On the Complexity of Computations under Varying Sets of Primitives*

David P. Dobkin and Richard J. Lipton
Department of Computer Science
Yale University
New Haven, Connecticut 06520 USA

Research Report #42

*Portions of the research of the first author were supported by ONR Grant N00014-75-C-0450.

1. Introduction

The principal goal of research in computational complexity is the determination of tight lower bounds on the complexity, in terms of primitive operation executions, of solving problems or performing larger operations. While algorithms now exist that yield better than naive upper bounds for various operations (e.g. [1,9,11,12]), finding lower bounds for the solution of a problem using a general model has proved to be more difficult. In order to circumvent these difficulties, many authors (e.g. [3,5,6,8,13]) have chosen to work with models that place some restriction on the primitive operations that can be used or on the flow of output that can occur. Typical of the restrictions that have been placed on models are: allowing the use of only a monotone basis of functions [6,8] or requiring that all circuits be restricted to fan-out one [3,5,13]. The value of using such models is in the insights they produce into the general process of finding lower bounds; many of the actual lower bounds they produce are shown, however, to be invalid for more general models.

The goal of the current research is the study of lower bounds on the complexity of a set of searching problems under various restrictions on the nature of the primitive operation used to determine each branch within a search tree. Our model, to be described in more detail in the next section, has programs consisting of two types of statements, query statements of the form:

L_k: if f(x) R 0 then goto L_m else goto L_n

where R is one of the relations (> or =) and f is a function of restricted form on the input of x. An output statement of the form

L_s: accept (or reject)

occurs for each possible outcome of the problem.

The problems we consider all involve searching a set of geometric objects in Euclidean space to determine in which region of their partition of space a given point lies or whether the point lies in any of the given regions. Among the new results obtained are exponential lower bounds for searching for solutions to a knapsack problem, viewed as a hyperplane search problem, for various models involving restrictions on the primitive operations allowed. A non-linear (in the number of hyperplanes) lower bound is given for a generalized hyperplane search problem along with an O(n log n) bound for a problem in the plane.

2. Basic Model

Our model of computation is based on the notion of a search program. A search program P with input $(x_1,\ldots,x_n)$ is a finite list of instructions of the following three types:

1) L_k: if $f(x_1,\ldots,x_n)$ R 0 then goto L_m . $(R \in \{>,=\})$
else goto L_p

2) L_k: accept

3) L_k: reject

Control initially starts at the first instruction. An instruction of type (1) determines whether or not the indicated test is true: If it is true, then control passes to the statement with label L_m; otherwise, control passes to the statement with label L_p. An instruction of type (2) denotes that the program has halted and it has accepted the input. Correspondingly, an instruction of type (3) denotes that the program has halted and it has rejected the input.

We will restrict search programs in two distinct ways. The functions allowed in instructions of type (1) are called primitives. Often we will restrict the class of allowed primitives. We will also restrict at times the relations R allowed in instructions of type (1). Thus an equality search program can have R equal only to =. On the other hand, a linear search program can have only functions f that are linear.

The complexity measure we will use on our search programs is "time." Each possible input $(x_1,\ldots,x_n)$ determines a computation through the search program. The length of this computation is the number of steps associated with the input $(x_1,\ldots,x_n)$. We are always interested in the worst-case behavior, i.e. the maximum number of steps required by a given search program.

3. Restricted Linear Programs

In this section we will investigate the n-dimensional knapsack problem (KS_n). We can view this problem as follows: Given a point $(x_1,\ldots,x_n,b) \in E^{n+1}$ we are to determine whether or not there exists an index set I such that

$$\sum_{i\in I} x_i - b = 0.$$

The first question we ask is: If we restrict our search programs to queries of the form

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

can we show that they must take exponential time? The answer is yes:

Theorem 1. Any search program having as its primitive operation functions of the form

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

for some index set I and any tests >, =, or < must require $O(2^n)$ primitive steps to solve the n-dimensional Knapsack Problem.

Proof. We adopt an adversary approach and provide a set of data such that if less than $\binom{n}{n/2}$ primitive operations are executed the data can be altered so as to make it possible for the solution to the problem to change without changing previous results.

Our adversary will return answers to queries according to the following plan:

i) if $|I| < n/2$, then $\sum_{i\in I} x_i < b$

ii) if $|I| > n/2$, then $\sum_{i\in I} x_i > b$

iii) if $|I| = n/2$ and less than $\binom{n}{n/2} - 1$ tests on index sets of exactly n/2 elements have been done, then $\sum_{i\in I} x_i > b$.

