Theoretical Computer Science 19 (1982) 321–330
North-Holland Publishing Company

COMPARISONS BETWEEN LINEAR FUNCTIONS CAN HELP

Marc SNIR*
Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom

Communicated by M.S. Paterson
Received July 1980
Revised April 1981

Abstract. An example is provided of a sorting-type decision problem which can be solved in fewer steps by using comparisons between linear functions of the inputs, rather than comparisons between the inputs themselves. This disproves a conjecture of Yao [14] and Yap [16]. Several extensions are presented.

1. Introduction

The worst case behavior of decision algorithms using comparisons between inputs has been extensively studied for various sorting-type problems. When the inputs are numbers, one can consider the use of more general comparisons. Thus Rabin [10] considered the use of comparisons between meromorphic functions of the inputs. Many other authors [1–5], [11–16] examined the use of comparisons between linear functions of the inputs. No unconstrained sorting-type problem was found which can be solved in fewer steps using linear comparisons, and it was conjectured by Yao [14] and Yap [16], and proved in a few special cases by the latter that linear comparisons cannot help.

We exhibit a counter-example to this conjecture; this counter-example is then modified to refute weaker versions of the conjecture. It is shown false, even for Yes/No problems, and for problems defined by convex sets. In the process, several general results concerning decision algorithms using linear comparisons are obtained.

2. Terminology

Decision trees, as a model for decision algorithms, occur initially in Rabin's paper [10], and are used with minor modifications by all the authors mentioned in the

* Present address: Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, U.S.A.

0304-3975/82/0000–0000/$02.75

introduction. Although we must settle, for the sake of definiteness, on a particular formulation, our results are valid for the different varieties of models used by these authors.

Let $\mathbb{R}^n$ be the set of n-tuples of real numbers (n-tuples of inputs). A *decision problem* P in $\mathbb{R}^n$ is given by finite family (D_i) of disjoint open sets such that $\bigcup \mathrm{cl}\, D_i = \mathbb{R}^n$. If Ω is a family of continuous functionals, then P *is of type* Ω if the sets D_i can be built from sets of the form $[f_1(x) > f_2(x)]$, with $f_1, f_2 \in \Omega$, using the operations of (finite) union and intersection. In particular, P is of *sorting-type* if Ω is the family of projections $\Pi_i(x) = x_i$, P is linear if Ω is the family of linear (affine) functionals.

An *Ω-decision tree* consists of a labelled binary tree T: each internal node of T is labelled with a *comparison* of the form $f_1(x) : f_2(x)$, where $f_1, f_2 \in \Omega$, and the two edges leaving this node are associated with different outcomes $f_1(x) < f_2(x)$, $f_1(x) > f_2(x)$. In particular, in sorting type decision trees comparisons are between inputs, in a linear decision tree linear functions of the inputs are compared.

The tree T encodes the decision procedure executed on inputs $x \in \mathbb{R}^n$ by moving down the tree beginning from the root, performing on x each comparison encountered, and branching according to the result, until a leaf is reached, or equality occurs at some comparison. Thus, we say that x *reaches* node u of T if x satisfies all the inequalities associated with edges on the path leading from the root of T to u.

T solves the problem P if each leaf of T can be associated with one of the sets defining P, such that x reaches a leaf associated with D_i only if $x \in \mathrm{cl}\, D_i$. For each node u of T let S_u be the set of inputs reaching u; T solves P iff the partition of $\mathbb{R}^n$ given by $\{S_u : u \text{ is a leaf of } T\}$ is a refinement of the partition defining P – ignoring a boundary set.

We make two remarks concerning our model:

(i) The procedure encoded by T does not yield an answer on the boundary set of inputs for which equality obtains at some comparison. This can be remedied by associating with each comparison $f_1(x) : f_2(x)$ complementary outcomes $f_1(x) > f_2(x)$, $f_1(x) \leq f_2(x)$. Indeed, T solves P iff the modified tree solves P and the latter yields a correct answer for any tuple of inputs. However, trees which are labelled only with strong inequalities are easier to deal with.

(ii) There is more than one correct answer for a tuple of inputs which belongs to the common boundary of two or more sets D_i; the decision procedure yields (at most) one of these answers.

The *Ω-complexity* of a decision problem P is defined to be the minimal depth of an Ω-decision tree solving P.

A problem of type Ω can always be solved by an Ω-decision tree. Use of a richer family Ω might however lead to a less complex solution. Yao and Yap conjectured that this does not happen when the family of projections is augmented by the addition of linear functions.

3. Counter-example

Consider the following 4 permutations on the numbers 0 . . . 9:

$$\begin{array}{lcccccccccc} S_0: & & 0 & 4 & 5 & & 1 & 2 & 8 & 6 & 7 & 3 & 9 \\ S_1: & 8 & 1 & 4 & 5 & & 2 & 3 & & 6 & 7 & 0 & 9 \\ S_2: & 8 & 2 & 4 & 5 & & 3 & 0 & & 6 & 7 & 1 & 9 \\ S_3: & 8 & 3 & 4 & 5 & 9 & 0 & 1 & & 6 & 7 & 2 & \end{array}$$

Each sequence S_i defines on $0 \ldots 9$ an *order* relation $<_i$ and a *contiguity* relation $|_i$:

$a <_i b$ if a occurs before b in the sequence S_i, and

$a |_i b$ if a and b occur (in some order) in consecutive positions in S_i.

