$(\frac{x_1}{b}, \frac{x_2}{b},\ldots,\frac{x_n}{b})$$

lies on the hyperplane H_I defined as

$$H_I = \{\underline{y} \in E^n \mid \sum_{i\in I} y_i = 1\}.$$

As there are $2^n - 1$ potential packings, and a similar number of such sets I, we observe that solving the knapsack problem is equivalent to determining where a point in E^n lies with respect to the partition induced by these hyperplanes.

Before proceeding to our major new results on the knapsack problem, we pause to review some previous results in order to set this result in its proper framework. Previously [4], we have shown that any linear search tree for solving the knapsack problem with queries limited to the original hyperplanes requires

$$O(\frac{2^n}{\sqrt{n}})$$

queries for almost all inputs. This proof proceeded by an elementary adversary argument showing that all the hyperplanes corresponding to sets I with $\frac{n}{2}$ elements would have to be considered. We also proved a result on determining membership in a union of open sets, which will form the basis of our current result:

Theorem [4]: Let $\{A_j\}_{j\in J}$ be a set of pairwise disjoint open sets in E^n. Then any linear search tree program (with queries not necessarily restricted to the original hyperplanes) must require at least $\log_2|J|$ to determine membership in

$$\bigcup_{j\in J} A_j$$

for almost all inputs.

This result was previously used to show a lower bound of n log n on this problem [2]. The knapsack problem of dimension n consists of open sets formed as the intersections of the halfspaces determined by the original $2^n - 1$ hyperplanes. Potentially,

$$2^{2^n-1}$$

such open sets could exist, as we might have to consider all possible orientations with respect to all possible hyperplanes. Were this the case, we would be able to prove exponential lower bounds. Unfortunately, for $n = 4$, we quickly notice that the region

$$x_1 + x_2 > 1,\ x_1 + x_3 < 1,\ x_3 + x_4 > 1,\ x_2 + x_4 < 1$$

is non-existent. So we must determine which halfspaces have non-empty intersections. To do so, we must make contact with results from the theory of threshold logic. Pioneering work on this problem was done by Winder in the early 1960's [15]. As observed above, there are about

$$2^{2^n}$$

possible partitions of $2^{\{1,\ldots,n\}}$, each of which potentially gives rise to a non-empty region. We shall call a partition

$$P_A = \bigcup_{j \in A} I_j$$

acceptable if and only if there exists a point $\underline{x} \in R^n$ such that for each $j \in A$ $\underline{x}$ lies above the hyperplane

$$H_{I_j}$$

and for each $j \notin A$ $\underline{x}$ lies below

$$H_{I_j}.$$

A point $\underline{x}$ is said to be below hyperplane H_I if

$$\sum_{i \in I} x_i < 1,$$

above H_I if

$$\sum_{i \in I} x_i > 1,$$

and on H_I otherwise. Clearly, each acceptable partition corresponds to a nonempty region and two different partitions correspond to disjoint regions; hence a lower bound on the knapsack problem is given by the logarithm of the number of acceptable partitions.

A development parallel to the one given above is given in the theory of threshold logic with switching functions corresponding to arbitrary partitions and threshold functions to acceptable partitions. Furthermore, it is known that the number N(n) of threshold functions on n inputs is between

$$2^{\frac{1}{2} n^2}$$

and

$$2^{n^2}.$$

Hence, we have proven:

Theorem: Any linear search tree for the n-dimensional knapsack problem requires at least $\frac{1}{2} n^2$ queries for almost all inputs.

Corollary: Any search tree allowing for queries of degree p or less that solves the n-dimensional knapsack problem requires at least

$$\frac{n^2}{2p}$$

queries for almost all inputs.

Before proceeding to a consideration of upper bounds on this problem, we observe that this is the first lower bound of better than n log n given for an NP-complete problem under a realistic model of computation. While linear search trees do not have the full power of Turing machines, for which the P vs. NP problem was originally stated, we conjecture that this extra computing power does not assist in solving the knapsack problem. Although we have proved that at least

$$2^{\frac{1}{2}n^2}$$

regions exist that are non-empty in the partition of E^n by the knapsack hyperplanes, we also observe that almost all of these regions have very small volume. A result due to Karp [9] gives an algorithm that in time

$$O(\frac{n \log n}{\varepsilon})$$

determines whether a solution of the ε-approximate knapsack problem exists. That is, given $\{x_1,\ldots,x_n,b\}$, does there exist $I \subseteq \{1,2,\ldots,n\}$ such that

$$\sum_{i\in I} x_i$$

lies between $(1-\varepsilon)b$ and $(1+\varepsilon)b$? If we consider linear search tree programs for this problem, we can view the hyperplanes as having been extended to solids such that H_I is the set of points with

$$1-\varepsilon \leq \sum_{i\in I} x_i \leq 1+\varepsilon$$

and we then consider the partition of space induced by these solids. This partition can consist of at most

$$2^{\frac{n \log n}{\epsilon}}$$

pairwise disjoint open sets.

