JOURNAL OF COMPUTER AND SYSTEM SCIENCES 36, 313–335 (1988)

Lower Bound Arguments with "Inaccessible" Numbers

MARTIN DIETZFELBINGER*,† AND WOLFGANG MAASS*

Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago, Chicago, Illinois 60680

Received September 23, 1986; revised March 13, 1987

The first result presented in this paper is a lower bound of $\Omega(\log n)$ for the computation time of concurrent-write parallel random access machines (PRAMs) with operation set $\{+, -,$ multiplication by constants$\}$ that recognize the "threshold set" $\{\bar{x} \in \mathbf{Z}^n \mid x_1 + \cdots + x_{n-1} \leqslant x_n\}$ for inputs from $\{0, 1, 2, ..., 2^{O(n \cdot \log n)}\}^n$. The same bound holds for PRAMs with arbitrary binary operations, if the size of the input numbers is not restricted. The second lower bound regards languages in $\mathbf{R}^n$ corresponding to KNAPSACK, MINIMUM PERFECT MATCHING, SHORTEST PATH, and TRAVELING SALESPERSON on linear decision trees (LDTs) with the restriction that the number of negative coefficients α_i in each test $\sum_{1 \leqslant i \leqslant n} \alpha_i x_i$: α_0 is bounded by $f(n)$. The lower bounds on the depth of such LDTs that recognize these languages are $\Omega(2^{\lfloor n/2f(n) \rfloor})$ for KNAPSACK and $\Omega(2^{\lfloor \sqrt{n}/4f(n) \rfloor})$ for the graph problems. The common new tool in the proofs of these lower bounds is the method of constructing "hard" instances $(x_1, ..., x_n)$ of the respective problem by building up the input numbers x_i from "mutually inaccessible" numbers, i.e., numbers of different orders of magnitude.

1. INTRODUCTION

We present two lower bound results for several number problems on standard models for parallel and sequential computers. The common new tool employed in the proofs of these lower bounds is the method of constructing "hard" instances $(x_1, ..., x_n)$ of the considered problem by building up the input numbers $x_1, ..., x_n$ from numbers that are "mutually inaccessible," i.e., numbers of different orders of magnitude from the point of view of the machine under consideration. Inputs $(x_1, ..., x_n)$ chosen in such a way that they do not solve certain linear or algebraic equations have already been used in previous lower bound arguments (see [17, 21, 12]). The inputs constructed in the proofs of this paper also have the property that they do not solve certain equations, but in addition we gain a "stability property" with respect to certain inequalities satisfied by the input $(x_1, ..., x_n)$. This stability allows us to argue that certain slight perturbations of the input $(x_1, ..., x_n)$, which arise in the particular "fooling argument," satisfy the same inequalities.

* Written under partial support by NSF Grant DCR-8504247.

† Current address: Lehrstuhl Informatik II, Universität Dortmund, D-4600 Dortmund 50, Fed. Rep. Germany. Based upon a part of the first author's Ph.D. thesis at the University of Illinois at Chicago.

0022-0000/88 $3.00

Geometrically phrased, this means that the input can be slightly changed without certain hyperplanes or algebraic varieties being crossed.

Apart from this common feature the two lower bound arguments exploit geometrical and combinatorial properties specific to the considered problems, which include KNAPSACK and SHORTEST PATH. The computational models we consider are parallel random access machines with concurrent-write memory access (CRCW-PRAMs) in Section 2 and linear decision trees (LDTs) in Section 3.

In each case, the length of the computation is analyzed in terms of the "dimension" n of the input $(x_1, ..., x_n)$, i.e., the number of input variables x_i. This corresponds to the "uniform cost criterion" for random access machines (see [1]), where one charges one unit for each step of the computation, independently of the size of the operands, and measures the computation time in terms of the number of input variables x_i (as opposed to the bitwise complexity measure, which is used, for example, in the analysis of Turing machine computations).

In the remainder of the Introduction we define the machine models and outline the results and proofs. In comparison with the preliminary version of this paper [6] we have added here the lower bound argument for PRAMs in Section 2.

The main result of Section 2 is a lower bound for a threshold problem ("is $x_1 + \cdots + x_{n-1} \leqslant x_n$?") on a standard model for a parallel computer, namely a CRCW-PRAM with restricted instruction set (addition and multiplication by constants are permitted), as has been considered, e.g., in [13, 18].

A parallel random access machine (PRAM) consists of an infinite sequence $p_1, p_2, p_3, ...$ of usual RAMs (as defined in [1]), called processors, and an infinite sequence $c_0, c_1, c_2, ...$ of common memory cells. Each processor p_i is equipped with an infinite sequence $r_0, r_1, r_2, ...$ of local memory cells (r_0 is the accumulator; each memory cell is capable of holding an integer). Each processor also has a program, i.e., a sequence of labelled instructions, which may be different for different processors and for different dimensions n of the input $(x_1, ..., x_n)$. We do require, though, that there be some upper bound $P(n)$ on the number of processors taking part in the computation on inputs of dimension n, independently of the size of the input numbers. The programs consist of read- and write-instructions (directly or indirectly addressed into local or global memory), test-instructions (to allow branching depending on the contents of the accumulator), and numerical instructions from a certain set $\{\mathrm{Op}_i \mid i \in \mathbf{N}\}$ of operations, which are functions from $\mathbf{Z}^2$ or $\mathbf{Z}$ into $\mathbf{Z}$. Unary operations apply to the contents of the accumulator and binary operations apply to the contents of the accumulator and of another cell in local or global memory, specified by a direct or indirect address. (The lower bounds would change only by a constant factor if operations of any—bounded—arity were allowed.) At the beginning of the computation, the input $x_i \in \mathbf{Z}$ is given in the common memory cell c_i, for $i = 1, ..., n$. Starting from this initial configuration, the processors compute synchronously in parallel, each one executing one instruction per step. Read- and write-conflicts may occur during the computation, i.e., it may happen that several processors try to read from or write into the same common cell at the same step. Here we consider only the PRIORITY-PRAM, where this

situation is dealt with as follows: concurrent reading is permitted; if several processors try to write into the same common cell at the same time, that one with the lowest number succeeds. The output of a computation is the content of common cell c_0 at the end of the computation. We say that the PRAM recognizes a language $L \subseteq \mathbf{Z}^n$ for inputs from $I \subseteq \mathbf{Z}^n$ in t steps if on input $\bar{x} \in I$ the content of c_0 after t steps is 0 if $\bar{x} \notin L$ and is 1 if $\bar{x} \in L$.

Several lower bounds have been established so far for this model or variations of it. In [13] a lower bound for a version of the KNAPSACK problem and a lower bound of $\log n + 1$ for the function $f(\bar{x}) = x_1 + \cdots + x_n$ are proved. [2] also proves an $\Omega(\log n)$ lower bound for addition of n n-bit numbers on a PRAM with unrestricted instruction set and polynomially many processors. The other lower bounds for this kind of machine either apply to functions with large ranges (see, e.g., [9 or 11]), to machines with bounded common memory (see, e.g., [9, 20]), or they lie strictly below $\Omega(\log n)$ (e.g., in [3] an $\Omega(\log n/\log\log n)$ lower bound for PARITY on PRAMs with polynomially in n many processors).

Here we prove a tight lower bound of $\Omega(\log n)$ for a decision problem in $\mathbf{Z}^n$ (with relatively small inputs) on PRAMs with infinite common memory and with a very weak restriction on the number of processors. Only the instruction set must be restricted. (Note that some restriction is necessary: one cannot prove an $\Omega(\log n)$ lower bound for problems with inputs $(x_1, ..., x_n)$, where the input numbers x_i have polynomially in n many bits, on PRAMs with exponentially many processors and arbitrary instructions, see [2].) The first step of the proof is a lemma: with a computation time of $t < \log n$ parallel steps a PRAM can compute only functions that can be expressed by a definition of cases in terms of functions which depend on $\leqslant 2^t < n$ variables. This is then used to show that we can fool a machine with too short a computation time, using the fact that the answer to the question "is $x_1 + \cdots + x_{n-1} \leqslant x_n$?" depends *arithmetically* on all n variables.

Using the same lemma, we also show that the lower bound of $\Omega(\log n)$ remains valid even if the operation set of the PRAM consists of arbitrary functions of bounded arity (which makes the machine as powerful as that in [2]). In the proof, Gallai's theorem from Ramsey theory (see [10]) is applied in a similar way as in [5]. As is usually the case with Ramsey theoretic arguments, the input numbers used to prove that the machine makes mistakes might have to be very large.

Note here that this result pinpoints a difference in the behavior of two types of problems on PRAMs with very many processors: problems that depend only on the order type of the input $(x_1, ..., x_n)$ (like sorting or ELEMENT DISTINCTNESS) and "linear" problems (like our threshold problem or KNAPSACK). A PRAM with $O(2^{n^{\log n}})$ processors can sort in constant time, but no bounded (in n) number of processors suffices to solve the threshold problem in constant time.

In Section 3, we turn to lower bounds for linear decision trees. A linear decision tree (LDT; often also called LSA: linear search algorithm) for n input variables is a rooted ternary tree in which each internal node is labelled by a certain linear test $\sum_{i=1}^{n} \alpha_i x_i : \alpha_0$, with $\alpha_0, \alpha_1, ..., \alpha_n \in \mathbf{R}$. The edges from such a node to its three sons are labelled by $<$, $=$, and $>$, respectively. Each leaf is labelled "accept" or "reject."

Inputs for such LDTs have the form $(x_1, ..., x_n)$, consisting of n numbers x_i from $\mathbf{N}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{Q}_+$, or $\mathbf{R}_+$, depending on the context. ($\mathbf{Q}_+$ ($\mathbf{R}_+$) is the set of nonnegative rational (real) numbers.) Each such input defines a path in the tree in the obvious way: start at the root; at an internal node with label $\sum_{i=1}^n \alpha_i x_i : \alpha_0$ follow the outgoing edge labelled $<$, $=$, or $>$, according as the input satisfies $\sum_{i=1}^n \alpha_i x_i < \alpha_0$, $\sum_{i=1}^n \alpha_i x_i = \alpha_0$, or $\sum_{i=1}^n \alpha_i x_i > \alpha_0$. The LDT is said to accept a language $L \subseteq \mathbf{R}^n$ if for all $\bar{x} = (x_1, ..., x_n) \in \mathbf{R}^n$ the path determined by $\bar{x}$ ends at an "accepting" leaf if and only if $\bar{x} \in L$ (similarly for $L \subseteq \mathbf{N}^n$, etc.). As a complexity measure serves the depth of the LDT, which corresponds to the maximum number of tests needed for some input.