We now make the claim that it is possible to provide three sets of data satisfying conditions (i), (ii), and (iii) such that each set yields a different result on the final query. From this claim, the theorem follows since although an algorithm knowing this adversary's strategy could eliminate all tests of index sets with cardinality not equal to n/2 the $\binom{n}{n/2} = O(2^n)$ tests of index sets of cardinality n/2 must all be performed.

Claim. Assume the last test performed on an index set of exactly n/2 elements compares $x_1 + x_2 + \ldots + x_{n/2}$ to b; then there are choices of $x_1 = \gamma$, $x_2 = x_3 = \ldots = x_{n/2} = \alpha$ and $x_{(n/2)+1} = x_{(n/2)+2} = \ldots = x_n = \beta$ for $0 < \gamma \leq \alpha \leq \beta < b$ satisfying the three conditions of the adversary and yielding any of the three

possible results $x_1 + \ldots + x_{n/2} \lessgtr b$.

Proof. The conditions (i), (ii), and (iii) can be restated as

i) $((n/2)-1)\cdot\beta < b$

ii) $\gamma + ((n/2)-1)\cdot\alpha + \beta > b$

iii) $\gamma + ((n/2)-2)\cdot\alpha + \beta > b$

and we observe that (iii) implies (ii), so that we need show only that conditions (i) and (iii) can be met along with the result of one of the cases:

Case I: $\gamma + ((n/2)-1)\cdot\alpha > b$

Case II: $\gamma + ((n/2)-1)\cdot\alpha < b$

Case III: $\gamma + ((n/2)-1)\cdot\alpha = b$.

In the first case, the choice $\gamma = \alpha = \beta = 2b/(n-1)$ works since

i) $((n/2)-1)\cdot(2b/(n-1)) = ((n-2)/(n-1))\cdot b < b$

iii) $(2b/(n-1)) + ((n/2)-2)\cdot(2b/(n-1)) + (2b/(n-1)) = (n/(n-1))\cdot b > b$

and $(2b/(n-1)) + ((n/2)-1)\cdot(2b/(n-1)) = (n/(n-1))\cdot b > b$.

The second case is handled by the choice $\gamma = b\cdot\left(\frac{2n-5}{(n-(1/2))^2}\right)$,

$\alpha = 2b\cdot\left(\frac{1}{n-2} + \frac{2}{(n-(1/2))^2}\right)$, $\beta = 2b\left(\frac{1}{n-2} - \frac{1}{(n-(1/2))^2}\right)$ since

i) $((n/2)-1)\cdot\left(2b\left(\frac{1}{n-2} - \frac{1}{(n-(1/2))^2}\right)\right) = b\cdot\left(1 - \frac{n-2}{(n-(1/2))^2}\right) < b$

iii) $b\cdot\left(\frac{2n-5}{(n-(1/2))^2}\right) + ((n/2)-2)\cdot 2b\cdot\left(\frac{1}{n-2} + \frac{2}{(n-(1/2))^2}\right) + 2b\cdot\left(\frac{1}{n-2} - \frac{1}{(n-(1/2))^2}\right)$

$= b\cdot\left(\frac{4n-11}{(n-(1/2))^2} + 1\right) > b$

and $b\cdot\left(\frac{2n-5}{(n-(1/2))^2}\right) + ((n/2)-1)\cdot 2b\cdot\left(\frac{1}{n-2} + \frac{2}{(n-(1/2))^2}\right) = b\cdot\left(\frac{4n-9}{(n-(1/2))^2} + 1\right) > b$

Finally, the choices $\gamma = \frac{2b(n-2)}{n^2-4}$, $\alpha = \frac{2bn}{n^2-4}$, $\beta = \frac{2b(n+1)}{n^2-4}$ settle the third case via

i) $((n/2)-1)\cdot\left(\frac{2b(n+1)}{n^2-4}\right) = \frac{b(n+1)}{n+2} < b$

iii) $\frac{2b(n-2)}{n^2-4} + ((n/2)-2)\cdot\left(\frac{2bn}{n^2-4}\right) + \frac{2b(n+1)}{n^2-4} = b\cdot\left(\frac{n^2-2}{n^2-4}\right) > b$

and $\frac{2b(n-2)}{n^2-4} + ((n/2)-1)\cdot\left(\frac{2bn}{n^2-4}\right) = b$. □