In order to extend this lower bound, a number of approaches are possible. For each of these it is necessary to have a more complete understanding of the knapsack partition than merely a knowledge of the number of regions it produces. For example, as we have observed [4], the

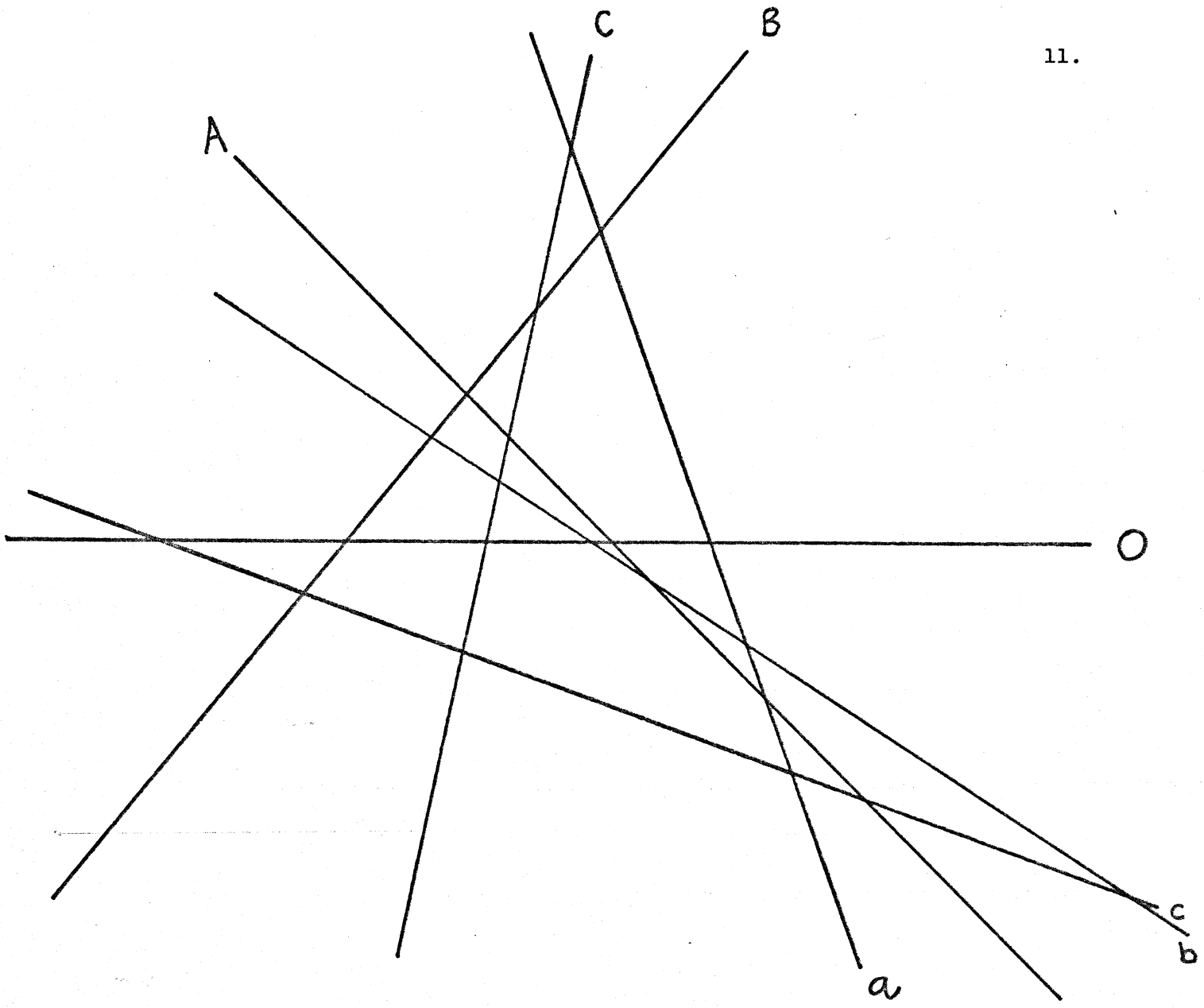
$$\binom{n}{\frac{n}{2}}$$

hyperplanes corresponding to exact packings of exactly $\frac{n}{2}$ of the original objects are all facets of a polyhedron; if we restrict our queries to the original hyperplanes, we must query about each hyperplane to determine whether a point lies within this polyhedron or not. In the case where queries can be arbitrary linear forms, this result is not known to hold nor is a better upper bound known to exist. By further analysis of subsets of this problem improved bounds may result. As we shall show below, however, these analyses are nontrivial even in 2 dimensions. The arguments above should make it clear that another view of the problem of P vs. NP is as a problem in geometry regarding the partition of n-dimensional Euclidean space by a set of hyperplanes. And the goal of this problem is to be able to find the location of arbitrary points with respect to this partition.