Most lower bounds on the depth of LDTs T for decision problems P are "connectivity arguments" (see [4, 7]), where one exploits that, for each leaf l of T, the set of all inputs $(x_1, ..., x_n) \in \mathbf{R}_+^n$ that lead to leaf l forms a connected subset of $\mathbf{R}^n$ (it is an intersection of halfspaces and hyperplanes). Therefore the number of leaves of T must be at least as large as the number of connected components of the considered problem P (resp. its complement $\mathbf{R}^n - P$). Unfortunately, KNAPSACK and the other common NP-complete "number problems" have only $2^{O(n^2)}$ many connected components (see [7, 15]), and the best lower bound we can get in this way is quadratic in n. The (simplified) version of the KNAPSACK problem considered in the cited literature (which we will study in this paper) is defined by

$$\text{KNAPSACK} = \bigcup_{n \in \mathbf{N}} K(n),$$

where

$$K(n) = \left\{ (x_1, ..., x_n) \in \mathbf{R}_+^n \mid \exists S \subseteq \{1, ..., n\} \left(\sum_{i \in S} x_i = 1 \right) \right\}.$$

Actually one usually focuses instead on the discrete version of KNAPSACK, where $K(n)$ is restricted to $\mathbf{Q}_+^n$. This version of the problem is NP-complete.

In order to achieve a larger than quadratic lower bound for KNAPSACK one has to undertake a finer analysis of the mathematical structure of this problem. Dobkin and Lipton [8] and Ukkonen [19] made some progress in this direction: they exploited a geometrical property of the KNAPSACK problem in order to prove an exponential lower bound for KNAPSACK on a very restricted class of LDTs (only linear tests $\sum_{i=1}^n \alpha_i x_i : \alpha_0$ with $\alpha_i \in \{0, 1\}$ are allowed). Unfortunately, their restriction is so severe that one is not even able to sort the n input numbers $(x_1, ..., x_n)$ on an LDT of this type. This entails that on such a model one gets exponential lower bounds for a variety of problems that are in fact computationally trivial but require comparing the size of some of the input numbers x_i (e.g., for the problem of deciding whether $\sum_{i \in S} x_i \geq 1$ for some set $S \subseteq \{1, ..., n\}$ of size $n/2$).

In this paper we use combinatorial and geometrical arguments in order to

achieve a superpolynomial lower bound for KNAPSACK on a more general class of LDTs. Inaccessibles are used here in a slightly more complicated way than in the context of Section 2, since here also some equalities between certain sums of some input numbers have to be satisfied. We consider LDTs where the coefficients α_i in the linear tests $\sum_{i=1}^{n} \alpha_i x_i : \alpha_0$ may be arbitrary real numbers. We show in Theorem 2 that if $f(n) > |\{i | \alpha_i < 0\}|$ for all linear tests $\sum_{i=1}^{n} \alpha_i x_i : \alpha_0$ in the tree, then the depth of the tree is at least $2^{\lfloor n/2f(n) \rfloor}$. This implies a superpolynomial lower bound for KNAPSACK on LDTs where the coefficients α_i in each linear test $\sum_{i=1}^{n} \alpha_i x_i : \alpha_0$ may be arbitrary real numbers provided that $|\{i | \alpha_i < 0\}| = o(n/\log n)$ (it is known that this restriction on the number of negative coefficients can *not* be totally eliminated: without this restriction the *upper* bound on the depth of LDTs for KNAPSACK and other problems like TRAVELING SALESPERSON is known to be polynomial in n, see [14, 16]). Note that linear tests with $o(n/\log n)$ negative coefficients allow not only to sort the x_i (for a comparison of two input numbers x_i, x_j one only needs a single negative coefficient in the respective linear test), but also to sort *sums* $x_i + x_j$, $x_i + x_j + x_k$, ..., where up to $o(n/\log n)$ many of the input numbers x_i occur in a term.

The technique that we use in the proof of Theorem 2 allows us to show for a variety of graph problems that they *inherently* require comparing sums of many input numbers. For example, Theorem 3 exhibits an intrinsic difference between the computation of a minimal spanning tree (where the weights of the edges have to be compared, but no sums of several edge weights need be compared) and the decision problem associated with the shortest path problem, respectively the maximum weight matching problem, for which all familiar algorithms involve the comparison of sums of many edge weights. Theorem 3 shows that in fact there exist no polynomial time algorithms (based on linear tests) for the latter two problems where only sums of up to $o(\sqrt{n}/\log n)$ many weights are compared. The argument of the proof of Theorem 3 yields a number of refinements of this negative result: One can show that even algorithms that are only required to handle particularly "nice" types of problem instances in polynomial time (e.g., only graphs that are planar, or where the weights are given by the Euclidean distance of points in the plane) are forced to compare sums of many edge weights.

These results for problems whose discrete versions are polynomial time computable indicate that the lower bound for KNAPSACK of Section 3 may not be caused by the fact that this problem is NP-complete. It would be desirable to find further evidence that KNAPSACK is hard for LDTs by exploiting features of this problem that are not shared by problems in P.

Finally we would like to point out that "inaccessible numbers" can also be used to prove optimal $\Omega(n \log n)$ lower bounds on the computation time for ELEMENT DISTINCTNESS, DISJOINT SETS, and other decision problems on random access machines with polynomially in n (n = the number of input words) many registers. This application has been discussed in Section 4 of the preliminary version of this paper [6] (detailed proofs will appear in a separate paper [12]).

2. A Lower Bound for a Number Problem on a PRAM

In this section, we prove a lower bound of $\Omega(\log n)$ for the language $\{\bar{x} \in \mathbf{Z}^n \mid x_1 + \cdots + x_{n-1} \leqslant x_n\}$ on a PRAM. (For the definition of this model see Section 1.) As a preliminary, we define the set of "*t-step functions*," for $t \geqslant 0$. This set includes the functions computable in t steps on an "oblivious" PRAM with operation set $\{\mathrm{Op}_1, \mathrm{Op}_2, \ldots\}$, i.e., a PRAM where the instructions executed and addresses used in each step do not depend on the input.

Definition. For functions $f: \mathbf{Z}^n \to \mathbf{Z}$ define by induction on t:

$t = 0$. f is a 0-step function if $f \equiv 0$ or $f \equiv 1$ or $f(\bar{x}) = x_i$ for some i, $1 \leqslant i \leqslant n$.

$t > 0$. f is a t-step function if $f(\bar{x}) = \mathrm{Op}_i(g(\bar{x}), h(\bar{x}))$ or $f(\bar{x}) = \mathrm{Op}_i(g(\bar{x}))$ or $f = g$ for some $(t-1)$-step functions g and h and some unary or binary operation Op_i.

Note that a t-step function depends on $\leqslant 2^t$ of the input variables, as is easily seen by induction on t. Functions computable in t steps on an arbitrary PRAM can be expressed in terms of t-step functions:

Lemma 2.1. *Suppose $I \subseteq \mathbf{Z}^n$ and $f: I \to \mathbf{Z}$ is a function that can be computed in t steps by a* PRAM *M with instruction set $\{\mathrm{Op}_1, \mathrm{Op}_2, \ldots\}$ (finite or infinite). Then f can be expressed by a definition by cases*

$$f(\bar{x}) = \begin{cases} f_1(\bar{x}), & \text{if } \bar{x} \in R_1, \\ \vdots & \vdots \\ f_N(\bar{x}), & \text{if } \bar{x} \in R_N, \end{cases}$$

for $\bar{x} \in I$, where the f_i are t-step functions and the R_i are Boolean combinations of sets of the form $\{\bar{x} \in \mathbf{Z}^n \mid g(x) \geqslant 0\}$ or $\{\bar{x} \in \mathbf{Z}^n \mid g(\bar{x}) = h(\bar{x})\}$ for certain $(t-1)$-step functions g and h.

Remark. In particular, if L is a language and M recognizes L for inputs from I, then $L \cap I$ can be expressed as $R \cap I$, where R is a Boolean combination of sets of the form $\{\bar{x} \mid g(\bar{x}) \geqslant 0\}$, $\{\bar{x} \mid g(\bar{x}) = h(\bar{x})\}$ (g, h $(t-1)$-step functions), and $\{\bar{x} \mid g(\bar{x}) = 1\}$ (g a t-step function).

Proof. To every input $\bar{x}$ and every $t \geqslant 0$ we associate the "*computation pattern of $\bar{x}$ up to step t*," which records

(1) for all processors p' and all steps $t' \leqslant t$, the instruction executed by p' at step t'

(2) the "flow of data" up to step t: if the instruction executed by processor p' at step $t' \leqslant t$ causes a "reading access" to a memory cell m (i.e., the contents of m are loaded, or used as operand, or used as indirect address), then we associate the pair (p'', t'') with this memory access, where t'' is the last time before t' at which

some processor wrote into m, and p'' was the unique processor to do so. If no processor ever wrote into m before t', we record the pair $(0, i)$ if m is the common cell c_i (then m contains x_i at step t'), $1 \leqslant i \leqslant n$, $(0, 0)$ if m is any other cell (then m contains 0).

An important property to notice is that the indirect addresses themselves do not occur in the computation pattern. To prove the lemma, it suffices to show the following:

(*) for inputs of a fixed computation pattern (up to t), the content of the accumulator of each processor after step $t' \leqslant t$ can be expressed as a t'-step function of $x_1, ..., x_n$.

(**) for every fixed computation pattern (up to t), the set of inputs $\bar{x}$ that have this pattern can be expressed as a Boolean combination of sets $\{\bar{x} \mid g(\bar{x}) \geqslant 0\}$ or $\{\bar{x} \mid g(\bar{x}) = h(\bar{x})\}$ for g and h certain $(t-1)$-step functions.