The result of this theorem is that any polynomial-time algorithm for solving the knapsack problem must use comparisons to hyperplanes not in the original set but generated from the original set. While such an algorithm is possible, it is unlikely to exist as a general procedure but might rather exist as a set of procedures $\{P_i\}_{i=1}^{\infty}$ such that solving the n-dimensional knapsack problem involves

using procedure P_n to generate new hyperplanes and solving the n+1-dimensional knapsack problem involves using (possibly different) procedure P_{n+1} to generate new hyperplanes. Examples of such procedures as well as a brief discussion of the implications of such a system for the question "P = NP?" are contained in [2]. The present result in conjunction with those discussions makes it extremely unlikely that P and NP are the same.

4. Linear Programs

Next we will study linear programs. That is, we will allow any tests of the form

$$f(x_1,\ldots,x_n) \gtreqless 0$$

where f is a linear function. The next theorem allows us to obtain lower bounds for the complexity of various membership problems:

Theorem 2. Any linear search tree that solves the membership problem for a disjoint union of a family $\{A_i\}_{i\in I}$ of open subsets of R^n requires at least $\log_2 |I|$ queries in the worst case.

Proof. We prove that any such search tree T with leaves $D_1,\ldots,D_r$ has $r \geq |I|$ and hence a path of depth $\geq \log_2 |I|$. The leaves partition R^n and, for each j, D_j is an accepting leaf if $D_j \subseteq \bigcup_{i\in I} A_i$ and a rejection leaf otherwise. The theorem then follows from the observation that the function Y: $I \to \{1,\ldots,r\}$ defined by Y(i)'s being the least i such that $A_i \cap D$ is non-empty is an injective function. This observation is true since if $Y(i) = Y(j) = \ell$ for $i \neq j$ then there exist distinct points x and y such that $x \in A_i \cap D_\ell$ and $y \notin A_j \cap D_\ell$. By the convexity of D_ℓ, each point on the line joining x and y lies in D_ℓ and hence is accepted as a point of $\bigcup_{i\in I} A_i$. Defining the function g: $L \to I$ by $g(Z) = k$ whenever $Z \in A_k$ yields the contradiction that g is the constant function i, since A_i is open and I is finite. □

Let us now generalize the knapsack problem (KS_n) to the generalized knapsack problem (GKS_n): We are given 2^n hyperplanes $H_1,\ldots,H_{2^n}$ in E^{n+1} space that form a simple arrangement, i.e. no n+2 hyperplanes have a common point. For each new point x we are to determine whether or not x lies in any of these hyperplanes. Note, we do not insist that the search tree determine which hyperplane x lies in, only that it determine whether or not x lies in some hyperplane.

From this result we obtain the following corollaries:

Corollary 1. The membership problem for GKS_n takes at least $O(n^2)$ queries for any search tree.

Proof. Since the hyperplanes of this problem form a simple arrangement, we can find

a family $\{A_i\}_{i\in I}$ of open subsets of R^n such that

$$x \in \bigcup_{i\in I} A_i \leftrightarrow x \in GKS_n$$

and $|I| \geq O(2^{n^2})$ [4]. The corollary then follows from the theorem. □

This result improves a lower bound of $O(n)$ due to Spira [10].

Corollary 2. (Element Uniqueness Problem.) Let E_n be the set of points in R^n that have two coordinates equal; then any algorithm for determining membership in E_n requires at least $O(n \log n)$ queries.

Proof. Solving the membership problem for E_n corresponds to solving the membership problem for the family $\bigcup_{\pi\in S_n} \{A_\pi\}$ where

$$A_\pi = \{(x_1,\ldots,x_n) \in R^n \mid x_{\pi(1)} < x_{\pi(2)} < \ldots < x_{\pi(n)}\}$$

and S_n is the set of permutations on n objects. The result then follows from $|S_n| = n!$. □

5. Equality Programs

In the previous section, we considered the problem of determining whether a point belonged to the union of a family of open sets allowing linear search programs. Here, we extend our methodology to the problem of determining whether a point belongs to the union of a family of varieties allowing search programs that determine at each step whether the point is the root of an irreducible polynomial. Before proceeding, we state some results from algebraic geometry [7] that will be necessary to our development.