To begin our study of this problem, we set out to understand

partitioning of E^2 by lines. Previously, we had observed an algorithm for searching a set of n lines in the plane in 3 log n queries using a linear search tree [3]. We can also show that 2 log n is a lower bound on this problem. But in the case of linear search trees with queries restricted to the original lines, the results obtained are far more precise. It is clear that certain sets of lines in the plane are very easy to search. For example, suppose we have a set of n parallel lines or a set of n lines each passing through a common point. Each of these sets can be easily searched in log n queries by taking advantage of their natural ordering. In order to make the problem most interesting, we then restrict ourselves to lines in general position. That is, no three lines have a single point in common and each pair of lines have a non-empty intersection. In this case, it would seem reasonable to assume that all sets of n lines require the same number of queries in any linear search tree algorithm. Furthermore, since the general position restriction appears to rule out any natural orderings on the lines, it would seem reasonable to assume that O(n) queries are required. Furthermore, since the general position restriction appears to rule out any natural orderings on the lines, it would seem reasonable to assume that O(n) queries are required. Furthermore, we can add further credence to this conjecture by demonstrating, for each n, a set of n lines that require n queries for their search. This set is given as a set of n lines forming an n-gon where, to determine membership in the interior, on the boundary, or in the exterior of the n-gon, n queries are required.

We can, however, construct a set of 7 lines that can be searched in 6 queries. This set is given as:



We observe that all intersections of A, B, and C occur above O and all intersections of a, b, and c occur below O. Hence, if we wish to locate x in this partition, we can begin by locating x with respect to O. If x is above O, we can locate x with respect to a, b, and c in 2 queries and with respect to A, B, and C in 3 queries. A similar result holds if x lies below O. Thus, not all sets of lines are of equal complexity. Furthermore, we can extend

this argument to demonstrate, for each n, the existence of a set of n lines that can be searched in $\frac{1}{2}\log^2 n$ queries (n sufficiently large). Thus, there is an exponential gap between easiest and hardest sets of lines. This result can be extended further to:

Theorem: For any function f(n) such that $\log^2 n \leq f(n) \leq n$ and any n, there is a set of n lines that can be searched in f(n) queries but no fewer.

As a further research effort, we hope to extend these results to enable a study of partitions of higher dimensional spaces with the hope of generating better upper and/or lower bounding techniques on the knapsack problem.

References

1] D. Arden.
Private communication, 9 March 1976.

2] D. Dobkin.
A non-linear lower bound on linear search tree programs for solving knapsack problems.
Yale Computer Science Research Report #52, 1975. To appear in the Journal of Computer and System Science.

3] D. Dobkin and R. Lipton.
On some generalizations of binary search.
ACM Symposium on the Theory of Computing, May 1974. To appear in the SIAM Journal of Computing.

4] D. Dobkin and R. Lipton.
On the complexity of computations under varying sets of primitive operations.
Automata and Formal Languages. Springer-Verlag Lecture Notes in Computer Science #33, 1975.

5] D. Dobkin and R. Lipton.
The complexity of searching lines in the plane.
Unpublished manuscript.

6] J. George.
A Computer Implementation of the Finite Element Method.
PhD thesis, Stanford University, 1971.

7] E. Horowitz and S. Sahni.
Computing partitions with applications to the knapsack problem.
Cornell University Computer Science Technical Report 72-134, 1972.

8] R. Karp.
Reducibility among combinatorial problems.
In R. Miller and J. Thatcher, editors, Complexity of Computer Computations. Plenum Press, 1972.

9] R. Karp.
Private communication, 24 March 1976.

10] D. Knuth.
The Art of Computer Programming. Volume 3: Sorting and Searching.
Addison-Wesley, 1973.

11] S. Levinson.
Private communication, 11 February 1974.

12] M. Rabin.
Proving simultaneous positivity of linear forms.
Journal of Computer and System Science 6:639-650, 1972.

13] E. M. Reinhold.
Computing the maxima and the median.
12th Annual Symposium on Switching and Automata Theory, 216-218. 1971.

14] M. Shamos.
Computational Geometry.
To appear as a PhD thesis, Yale University.

15] E. Sheng.
Threshold Logic.
Academic Press, 1969.

16] P. Spira.
Complete linear proofs of systems of linear inequalities.
Journal of Computer and System Science 6:205-216, 1972.

17] P. Spira.
On the number of comparisons
necessary to rank an element.
Courant Computer Science Symposium
7, edited by Randall Rustin.
Algorithmics Press, 1973.

18] A. Yao.
On the complexity of comparison
problems using linear functions.
16th Annual Symposium on Switching
and Automata Theory, 85-89.
1975.