Property (*) can be proved easily by induction on t'. The only interesting case in the induction step is when a processor p' executes a binary operation Op_i at step t' (for all inputs of a fixed computation pattern). The first operand is the content of the accumulator of p' before step t', the second operand is the content of the accumulator of p'' before step t'', where (p'', t'') is the pair associated with this instruction in the computation pattern. Both these register contents are $(t-1)$-step functions by the induction hypothesis.

Property (**) is then proved by induction on t. The initial step is clear. Suppose (*) and (**) are true for $t-1$. Let R be the set of all inputs of a particular computation pattern up to step $t-1$. By induction, R can be written as described in (**). R splits into subsets according to the different behavior of the PRAM in step t on inputs $\bar{x} \in R$. This behavior in step t for inputs $\bar{x}$ and $\bar{x}'$ from R may be different for two reasons:

—different instructions are executed by a processor p

—the tag (p'', t'') associated with the instruction executed at step t may be different.

The first distinction can be made by sets of the form $\{\bar{x} \mid g(\bar{x}) \geqslant 0\}$: the instruction executed by p in step $t-1$ was a test ("if $r_0 \geqslant 0$ then ..."), and by the induction hypothesis for (*) there is a $(t-1)$-step function g that gives the content of the accumulator r_0 of p after step $t-1$, for all $\bar{x} \in R$. As for the second case (different tags (p'', t'')), we have to define the set of all $\bar{x} \in R$ with a specific tag (p'', t'') as a Boolean combination of sets as described in (**). But all we have to express here is that the addresses used by p in step t and by p'' in step t'' are the same, and these addresses are either constant or are register contents of fixed (for inputs in R) other processors before steps t'' resp. t. This can be written as "$g(\bar{x}) = h(\bar{x})$" for certain $(t-1)$-step functions g and h (by (*)). Additionally, we must express that no other

processor used this address to write to between steps t'' and t. This can be done by inequalities "$g(\bar{x}) \neq h(\bar{x})$" for those $(t-1)$-step functions g and h that give the appropriate register contents. ▮

Combining the lemma with the proper choice of an input now yields the desired lower bound.

THEOREM 1. *Let M be a PRAM with instruction set $\{+, -,$ multiplication by constants$\}$, and suppose that the programs of the processors involved in the computations for inputs $(x_1, ..., x_n)$ have bitlength bounded above by $c(n)$. Then M needs $>\log(n-1)$ steps to recognize the "threshold set" $L = \{\bar{x} \in \mathbf{Z}^n \mid x_1 + \cdots + x_{n-1} \leqslant x_n\}$ for inputs from $I = \{0, 1, 2, ..., 2^{c(n)\cdot n \cdot \log n}\}^n$.*

Proof. Suppose for a contradiction that M recognizes L for inputs from I in t steps, where $2^t \leqslant n-1$. Then, by the remark following Lemma 2.1, $L \cap I$ has the form $R \cap I$, where R is a Boolean combination of sets of the form $\{\bar{x} \mid g(\bar{x}) \geqslant 0\}$, $\{\bar{x} \mid g(\bar{x}) = h(\bar{x})\}$, and $\{\bar{x} \mid f(\bar{x}) = 1\}$ for $(t-1)$-step functions g and h, and t-step functions f. An easy induction on t shows that t-step functions over the operation set $\{+, -,$ multiplication by constants $\leqslant 2^{c(n)}\}$ have the form $\alpha_0 + \sum_{i=1}^n \alpha_i x_i$ for certain integer coefficients α_i with $|\alpha_i| \leqslant 2^{c(n)\cdot t}$. Hence R can be written as a Boolean combination of sets of the form $S = \{\bar{x} \mid \alpha_0 + \sum_{i=1}^n \alpha_i x_i \geqslant 0\}$, where $|\alpha_i| \leqslant 2^{c(n)\cdot t}$ and not more than $2^t \leqslant n-1$ many of the α_i are $\neq 0$. Using "inaccessibles," we can now easily exhibit two inputs $\bar{a}, \bar{a}' \in I$, one in L, the other not in L, that behave in the same way w.r.t. all such sets S, hence cannot be told apart by M. (This is the desired contradiction.) Define a large integer b (the "basis") by $b := 2^{c(n)\cdot t+3}$. Then the powers of b are "inaccessible" for M in the following sense: if $d = \alpha_0 + \sum_{i=1}^{n-1} \alpha_i b^i$ is a linear combination of the b^i that M can compute in t steps, i.e., where $|\alpha_i| \leqslant 2^{c(n)\cdot t}$ for all i, then d is 0 if and only if all the α_i are 0; further, if one of the α_i, $i \geqslant 1$, is not 0, and i_0 is the maximal i with $\alpha_i \neq 0$, then the sign of d equals the sign of α_{i_0}. The last property is even "stable" in the sense that if α_0 is changed slightly (by less than b), then the sign of d does not change. (The b^i, $i \geqslant 1$, are of a larger "order of magnitude" than α_0, as far as M can see.) These properties are used here in a very simple manner: Consider inputs $\bar{a} = (a_1, ..., a_n)$ and $\bar{a}' = (a_1', ..., a_n')$, where

$$a_i = a_i' = b^i, \qquad \text{for} \quad i = 1, ..., n-1,$$

and

$$a_n = a_1 + \cdots + a_{n-1}, \qquad a_n' = a_1 + \cdots + a_{n-1} - 1.$$

(Clearly, $\bar{a}, \bar{a}' \in I$.) How do $\bar{a}$ and $\bar{a}'$ behave w.r.t. a set $S = \{\bar{x} \mid \alpha_0 + \sum_{i=1}^n \alpha_i x_i \geqslant 0\}$, where $|\alpha_i| \leqslant 2^{c(n)\cdot t}$ and $\leqslant n-1$ of the α_i are nonzero? We have $\bar{a} \in S$ if and only if $\bar{a}' \in S$, as can be seen by examining two cases:

—if $\alpha_n = 0$, then $\alpha_0 + \sum_{i=1}^n \alpha_i a_i = \alpha_0 + \sum_{i=1}^n \alpha_i a_i'$.

—if $\alpha_n \neq 0$, then one of the α_i, $1 \leqslant i < n$, must be 0. Let i_0 be the maximal i with $\alpha_i + \alpha_n \neq 0$. Then the sign of $\alpha_{i_0} + \alpha_n$ is the same as the sign of both

$$\alpha_0 + \sum_{i=1}^{n} \alpha_i a_i = \alpha_0 + \sum_{i=1}^{n-1} (\alpha_i + \alpha_n) b^i,$$

$$\alpha_0 + \sum_{i=1}^{n} \alpha_i a_i' = (\alpha_0 - \alpha_n) + \sum_{i=1}^{n-1} (\alpha_i + \alpha_n) b^i,$$

by the observations about the powers of b we just made.

Since R can be described in terms of such sets S, we conclude that $\bar{a} \in R$ if and only if $\bar{a}' \in R$, which shows that M cannot distinguish between $\bar{a}$ and $\bar{a}'$. This contradicts the fact that $\bar{a} \in L$ and $\bar{a}' \notin L$. ▮

Remark. (a) The lower bound just proved is, of course, tight. A PRAM with n processors can decide in $\log(n-1)+3$ steps whether $x_1 + \cdots + x_{n-1} \leqslant x_n$ or not.

(b) In an analogous way we can obtain a lower bound for PRAMs with linear instruction set for SHORTEST PATH. Here we mean the following version of the SHORTEST PATH problem: Let m be such that $\frac{1}{2}m(m-1) = n-1$, and fix a bijection between the variables $x_1, ..., x_{n-1}$ and the edges $e_1, ..., e_{n-1}$ of K_m, the complete graph on vertices $v_1, ..., v_m$. Then SHORTEST PATH is the problem to decide for an input $\bar{x} = (x_1, ..., x_n) \in \mathbf{N}^n$ whether there is a path in K_m from v_1 to v_2 of total weight $\leqslant x_n$, the weight of edge e_i being given by x_i. The proof of Theorem 1 is adapted in the following way: We choose edges $e_{i_1}, ..., e_{i_{m-1}}$ that form a path from v_1 to v_2, and let $x_{i_1}, ..., x_{i_{m-1}}, x_n$ play the role that $x_1, ..., x_{n-1}, x_n$ had in the previous proof. Only inputs are considered where all other variables are set equal to $2 \cdot b^{m+2}$. The lower bound obtained in this way is $\lfloor \log(m-1) \rfloor$, which is approximately $\frac{1}{2} \log n$. (The lowest known upper bound for this problem on PRAMs with linear instruction set is $O(\log^2 n)$.)

(c) The proof of Theorem 1 also yields a lower bound of $\Omega(\log n)$ for the following language, which is a version of the KNAPSACK problem:

$$L = \left\{ \bar{x} \in \mathbf{N}^n \mid \exists S \subseteq \{1, ..., n-1\} : \sum_{i \in S} x_i = x_n \right\},$$

on PRAMs with linear instruction set and arbitrarily many processors.

Remark. The lower bound of Theorem 1 also applies to PRAMs with arbitrary operations of bounded arity, if one admits very large input numbers: if M is a PRAM with operations $\mathrm{Op}_1, \mathrm{Op}_2, ...$ (arbitrary unary and binary functions), and if for inputs from $\mathbf{Z}^n$ not more than $P(n)$ processors are active in the computation, then M needs $> \log(n-2)$ steps to recognize $L = \{\bar{x} \in \mathbf{Z}^n \mid x_1 + \cdots + x_{n-1} \leqslant x_n\}$. (The same is true for the KNAPSACK problem of the previous remark.) The proof is as follows: Suppose for a contradiction that M recognizes L in $\leqslant \log(n-2)$ steps.