Definition. A variety $V(f_1,\ldots,f_m)$ is a subset of R^n defined by

$$V(f_1,\ldots,f_m) = \{(x_1,\ldots,x_n) \in R^n \mid f_1(x_1,\ldots,x_n) = \ldots = f_m(x_1,\ldots,x_n) = 0\}$$

for polynomials $f_1,\ldots,f_m$.

Definition. The polynomials f and g are said to be equivalent iff there exists a non-zero constant λ such that $f = \lambda g$.

Fact 1. If the dimension of $V(f_1,\ldots,f_n)$ is denoted by $\dim(V(f_1,\ldots,f_n))$ then

i) $\dim(A) = 0$ if and only if A is empty

ii) if $R^n = \bigcup_{i=1}^{k} V(f_i)$, then one of the polynomials f_i is trivial

iii) if f and g are non-trivial irreducible polynomials that are not equivalent, then $\dim(V(f,g)) < \dim(V(f))$.

Theorem 3. If $f_1,\ldots,f_m$ are irreducible polynomials of n real variables that are not equivalent, then any equality search program for

$$\bigcup_{i=1}^{m} V(f_i)$$

using only irreducible polynomials requires at least m queries.

Proof. Let T be a search program of depth k that determines for any $x \in R^n$ whether $x \in \bigcup_{i=1}^{m} V(f_i)$. Suppose that a path through T makes queries as to whether $g_i(x) = 0$ for $i = k$ if $x \in \bigcup_{i=1}^{m} V(f_i)$ and $\neq 0$ for $i = k$ if $x \notin \bigcup_{i=1}^{m} V(f_i)$. We will show by dimension arguments that this is possible only if $k \geq m$.

To begin, we define the sets $F = \bigcup_{i=1}^{m} V(f_i)$ and $G_i = (\bigcup_{j=1}^{i} V(g_j))^c$ (A^c is the complement of the set A) and observe that $x \in G_{t-1}$ if and only if $k \geq t$. Furthermore, either $G_k \subseteq F$ or $G_k \subseteq F^c$ since after k queries we can determine for each x whether x belongs to F or not. Now, if $k < m$, then for some i $V(f_i)$ is not one of the sets $V(g_1),\ldots,(V(g_k)$ and $\dim(V(g_j) \cap V(f_i)) < \dim(V(f_i))$. Thus the set $G_k \cap V(f_i)$ is of the same dimension as the variety $V(f_i)$ and is non-empty. Hence $G_k \cap F \neq 0$. Assume next that $G_k \subseteq F$; then $G_k^c \cup F = R^n$. But this implies that R^n can be written as the union of non-trivial varieties, which is not true, and thus $G_k \cap F^c \neq 0$ and so k queries are insufficient for $k < m$. □

Corollary 3. Any equality search program for KS_n that uses only irreducible polynomials requires at least 2^n queries.

References

[1] Blum, Floyd, Pratt, Rivest, Tarjan. Linear time bounds for selection. JCSS 7:448-461, 1973.

[2] Dobkin, Lipton. On some generalizations of binary search. ACM Symposium on the Theory of Computing, Seattle, Washington, May 1974.

[3] Fischer, Meyer, Paterson. Lower bounds on the size of Boolean formulas. ACM Symposium on the Theory of Computing, Albuquerque, New Mexico, May 1975.

[4] Grünbaum. Convex Polytopes. Interscience Publishers, 1967.

[5] Harper, Savage. On the complexity of the marriage problem. Advances in Mathematics 9:299-312, 1972.

[6] Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication. PhD thesis, Cornell University, Ithaca, New York, 1970.

[7] Lefschetz. Algebraic Geometry. Princeton University Press, 1953.

[8] Schnorr. A lower bound on the number of additions in monotone computations of monotone rational polynomials. Unpublished manuscript.

[9] Schönhage, Strassen. Fast multiplication of large numbers (in German). Computing 1:182-196, 1966.

[10] Spira. On the number of comparisons necessary to rank an element. Computational Complexity Symposium, Courant Institute, 1973.

[11] Strassen. Gaussian elimination is not optimal. Numerische Mathematik 13: 354-6, 1969.

[12] Tarjan. Depth-first search and linear graph algorithms. SIAM Journal on Computing 1, 1972.

[13] Vilfan. The complexity of finite functions. Technical Report 97, Project MAC, MIT, 1972.