By Lemma 2.1, L can be written as a Boolean combination of finitely many sets of the form $\{\bar{x} \in \mathbf{Z}^n \mid Q(\bar{x})\}$, where Q is a predicate that depends on not more than $n-2$ of the variables $x_1, ..., x_n$. Hence the set $L' = \{\bar{x} \in \mathbf{N}^n \mid x_1 + \cdots + x_{n-1} = x_n\}$ is a Boolean combination of sets $\{\bar{x} \in \mathbf{N}^n \mid Q_j(\bar{x})\}$, $j = 1, ..., N$, where Q_j is independent of x_{i_j}, for some $i_j \leqslant n-1$. We use Gallai's theorem from Ramsey theory [10] to show that this is impossible, in a manner similar to the way it is used in [5] to prove a lower bound for "multi-party protocols." Define the following coloring on $\mathbf{N}^{n-1}$:

$$\chi(x_1, ..., x_{n-1}) = (d_1, ..., d_N),$$

where

$$d_j = \begin{cases} 1, & \text{if } Q_j(x_1, ..., x_{n-1}, x_1 + \cdots + x_{n-1}), \\ 0, & \text{otherwise,} \end{cases} \qquad \text{for } j = 1, ..., N.$$

By the version of Gallai's theorem for natural numbers [10, p. 38], there is a monochromatic set homothetic to $\{e_1, ..., e_{n-1}\}$ (the unit vectors in $\mathbf{N}^{n-1}$). I.e., there are $b_1, ..., b_{n-1} \in \mathbf{N}$ and some $c \in \mathbf{N} - \{0\}$ such that $\chi(b_1, ..., b_{i-1}, b_i + c, b_{i+1}, ..., b_{n-1})$ does not depend on i; that means, it does not depend on i whether $Q_j(\bar{b}_i)$ or not, for $\bar{b}_i = (b_1, ..., b_{i-1}, b_i + c, b_{i+1}, ..., b_{n-1}, b_1 + \cdots + b_{n-1} + c)$. Now consider $\bar{b} := (b_1, ..., b_{n-1}, b_1 + \cdots + b_{n-1} + c)$. Clearly $\bar{b} \notin L'$. For every $j \leqslant N$, choose some $i_j \leqslant n-1$ such that Q_j does not depend on x_{i_j}. Since $\bar{b}$ and $\bar{b}_{i_j}$ differ in just the i_jth component, we have, for all j,

$$Q_j(\bar{b}) \Leftrightarrow Q_j(\bar{b}_{i_j}) \Leftrightarrow Q_j(\bar{b}_i) \qquad \text{for all } i.$$

Hence, $\bar{b}$ behaves in the same way as the vectors $\bar{b}_i \in L'$ w.r.t. all the predicates Q_j. Since L' can be expressed in terms of the Q_j, we conclude that $\bar{b} \in L'$, a contradiction.

3. A Lower Bound for KNAPSACK on Linear Decision Trees

Theorem 2. *Let T_n be a linear decision tree for inputs $\bar{x} \in \mathbf{R}^n$, for all $n \in \mathbf{N}$, and let $f: \mathbf{N} \to \mathbf{N}$ be a function such that every test $\sum_{i=1}^n \alpha_i x_i : \alpha_0$ in T_n ($\alpha_i \in \mathbf{R}$; possible outcomes: $<$, $=$, $>$) satisfies $|\{i \geqslant 1 \mid \alpha_i < 0\}| < f(n)$. If T_n recognizes the* KNAPSACK *problem*

$$K(n) := \left\{ \bar{x} \in \mathbf{R}_+^n \mid \exists S \subseteq \{1, ..., n\} \left(\sum_{i \in S} x_i = 1 \right) \right\},$$

then $\operatorname{depth}(T_n) \geqslant 2^{\lfloor n/2f(n) \rfloor}$ *for all* $n \in \mathbf{N}$.

Note. This lower bound is superpolynomial if $f(n) = o(n/\log n)$.

Remark. It will be seem from the proof that it suffices to assume that T_n finds the correct answer for inputs $\bar{x} \in \mathbf{Q}_+^n$.

Proof of Theorem 2. Fix n and set $k := f(n)$ and $p := n/2k$. We show that $\text{depth}(T_n) \geqslant 2^p$.

Note here that we can assume w.l.o.g. that $2k$ divides n. If this is not the case, let $n_0 := 2k \cdot \lfloor n/2k \rfloor$, and consider the LDT T' obtained from T_n by replacing all tests $\sum_{i=1}^n \alpha_i x_i : \alpha_0$ by $\sum_{i=1}^{n_0} \alpha_i x_i : \alpha_0$. Then it is clear that T' recognizes $K(n_0)$ and that $|\{i \geqslant 1 \mid i \leqslant n_0 \text{ and } \alpha_i < 0\}| < k$ for all tests in T'. We have $2k \mid n_0$, hence, by the special case, $\text{depth}(T') \geqslant 2^{n_0/2k}$. Since $\text{depth}(T') = \text{depth}(T_n)$ and $\lfloor n/2k \rfloor = n_0/2k$, it follows that $\text{depth}(T_n) \geqslant 2^{\lfloor n/2k \rfloor}$.

We shall define a point $\bar{a} \in \mathbf{Q}_+^n - K(n)$, and distinct points $\bar{a}_I \in K(n)$, and distinct sets $S(I) \subseteq \{1, ..., n\}$, for $I \subseteq \{1, ..., p\}$, such that the only "knapsack hyperplane" $\{\bar{x} \mid \sum_{i \in S} x_i = 1\}$ (for some $S \subseteq \{1, ..., n\}$) on which $\bar{a}_I$ lies is $K_I := \{\bar{x} \mid \sum_{i \in S(I)} x_i = 1\}$. Since T_n gives different outputs for $\bar{a}$ and $\bar{a}_I$, there is for each $I \subseteq \{1, ..., p\}$ a test $\sum_{i=1}^n \alpha_i x_i : \alpha_0$ on the path in T_n taken by $\bar{a}$ such that the corresponding "test hyperplane" $\{\bar{x} \mid \sum_{i=1}^n \alpha_i x_i = \alpha_0\}$ intersects L_I, the closed line segment starting at $\bar{a}$ and ending at $\bar{a}_I$. The choice of $\bar{a}$ and $\bar{a}_I$ will ensure that the only "test hyperplane" which intersects L_I, if any, is K_I itself. This implies that at least the 2^p tests corresponding to the knapsack hyperplanes K_I are executed along the computation path for $\bar{a}$. (This is the desired lower bound.) The analogous task was quite easy in the models of [8, 19], since there the only "test hyperplanes" that were allowed in T_n were just the knapsack hyperplanes (therefore in those models one could choose the components of $\bar{a}$ to be equal). In our case the definition of $\bar{a}$ is more involved: its coordinates will satisfy two kinds of "inaccessibility conditions," together with equalities between certain sums of coordinates.

To simplify notation later, we note that w.l.o.g. we can assume the following: if $\delta = \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 \neq 0$, where $\gamma_i \in \{0\} \cup \{\gamma \mid \gamma \text{ or } -\gamma \text{ is a coefficient in } T_n\}$, then $|\delta| \geqslant 1$. (If this is not already the case, multiply all coefficients in T_n by C^{-1}, where $C := \min\{|\delta| \mid \delta \neq 0,\ \delta = \sum_{i=1}^4 \gamma_i \text{ for } \gamma_i \in \{0\} \cup \{\gamma \mid \gamma \text{ or } -\gamma \text{ is a coefficient in } T_n\}\}$.) This assumption allows us to prove the following lemma, which is the first step towards the definition of $\bar{a}$.

LEMMA 3.1. *There exists a number $b \in \mathbf{N}$ such that for all $m \in \mathbf{N}$, all $\delta_i = \gamma_{i1} + \gamma_{i2} + \gamma_{i3} + \gamma_{i4}$, where $\gamma_{ij} \in \{0\} \cup \{\gamma \mid \gamma$ or $-\gamma$ is a coefficient in $T_n\}$ $(i = 1, ..., m,$ $j = 1, ..., 4)$ and all ε with $|\varepsilon| \leqslant b^{-m-1}$: if $\sum_{i=1}^m \delta_i b^{-i} + \varepsilon = 0$, then $\delta_1 = \cdots = \delta_m = \varepsilon = 0$. (The powers of b are mutually "inaccessible" w.r.t. linear combinations with coefficients from T_n.)*

Proof. Choose $b \in \mathbf{N}$ so that $b > \max(5D, n+1)$, where $D := \max\{|\gamma| \mid \gamma \text{ is a coefficient in } T_n\} \geqslant 1$. Let m be arbitrary, and assume that ε is such that $|\varepsilon| \leqslant b^{-m-1}$, and that $\delta_1, ..., \delta_m$ are of the indicated form. Assume that $\sum_{i=1}^m \delta_i b^{-i} + \varepsilon = 0$. We show that $\delta_1 = 0$. (Then the lemma follows by induction on m.) Suppose for a contradiction that $\delta_1 \neq 0$. By the assumption preceding the

lemma we even have $|\delta_1| \geq 1$. Hence $b^{-1} \leq |\delta_1 b^{-1}| = |\sum_{i=2}^{m} \delta_i b^{-i} + \varepsilon| \leq \sum_{i=2}^{m+1} 4Db^{-i} \leq \frac{4}{5} \sum_{i=2}^{m+1} b \cdot b^{-i} < \frac{4}{5} \cdot b^{-1}/(1-b^{-1}) < b^{-1}$, a contradiction. ∎

For the rest of the proof, we fix some $b \in \mathbf{N}$ as in Lemma 3.1. Now we define a scaling factor B by

$$B := \sum_{i=1}^{p} b^{-i},$$

and choose $\delta > 0$ so small that $2p\delta < b^{-p-(2p+3)k-2}$, $\delta \in \mathbf{Q}$. For $1 \leq i \leq p$, let

$$a_i := b_i := \frac{1}{B} \cdot b^{-i} + \delta.$$

Note that the point $(1/B)(b^{-1}, b^{-2}, ..., b^{-p}, b^{-1}, b^{-2}, ..., b^{-p})$ lies on the 2^p hyperplanes $\{(y_1, ..., y_p, z_1, ..., z_p) \in \mathbf{R}^{2p} \mid \sum_{i \in I} y_i + \sum_{i \notin I} z_i = 1\}$, for $I \subseteq \{1, ..., p\}$. It turns out that the point $(a_1, ..., a_p, b_1, ..., b_p)$ lies strictly inside a polytope which has all these 2^p hyperplanes as supporting hyperplanes. This arrangement already allows us to prove the lower bound for linear decision trees with arbitrary nonnegative coefficients: For each $I \subseteq \{1, ..., p\}$ we consider a vector $(a_1^I, ..., a_p^I, b_1^I, ..., b_p^I)$, where

$$a_i^I := \begin{cases} a_i - \delta = \dfrac{1}{B} \cdot b^{-i}, & \text{if } i \in I \\ a_i, & \text{if } i \notin I, \end{cases} \qquad b_i^I := \begin{cases} b_i, & \text{if } i \in I \\ b_i - \delta = \dfrac{1}{B} \cdot b^{-i}, & \text{if } i \notin I. \end{cases}$$

Obviously, $\sum_{i \in I} a_i^I + \sum_{i \notin I} b_i^I = 1$. In the next lemma we show that there is at most one hyperplane in $\mathbf{R}^{2p}$ definable with nonnegative coefficients from T_n which makes a difference between $(a_1, ..., a_p, b_1, ..., b_p)$ and $(a_1^I, ..., a_p^I, b_1^I, ..., b_p^I)$ in the sense that the two points do not both lie on the hyperplane or in the same of the two open halfspaces defined by it. This hyperplane is

$$\left\{ (y_1, ..., y_p, z_1, ..., z_p) \in \mathbf{R}^{2p} \,\middle|\, \sum_{i \in I} y_i + \sum_{i \notin I} z_i = 1 \right\}.$$

LEMMA 3.2 (Use of inaccessibility of the "first kind"). *Let $0 \leq \eta \leq \delta$. Let for $1 \leq i \leq p$, $\alpha_i, \beta_i \in \{\gamma \mid \gamma$ or $-\gamma$ is a coefficient in $T_n\}$, $\alpha_i, \beta_i \geq 0$, but not all α_i, β_i equal 0. Let $\gamma \in \mathbf{R}$ be such that γ or $-\gamma$ is a coefficient in T_n. Finally, let $I \subseteq \{1, ..., p\}$ be arbitrary. Assume that for*

$$y_i^0 := \begin{cases} a_i - \eta, & \text{if } i \in I \\ a_i, & \text{if } i \notin I, \end{cases} \qquad z_i^0 := \begin{cases} b_i, & \text{if } i \in I \\ b_i - \eta, & \text{if } i \notin I \end{cases}$$

holds

$$\sum_{i=1}^{p} \alpha_i y_i^0 + \sum_{i=1}^{p} \beta_i z_i^0 = \gamma.$$

Then $\eta = \delta$, *and* $\forall i \in I$: $\alpha_i = \gamma \wedge \beta_i = 0$, *and* $\forall i \notin I$: $\alpha_i = 0 \wedge \beta_i = \gamma$, *i.e., the hyperplane* $\{(y_1, ..., y_p, z_1, ..., z_p) \in \mathbf{R}^{2p} \mid \sum_{i=1}^{p} \alpha_i y_i + \sum_{i=1}^{p} \beta_i z_i = \gamma\}$ *equals* $\{(y_1, ..., y_p, z_1, ..., z_p) \in \mathbf{R}^{2p} \mid \sum_{i \in I} y_i + \sum_{i \notin I} z_i = 1\}$.

Proof. The assumption $\sum_{i=1}^{p} \alpha_i y_i^0 + \sum_{i=1}^{p} \beta_i z_i^0 = \gamma$ means, by the definitions,

$$\sum_{i=1}^{p} \alpha_i \left(\frac{1}{B} \cdot b^{-i} + \delta\right) - \sum_{i \in I} \alpha_i \eta + \sum_{i=1}^{p} \beta_i \left(\frac{1}{B} b^{-i} + \delta\right) - \sum_{i \notin I} \beta_i \eta = \gamma.$$

Multiplying by $B = \sum_{i=1}^{p} b^{-i}$ and collecting summands with the same power of b yields

$$\sum_{i=1}^{p} (\alpha_i + \beta_i - \gamma) b^{-i} + B \cdot \left(\sum_{i \in I} (\alpha_i(\delta - \eta) + \beta_i \delta) + \sum_{i \notin I} (\alpha_i \delta + \beta_i(\delta - \eta))\right) = 0.$$

Since $0 \leqslant \delta - \eta \leqslant \delta$ and $\alpha_i + \beta_i \leqslant b$, the second summand is

$$\leqslant B \cdot p \cdot \frac{2b}{5} \cdot \delta < 2 \cdot b^{-1} \cdot p \cdot \frac{b}{2} \cdot \delta = p \cdot \delta < b^{-p-1},$$

by choice of b and δ. By Lemma 3.1 we get $\alpha_i + \beta_i - \gamma = 0$, i.e., $\alpha_i + \beta_i = \gamma$, for $1 \leqslant i \leqslant p$, and

$$\sum_{i \in I} (\alpha_i(\delta - \eta) + \beta_i \delta) + \sum_{i \notin I} (\alpha_i \delta + \beta_i(\delta - \eta)) = 0.$$

Since the α_i, β_i are $\geqslant 0$ and are not all $= 0$, the last equality can hold only if

$$\forall i \in I: \beta_i = 0 \text{ (hence } \alpha_i = \gamma), \qquad \forall i \notin I: \alpha_i = 0 \text{ (hence } \beta_i = \gamma), \text{ and } \delta = \eta.$$

Thus the equation $\sum_{i=1}^{p} \alpha_i y_i + \sum_{i=1}^{p} \beta_i z_i = \gamma$ is in fact the same as $\sum_{i \in I} \gamma y_i + \sum_{i \notin I} \gamma z_i = \gamma$. This proves the claim. ∎

An additional effort is needed if the tree T_n uses questions with both positive and negative coefficients. Clearly, tests like "$x_i - x_j : 0$" can distinguish $(a_1, ..., a_p, b_1, ..., b_p)$ from $(a_1^I, ..., a_p^I, b_1^I, ..., b_p^I)$, so Lemma 3.2 does not apply directly any more. To accommodate for negative coefficients, we are forced to use another "level" of inaccessible numbers (inaccessibility of the "second kind"): The numbers a_i and b_i $(i = 1, ..., p)$ are split into k parts each (e.g., $a_i = a_{i1} + \cdots + a_{ik}$) so that all the $2pk$ parts we obtain are mutually "inaccessible" (with regard to the coefficients which occur in T_n). The vector with all these a_{ij}'s and b_{ij}'s as components will be the vector $\bar{a} \in \mathbf{R}^n$, with the properties indicated at the beginning of the proof: $\bar{a}$ does not lie on any knapsack hyperplane in $\mathbf{R}^n$, but for each $I \subseteq \{1, ..., p\}$ one can reach from it on a straight line a knapsack hyperplane K_I without intersecting any "test-hyperplanes" other than K_I.

Notation. For the following, it is convenient to rename the components of vectors $(x_1, ..., x_n) \in \mathbf{R}^n$. They are split into two groups and given double indices:

$$\bar{x} = (x_1, ..., x_n) = (y_{ij}, z_{ij} \mid 1 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant k).$$

We write $(y_{ij}, z_{ij})_{i,j}$ for such vectors in $\mathbf{R}^n$.

DEFINITION. For $i = 1, ..., p$, let

$$a_{ij} := \frac{1}{B} \cdot b^{-p-2ik-j}, \qquad b_{ij} := \frac{1}{B} \cdot b^{-p-(2i+1)k-j} \qquad (2 \leqslant j \leqslant k),$$

$$a_{i1} := \frac{1}{B} \cdot b^{-i} - \sum_{j=2}^{k} a_{ij} + \delta = a_i - \sum_{j=2}^{k} a_{ij}$$

$$b_{i1} := \frac{1}{B} \cdot b^{-i} - \sum_{j=2}^{k} b_{ij} + \delta = b_i - \sum_{j=2}^{k} b_{ij}.$$

$$\bar{a} := (a_{ij}, b_{ij})_{i,j}.$$

For $I \subseteq \{1, ..., p\}$ let

$$K_I := \left\{ (y_{ij}, z_{ij})_{i,j} \,\middle|\, \sum_{i \in I} \sum_{j=1}^{k} y_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} z_{ij} = 1 \right\}.$$

(K_I is a knapsack hyperplane close to $\bar{a}$.) "Characteristic vectors" $\bar{c}_I = (c_{ij}^I, d_{ij}^I)$, defined by

$$c_{ij}^I := \begin{cases} 1, & \text{if } i \in I \text{ and } j = 1 \\ 0, & \text{otherwise,} \end{cases} \qquad d_{ij}^I := \begin{cases} 1, & \text{if } i \notin I \text{ and } j = 1 \\ 0, & \text{otherwise,} \end{cases}$$

are needed to define the line segments L_I from $\bar{a}$ to points $\bar{a}_I \in K_I$:

$$L_I := \{\bar{a} - \eta \bar{c}_I \mid 0 \leqslant \eta \leqslant \delta\}$$

$$\bar{a}_I := \bar{a} - \delta \bar{c}_I.$$

The following three lemmata verify that $\bar{a}$ and the L_I, K_I, $\bar{a}_I$ ($I \subseteq \{1, ..., p\}$) have the desired properties:

—$\bar{a} \notin K(n)$ (Corollary 3.6),

—$\bar{a}_I \in K_I \subseteq K(n)$ (Lemma 3.3),

—if L_I intersects a test hyperplane, then this test hyperplane equals K_I (Corollary 3.6).

As we have argued at the beginning of the proof, these properties together with the obvious fact that the K_I are all different from each other imply that the path in the tree T_n taken by $\bar{a}$ contains $\geqslant 2^p$ tests, which is what we wanted to show.

LEMMA 3.3. *For all* $I \subseteq \{1, ..., p\}$: $\bar{a}_I \in K_I \subseteq K(n)$.

Proof. Straightforward computation, using the facts that $(1/B) \cdot \sum_{i=1}^{p} b^{-i} = 1$,

$$a_{i1} + \sum_{j=2}^{k} a_{ij} - \sum_{j=1}^{k} \delta c_{ij}^I = \frac{1}{B} \cdot b^{-i}, \qquad \text{for} \quad i \in I,$$

and

$$b_{i1} + \sum_{j=2}^{k} b_{ij} - \sum_{j=1}^{k} \delta d_{ij}^I = \frac{1}{B} \cdot b^{-i}, \qquad \text{for} \quad i \notin I. \quad \blacksquare$$

LEMMA 3.4 (Use of inaccessibility of the "second kind"). *Let* $0 \leqslant \eta \leqslant \delta$. *Let* a_{ij}, β_{ij} *be real numbers which occur as coefficients in* T_n, *for* $1 \leqslant i \leqslant p$, $1 \leqslant j \leqslant k$. *Let* $\gamma \in \mathbf{R}$ *be such that* γ *or* $-\gamma$ *is a coefficient in* T_n. *Let* $I \subseteq \{1, ..., n\}$ *be arbitrary. Assume that* $\bar{x} = (y_{ij}, z_{ij})_{i,j} = \bar{a} - \eta \bar{c}_I \in L_I$ *satisfies*

$$\sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} = \gamma. \tag{*}$$

Then $\alpha_{ij} = \alpha_{i1}$ *and* $\beta_{ij} = \beta_{i1}$ *for* $1 \leqslant i \leqslant p$, $2 \leqslant j \leqslant k$.

Proof. We rewrite (*) according to the definitions:

$$\sum_{i,j} \alpha_{ij}(a_{ij} - \eta c_{ij}^I) + \sum_{i,j} \beta_{ij}(b_{ij} - \eta d_{ij}^I) = \gamma,$$

i.e.,

$$\sum_{i=1}^{p} \alpha_{i1} \left(\frac{1}{B} \cdot b^{-i} - \sum_{j=2}^{k} a_{ij} + \delta - \eta c_{i1}^I \right) + \sum_{i=1}^{p} \sum_{j=2}^{k} \alpha_{ij} a_{ij}$$
$$+ \sum_{i=1}^{p} \beta_{i1} \left(\frac{1}{B} \cdot b^{-i} - \sum_{j=2}^{k} b_{ij} + \delta - \eta d_{i1}^I \right) + \sum_{i=1}^{p} \sum_{j=2}^{k} \beta_{ij} b_{ij} = \gamma.$$

Multiply both sides by $B = \sum_{i=1}^{p} b^{-i}$, and recall that $Ba_{ij} = b^{-p-2ik-j}$, $Bb_{ij} = b^{-p-(2i+1)k-j}$, for $1 \leqslant i \leqslant p$, $2 \leqslant j \leqslant k$; then collect summands containing the same power of b:

$$\sum_{i=1}^{p} (\alpha_{i1} + \beta_{i1} - \gamma) b^{-i} + \sum_{i=1}^{p} \sum_{j=2}^{k} (\alpha_{ij} - \alpha_{i1}) b^{-p-2ik-j}$$
$$+ \sum_{i=1}^{p} \sum_{j=2}^{k} (\beta_{ij} - \beta_{i1}) b^{-p-(2i+1)k-j}$$
$$+ B \cdot \left(\sum_{i \in I} (\alpha_{i1}(\delta - \eta) + \beta_{i1}\delta) + \sum_{i \notin I} (\alpha_{i1}\delta + \beta_{i1}(\delta - \eta)) \right) = 0.$$

The absolute value of the last summand is $\leqslant B \cdot p \cdot 2 \cdot (b/5) \cdot \delta < b^{-p-(2p+3)k-1}$ by the choice of δ. We apply Lemma 3.1 to obtain $\alpha_{ij} - \alpha_{i1} = \beta_{ij} - \beta_{i1} = 0$ for $1 \leqslant i \leqslant p$, $2 \leqslant j \leqslant k$. ∎

LEMMA 3.5. *Let η, α_{ij}, β_{ij} $(1 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant k)$, γ, I be as in Lemma* 3.4, *such that the α_{ij}, β_{ij} are not all* 0. *Assume that $\bar{x} = (y_{ij}, z_{ij})_{i,j} = \bar{a} - \eta \bar{c}_I$ satisfies* (∗), *and that*

$$|\{(i,j) \mid \alpha_{ij} < 0\}| + |\{(i,j) \mid \beta_{ij} < 0\}| < k.$$

Then the hyperplane defined by (∗) *equals K_I, and $\eta = \delta$, i.e., $\bar{x} = \bar{a}_I$.*

COROLLARY 3.6. (i) *If $\sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} : \gamma$ is a test in T_n, and L_I has a point in common with $\{(y_{ij}, z_{ij})_{i,j} \mid \sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} = \gamma\}$, then this hyperplane equals K_I.*

(ii) $\bar{a} \notin K(n)$.

Proof of Lemma 3.5. Applying Lemma 3.4 yields $\alpha_{ij} = \alpha_{i1}$, $\beta_{ij} = \beta_{i1}$, for $1 \leqslant i \leqslant p$, $2 \leqslant j \leqslant k$. Hence none of the coefficients can be negative (otherwise $\geqslant k$ of them would be negative, contradicting the assumption). We now collect summands with the same coefficients in (∗) and obtain

$$\sum_{i=1}^{p} \alpha_{i1} \cdot \sum_{j=1}^{k} (a_{ij} - \eta c_{ij}^I) + \sum_{i=1}^{p} \beta_{i1} \cdot \sum_{j=1}^{k} (b_{ij} - \eta d_{ij}^I) = \gamma.$$

By the definition of the a_{ij}, b_{ij}, c_{ij}^I, d_{ij}^I, a_i, b_i, this is the same as

$$\sum_{i=1}^{p} \alpha_{i1}(a_i - \eta c_{i1}^I) + \sum_{i=1}^{p} \beta_{i1}(b_i - \eta d_{i1}^I) = \gamma.$$

To this equation we apply Lemma 3.2, and we get

$$\eta = \delta, \qquad \forall i \in I: \alpha_{i1} = \gamma \wedge \beta_{i1} = 0, \qquad \text{and} \qquad \forall i \notin I: \alpha_{i1} = 0 \wedge \beta_{i1} = \gamma.$$

γ cannot be 0, since some of the α_{ij}, β_{ij} were assumed to be $\neq 0$. By multiplying (∗) by γ^{-1} we finally get that (∗) is equivalent to

$$\sum_{i \in I} \sum_{j=1}^{k} y_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} z_{ij} = 1,$$

which is the equation defining K_I. ∎

Proof of Corollary 3.6. (i) If $\sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} : \gamma$ is a test in T_n, then w.l.o.g. not all coefficients are 0, and the number of negative coefficients among the α_{ij}, β_{ij} is less than k by the assumption we made about T_n. So Lemma 3.5 applies directly.

(ii) Let K be an arbitrary knapsack hyperplane. In our notation, K has the form $\{(y_{ij}, z_{ij})_{i,j} \mid \sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} = 1\}$ for certain $\alpha_{ij}, \beta_{ij} \in \{0, 1\}$, not all 0. Suppose that $\bar{a} - \eta \cdot \bar{c}_I \in K$ for some η, $0 \leqslant \eta \leqslant \delta$, and some $I \subseteq \{1, ..., p\}$. Choose an arbitrary $\gamma > 0$ such that γ or $-\gamma$ is a coefficient in T_n. Then $\bar{x} = (y_{ij}, z_{ij})_{i,j} = \bar{a} - \eta \cdot \bar{c}_I$ satisfies

$$\sum_{i,j} (\alpha_{ij}\gamma)\, y_{ij} + \sum_{i,j} (\beta_{ij}\gamma)\, z_{ij} = \gamma.$$

Applying Lemma 3.5 to this situation yields $\eta = \delta$, i.e., $\bar{x} = \bar{a}_I \neq \bar{a}$. In particular, $\bar{a} \notin K$. This holds for all knapsack hyperplanes K, hence $\bar{a} \notin K(n)$. ∎

This finishes the proof of Theorem 2. ∎

4. Lower Bounds for Graph Problems on Linear Decision Trees

In this section the method of Section 3 is applied to some languages defined in terms of graphs with weighted edges: the shortest path problem, the minimum perfect matching problem, and the traveling salesperson problem. The main result (Theorem 3) essentially says that a linear decision tree cannot solve these problems fast, i.e., recognize the corresponding languages fast, unless it can compare sums of many input numbers to each other. Thus the comparisons of lengths of paths in any of the standard polynomial time algorithms for the shortest path problem or the minimal perfect matching problem are essential. This observation pinpoints a difference between these problems and, say, the minimum spanning tree problem, which can be solved in polynomial time by algorithms that use only comparisons of weights of single edges.

An equivalent formulation of weighted graph problems as recognition problems in $\mathbf{R}^n$ is obtained as follows. For $m \in \mathbf{N}$ consider the complete graph K_m on m vertices $v_1, ..., v_m$. Fix a numbering $e_1, ..., e_n$ of the $n = \frac{1}{2}m(m-1)$ edges of K_m. Then there is a one-one correspondence between vectors $(x_1, ..., x_n) \in \mathbf{R}^n$ and weight functions w: $\{e_1, ..., e_n\} \to \mathbf{R}_+$, which assign a weight $w(e_i)$ to every edge e_i, the correspondence being given by $w(e_i) = x_i$, for $1 \leqslant i \leqslant n$.

The problem SHORTEST PATH as a decision problem can be formulated as follows: is there a path from v_1 to v_2 of total weight $\leqslant 1$? This problem corresponds to the language

$$\mathrm{SP}(n) := \left\{ \bar{x} \in \mathbf{R}_+^n \mid \exists S \subseteq \{1, ..., n\} \left(\{e_i \mid i \in S\} \text{ forms a path in } K_m \text{ between } v_1 \text{ and } v_2 \text{ and } \sum_{i \in S} x_i \leqslant 1 \right) \right\}.$$

Similarly, the problem MINIMUM PERFECT MATCHING gives rise to the following recognition problem:

$$\mathrm{PM}(n) := \Big\{ \bar{x} \in \mathbf{R}_+^n \mid \exists S \subseteq \{1, ..., n\} \Big(\{e_i \mid i \in S\} \text{ forms a perfect matching in } K_m \text{ and } \sum_{i \in S} x_i \leqslant 1 \Big) \Big\}.$$

Finally, the TRAVELING SALESPERSON problem, i.e., the problem to decide whether there is a Hamiltonian cycle in K_m of total weight $\leqslant 1$, gives rise to the language

$$\mathrm{TSP}(n) := \Big\{ \bar{x} \in \mathbf{R}_+^n \mid \exists S \subseteq \{1, ..., n\} \Big(\{e_i \mid i \in S\} \text{ forms a Hamiltonian cycle in } K_m \text{ and } \sum_{i \in S} x_i \leqslant 1 \Big) \Big\}.$$

There is a geometrical difference between the languages $K(n)$ of Section 3 and the languages just defined. $K(n)$ consists of a union of hyperplanes in $\mathbf{R}^n$, whereas $\mathrm{SP}(n)$, $\mathrm{PM}(n)$, $\mathrm{TSP}(n)$ are unions of closed halfspaces in $\mathbf{R}_+^n$. The following variation of Theorem 2 adapts the results of Section 3 to this situation.

COROLLARY 4.1. *Let, for* $k, p \in \mathbf{N}$,

$$L(p,k) := \Big\{ (y_{ij}, z_{ij})_{1 \leqslant i \leqslant p,\, 1 \leqslant j \leqslant k} \in \mathbf{R}_+^{2pk} \mid \exists I \subseteq \{1, ..., p\} \Big(\sum_{i \in I} \sum_{j=1}^{k} y_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} z_{ij} \leqslant 1 \Big) \Big\}.$$

Let $T_{p,k}$ *be a linear decision tree for inputs from* $\mathbf{R}_+^{2pk}$ *which recognizes* $L(p,k)$, *such that all tests in* $T_{p,k}$ *contain less than* k *negative coefficients. Then* $\mathrm{depth}(T_{p,k}) \geqslant 2^p$.

Proof. Consider the points $\bar{a}$ and $\bar{a}_I$ ($I \subseteq \{1, ..., p\}$) in $\mathbf{R}_+^{2pk}$ as in the proof of Theorem 2. We observe that $\bar{a} \notin L(p,k)$:

$$\sum_{i \in I} \sum_{j=1}^{k} a_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} b_{ij} = \sum_{i \in I} \left(\frac{1}{B} \cdot b^{-i} + \delta \right) + \sum_{i \notin I} \left(\frac{1}{B} \cdot b^{-i} + \delta \right) = 1 + p\delta > 1,$$

for all $I \subseteq \{1, ..., p\}$. But $\bar{a}_I \in L(p,k)$ for all $I \subseteq \{1, ..., p\}$, since

$$\sum_{i \in I} \sum_{j=1}^{k} (a_{ij} - \delta c_{ij}^I) + \sum_{i \notin I} \sum_{j=1}^{k} (b_{ij} - \delta d_{ij}^I) = \sum_{i \in I} \left(\frac{1}{B} \cdot b^{-i} + \delta - \delta d_{i1}^I \right) + \sum_{i \in I} \left(\frac{1}{B} \cdot b^{-i} + \delta - \delta d_{i1}^I \right) = 1.$$

Hence on the path in $T_{p,k}$ which is taken by $\bar{a}$ there must be a test for each $I \subseteq \{1, ..., p\}$ that can distinguish $\bar{a}$ from $\bar{a}_I$. By Corollary 3.6, the hyperplane corresponding to such a test is K_I. Hence the path in $T_{p,k}$ taken by $\bar{a}$ has length $\geqslant 2^p$. ∎

The languages $L_{p,k}$ will be "reduced" to the graph problems we consider here. Thus we get lower bounds in the following manner: from a linear decision tree T which solves the graph problem, using few negative coefficients in its tests, we obtain an LDT of the same structure which recognizes $L(p, k)$, for certain $p, k \in \mathbf{N}$. This implies that $\text{depth}(T) \geqslant 2^p$, by Corollary 4.1.

THEOREM 3. *Let $(T_n)_{n \geqslant 1}$ be a sequence of LDTs, $f: \mathbf{N} \to \mathbf{N}$ a function such that each test in T_n uses less than $f(n)$ negative coefficients. If T_n recognizes one of the languages* SP(n), PM(n), TSP(n), *then* $\text{depth}(T_n) \geqslant 2^{\lfloor \sqrt{n/4f(n)} \rfloor}$, *for n of the form* $\frac{1}{2}m(m-1)$, $m \in \mathbf{N}$.

Note. This lower bound is superpolynomial if $f(n) = o(\sqrt{n}/\log n)$.

Proof. In each of the three cases, we obtain from T_n an LDT $T_{p,k}$ of the same depth as T_n which recognizes $L(p, k)$, where $k := f(n)$ and $p := \lfloor m/2f(n) \rfloor$, and $T_{p,k}$ uses $< k$ negative coefficients in its tests. Then, by Corollary 4.1,

$$\text{depth}(T_n) = \text{depth}(T_{p,k}) \geqslant 2^p = 2^{m/2f(n)} \geqslant 2^{\lfloor \sqrt{n/4f(n)} \rfloor}.$$

(a) Suppose T_n recognizes SP(n). We restrict our attention to a fixed subgraph G_1 of K_m as sketched in Fig. 1. G_1 uses $(2k-1)p + 1 < m$ vertices (among them v_1 and v_2) and $2kp$ edges. The variables x_i corresponding to the edges of this subgraph are renamed y_{ij}, z_{ij} $(1 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant k)$, as indicated in Fig. 1. We consider only input vectors $\bar{x} = (x_1, ..., x_n) \in \mathbf{R}^n_+$ which give weight 2 to all edges not in G_1. Then it is clear that such edges cannot occur in a path of length $\leqslant 1$. We change T_n by fixing the values of the x_i's corresponding to such edges to be 2. The result is an LDT T' for inputs $(y_{ij}, z_{ij})_{1 \leqslant i \leqslant p,\ 1 \leqslant j \leqslant k}$ which accepts

$$\{(y_{ij}, z_{ij})_{i,j} \mid \text{one of the } 2^p \text{ possible paths from } v_1 \text{ to } v_2 \text{ in } G_1 \text{ has weight } \leqslant 1\}.$$

This language obviously equals $L(p, k)$. Furthermore, no test in T' uses $\geqslant f(n) = k$ negative coefficients, since this was true in T_n.

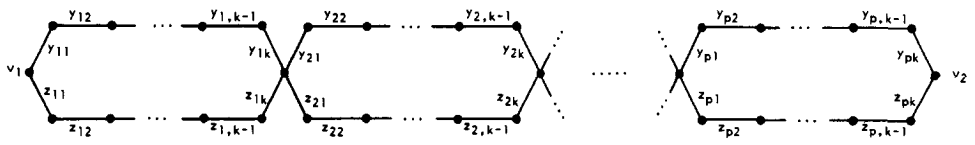

FIG. 1. A graph with 2^p paths from v_1 to v_2.

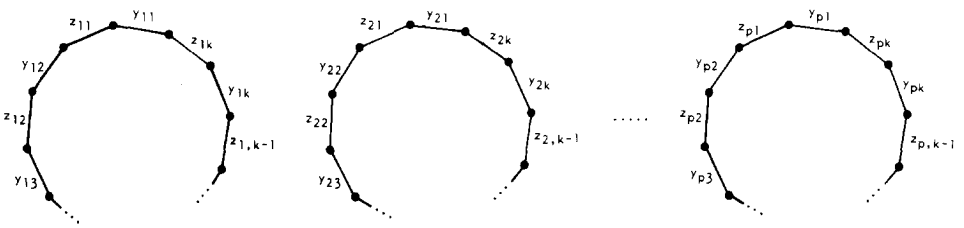

FIG. 2. A graph with 2^p perfect matchings.

(b) Suppose T_n recognizes PM(n). This time we consider only a fixed subgraph G_2 of K_m of the form given in Fig. 2. G_2 has $m = 2pk$ vertices and $2pk$ edges. The variables x_i corresponding to the edges of this subgraph are renamed y_{ij}, z_{ij} ($1 \leqslant i \leqslant p$, $1 \leqslant j \leqslant k$), as indicated in Fig. 2. We obtain a new LDT T'' from T_n by fixing the values of all other x_i to be 2. Clearly, a perfect matching made up from edges in G_2 either contains all edges corresponding to $y_{i1}, \ldots, y_{ik}$ or all edges corresponding to $z_{i1}, \ldots, z_{ik}$, for $1 \leqslant i \leqslant p$. Hence the language

$$\{(y_{ij}, z_{ij})_{i,j} \mid \text{there is a perfect matching in } G_2 \text{ of weight } \leqslant 1\}$$
(which is recognized by T'') equals $L(p, k)$.

(c) Suppose T_n recognizes TSP(n). We consider a fixed subgraph G_3 of K_m of the form given in Fig. 3. Give constant weight 2 to all edges not contained in G_3, and weight $\frac{1}{2} \cdot (1/kp)$ to the edges corresponding to variables u_{ij} ($1 \leqslant i \leqslant p$,

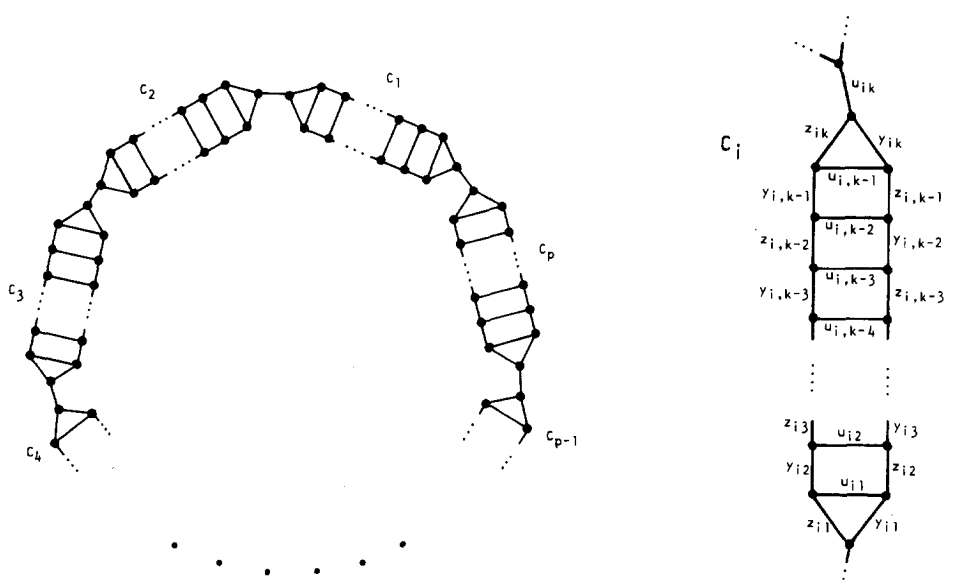

FIG. 3. A graph with 2^p Hamiltonian cycles.

$1 \leqslant j \leqslant k$). Consider the tree T''' obtained from T_n by fixing these input variables and renaming the other ones to y_{ij}, z_{ij} as indicated in Fig. 3. Then a Hamiltonian cycle of weight $\leqslant 1$ uses in component C_i either the edges $y_{i1}, u_{i1}, y_{i2}, ..., y_{ik}, u_{ik}$ or the edges $z_{i1}, u_{i1}, z_{i2}, ..., z_{ik}, u_{ik}$ (for $1 \leqslant i \leqslant k$). Hence the language

$$\{(y_{ij}, z_{ij})_{i,j} | \text{there is a Hamiltonian cycle of length } \leqslant 1 \text{ in } G_3\}$$

equals

$$\left\{(y_{ij}, z_{ij})_{i,j} | \exists I \subseteq \{1, ..., p\} \left(\sum_{i \in I} \sum_{j=1}^{k} y_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} z_{ij} \leqslant \frac{1}{2} \right) \right\},$$

a language so similar to $L(p, k)$ that obviously the lower bound 2^p for the depth of T''' is implied. ▮

Remark (Application to geometrical instances). The lower bound of Theorem 3 stays valid if the LDT T_n is only required to solve the respective graph problem for the following restricted class of "geometrical instances": incomplete graphs on m vertices that can be drawn in the Euclidean plane in such a way that the edges are straight lines, no two edges cross, and the weight of each edge equals its length. (We assume that T_n uses an additional type of test that allows it to find out whether an edge is present in the input graph or not.) To construct "difficult inputs" for an LDT T_n that can decide the graph problem only for instances from this restricted class, we can use the same subgraphs of K_m as in Figs. 1–3. But we cannot use the same weight sets as in the proof of Theorem 3, since there the weights were vastly different from each other (e.g., a_{11} is very much larger than a_{12}), and it is not clear if one can draw graphs in the required way with these weights as edge lengths. If, however, the edge weights y_{ij}, z_{ij} ($1 \leqslant i \leqslant p$, $1 \leqslant j \leqslant k$) of the graphs depicted in Figs. 1–3 are all equal, these graphs can be drawn in this manner. We will exploit in the following that the same is true if the edge weights are *nearly* the same, say if $(1/pk)(1-\varepsilon) \leqslant y_{ij}, z_{ij} \leqslant 1/pk$ for all i, j, for a sufficiently small $\varepsilon = \varepsilon(p, k)$. There is an easy way to obtain such "geometrical" weight sets from arbitrary ones: add a large constant to all edge weights, then scale down by a constant factor. More precisely, if any weight set $(\tilde{y}_{ij}, \tilde{z}_{ij})_{i,j}$ with $0 \leqslant \tilde{y}_{ij}, \tilde{z}_{ij} \leqslant 1$ is given, the new weight set $(y_{ij}, z_{ij})_{i,j}$ defined by

$$y_{ij} := \varepsilon \cdot \tilde{y}_{ij} + \frac{1-\varepsilon}{pk}, \qquad z_{ij} := \varepsilon \cdot \tilde{z}_{ij} + \frac{1-\varepsilon}{pk}$$

belongs to a graph which can be drawn in the required manner. This observation leads to the following "reduction procedure," described here for the case of SHORTEST PATH. Suppose an LDT T_n is given which accepts weight sets for subgraphs of K_m which admit a path of length $\leqslant 1$ between v_1 and v_2, but T_n does so only for input vectors which arise from "geometrical instances" in the way described above. We define k and p as before, and restrict our attention to the

subgraph G_1 of Fig. 1, renaming variables as in the proof of Theorem 3. Modify T_n as follows: replace tests

$$\sum_{i,j} \alpha_{ij} y_{ij} + \sum_{i,j} \beta_{ij} z_{ij} : \gamma$$

by

$$\sum_{i,j} \alpha_{ij} \tilde{y}_{ij} + \sum_{i,j} \beta_{ij} \tilde{z}_{ij} : \frac{1}{\varepsilon} \cdot \left[\gamma - \sum_{i,j} (\alpha_{ij} + \beta_{ij}) \cdot \frac{1-\varepsilon}{pk} \right],$$

and call the tree so obtained $\tilde{T}_n$. Then it is easily checked that for $0 \leqslant \tilde{y}_{ij}, \tilde{z}_{ij} \leqslant 1$

$$\tilde{T}_n \text{ accepts } (\tilde{y}_{ij}, \tilde{z}_{ij})_{i,j} \text{ if and only if (by definition of } \tilde{T}_n)$$

$$T_n \text{ accepts } \left(\varepsilon \tilde{y}_{ij} + \frac{1-\varepsilon}{pk}, \varepsilon \tilde{z}_{ij} + \frac{1-\varepsilon}{pk} \right)_{i,j} \text{ if and only if (by the structure of } G_1)$$

$$\exists I \subseteq \{1, \ldots, p\} \left(\sum_{i \in I} \sum_{j=1}^{k} \left(\varepsilon \tilde{y}_{ij} + \frac{1-\varepsilon}{pk} \right) + \sum_{i \notin I} \sum_{j=1}^{k} \left(\varepsilon \tilde{z}_{ij} + \frac{1-\varepsilon}{pk} \right) \leqslant 1 \right)$$

$$\text{if and only if } \exists I \subseteq \{1, \ldots, p\} \left(\sum_{i \in I} \sum_{j=1}^{k} \tilde{y}_{ij} + \sum_{i \notin I} \sum_{j=1}^{k} \tilde{z}_{ij} \leqslant 1 \right).$$

Thus $\tilde{T}_n$ recognizes $L(p, k)$ for inputs from $[0, 1]^{2pk}$ (the language $L(p, k)$ was defined in Corollary 4.1). The lower bound proof of Section 3 uses only inputs in $[0, 1]^{2pk}$, hence $\tilde{T}_n$ has depth $\geqslant 2^p$, and so does T_n.

Similar constructions yield the same result for PM(n) and TSP(n).

ACKNOWLEDGMENTS

We thank Friedhelm Meyer auf der Heide and György Turán for helpful discussions.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, MA, 1974.
2. P. BEAME, Limits on the power of concurrent-write parallel machines, *in* "Proceedings, 18th ACM Sympos. Theory of Computing, 1986," pp. 169–176.
3. P. BEAME AND J. HASTAD, Optimal bounds for decision problems on the CRCW PRAM, *in* "Proceedings, 19th ACM Sympos. Theory of Computing, 1987," pp. 83–93.
4. M. BEN-OR, Lower bounds for algebraic computation trees, *in* "Proceedings, 15th ACM Sympos. Theory of Computing, 1983," pp. 80–86.
5. A. K. CHANDRA, M. L. FURST, AND R. J. LIPTON, Multi-party protocols, *in* "Proceedings, 15th ACM Sympos. Theory of Computing, 1983," pp. 94–99.

6. M. DIETZFELBINGER AND W. MAASS, Two lower bound arguments with "inaccessible" numbers, *in* "Structure in Complexity Theory," pp. 161–183, Lecture Notes in Computer Science Vol. 223, Springer-Verlag, Berlin, 1986.
7. D. P. DOBKIN AND R. J. LIPTON, A lower bound of $n^2/2$ on linear search programs for the knapsack problem, *J. Comput. System Sci.* **16** (1978), 413–417.
8. D. P. DOBKIN AND R. J. LIPTON, On the complexity of computations under varying sets of primitives, *J. Comput. System Sci.* **18** (1979), 86–91.
9. F. E. FICH, F. MEYER AUF DER HEIDE, P. RAGDE, AND A. WIGDERSON, One, two, three ... infinity: Lower bounds for parallel computation, *in* "Proceedings, 17th ACM Sympos. Theory of Computing, 1985," pp. 48–58.
10. R. L. GRAHAM, B. L. ROTHSCHILD, AND J. H. SPENCER, "Ramsey Theory," Wiley, New York, 1980.
11. M. LI AND Y. YESHA, New lower bounds for parallel computation, *in* "Proceedings, 18th ACM Sympos. Theory of Computing, 1986," pp. 177–187.
12. W. MAASS, "On the Use of Inaccessible Numbers and Order Indiscernibles in Lower Bound Arguments for Random Access Machines," Research Report in Computer Science No. 4, University of Illinois at Chicago, 1985; *J. Symbolic Logic*, in press.
13. F. MEYER AUF DER HEIDE AND R. REISCHUK, On the limits to speed up parallel machines by large hardware and unbounded communication, *in* "Proceedings, 25th IEEE Sympos. Found. of Comput. Sci., 1984," pp. 56–64.
14. F. MEYER AUF DER HEIDE, A polynomial linear search algorithm for the n-dimensional knapsack problem, *J. Assoc. Comput. Mach.* **31** (1984), 668–676.
15. F. MEYER AUF DER HEIDE, Lower bounds for solving linear diophantine equations on random access machines, *J. Assoc. Comput. Mach.* **32** (1985), 929–937.
16. F. MEYER AUF DER HEIDE, Fast algorithms for n-dimensional restrictions of hard problems, *in* "Proceedings, 17th ACM Sympos. Theory of Computing, 1985," pp. 413–420.
17. W. J. PAUL AND J. SIMON, Decision trees and random access machines, *in* "Logic et Algorithmic," Monographies de l'Enseignement mathématique no. 20, pp. 331–340, Univ. Geneva, Switzerland, 1982.
18. L. J. STOCKMEYER AND U. VISHKIN, Simulation of parallel random access machines by circuits, *SIAM J. Comput.* **5** (1984), 409–422.
19. E. UKKONEN, Exponential lower bounds for some NP-complete problems in a restricted linear decision tree model, *BIT* **23** (1983), 181–192.
20. U. VISHKIN AND A. WIGDERSON, Trade-offs between depth and width in parallel computation, *SIAM J. Comput.* **14** (1985), 303–314.
21. A. C. YAO, On the complexity of comparison problems using linear functions, *in* "Proceedings, 16th IEEE Sympos. Found. of Comput. Sci., 1975," pp. 85–89.