The four sequences above have the following properties:

(i) $a |_i b$ for at most one index i, with the following exceptions:

(i.i) $4|_i 5$ and $6|_i 7$ for $i = 0, \ldots, 3$; these pairs of numbers occur in the same order in each sequence;

(i.ii) $2|_i 8$ for $i = 0, 2$ and $0|_i 9$ for $i = 1, 3$; these pairs of numbers occur in opposite order in the two sequences where they are contiguous.

(ii) $a <_i b$ or $b <_i a$ for at least 3 indices i, with the exception of the pairs $\{0, 2\}$ and $\{1, 3\}$; the numbers 0, 1, 2, 3 are permuted cyclically in $S_0, \ldots, S_3$, so that $0 <_i 2$ for $i = 0, 3$, $2 <_i 0$ for $i = 1, 2$ and similarly, $1 <_i 3$ for $i = 0, 1$ and $3 <_i 1$ for $i = 2, 3$.

Define the following 5 subsets of $\mathbb{R}^{10}$:

$$\begin{aligned} D_i &= \{x : a <_i b \Rightarrow x_a < x_b\}, \quad i = 0, \ldots, 3, \\ D_4 &= \mathbb{R}^{10} \Big\backslash \Big(\bigcup_{i=0}^{3} \operatorname{cl} D_i \Big). \end{aligned} \tag{1}$$

The problem P defined by these 5 sets provides the required counter-example. It consists of deciding whether 10 inputs are ordered according to one of the permutations defined by $S_0, \ldots, S_3$, and if so, which one.

Claim 1. *The problem P defined above can be solved by a linear decision tree of depth* 10 *but by no sorting-type decision tree of depth less than* 11.

Proof. A linear decision tree of depth 10 solving P is illustrated in Fig. 1. We have put in brackets the information available after each comparison. The subtree T_i checks that $x \in D_i$; it consists of 8 comparisons performed sequentially, one for each pair of contiguous numbers in S_i, with the exception of the pair occurring at the node immediately above T_i.

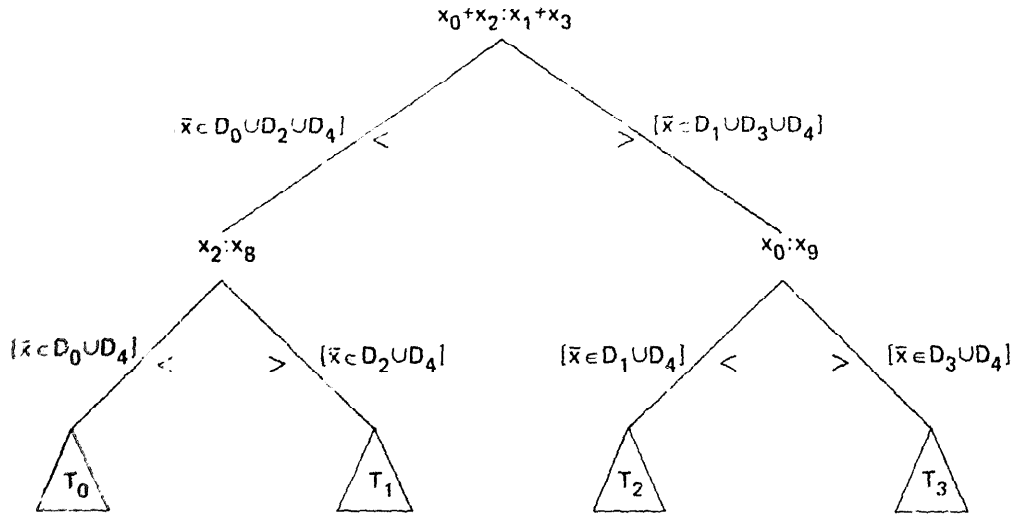

Fig. 1.

Now let T be a sorting-type decision tree solving P. If a, b are consecutive numbers in S_i, and u is a leaf of T associated with D_i, then the inequality $x_a < x_b$ labels some edge on the path leading from the root of T to u. Indeed, from the set of inequalities labelling this path we can infer that $x_a < x_b$. It follows that this set contains a chain of inequalities

$$x_a = x_{r_0} < x_{r_1},\ x_{r_1} < x_{r_2},\ \ldots,\ x_{r_{k-1}} < x_{r_k} = x_b.$$

But no $x \in D_i$ can satisfy both inequalities $x_a < x_r < x_b$. Thus $k = 1$, and $x_a < x_b$ labels an edge on the path to u.

Since all the n-tuples in D_i $(0 \leq i \leq 3)$ are ordered according to the same permutation, they fulfil the same inequalities, and follow the same path p_i in T. On this path we have 9 comparisons $x_a : x_b$, one for each pair of contiguous numbers in S_i. The claim thus follows if we show that path p_i contains two comparisons $x_a : x_b$, where $a \not<_i b$. Equivalently, we have to show that there exists a node u in T, and an index $0 \leq i \leq 3$, such that the inequalities $x_a < x_b$ labelling the edges on the path to u fulfil $a <_i b$ whereas $a \not<_i b$ for at least two of them.

Since the comparisons $x_4 : x_5$ and $x_6 : x_7$ occur on each path p_i, we assume w.l.g. that they are performed first. Let $x_a : x_b$ be the comparison at the node reached by the path labelled with $x_4 < x_5$ and $x_6 < x_7$. We distinguish two cases:

Case 1. $a <_i b$ for 3 indices i_1, i_2, i_3. From (i) and (ii) we have that $a \|_i b$ for at most one of these indices, say i_1, so that $a \not<_{i_2} b$ and $a \not<_{i_3} b$. Let $x_c : x_d$ be the comparison at the node reached by the branch labelled with $x_a < x_b$. If $c \not<_{i_2} d$ or $c \not<_{i_3} d$ we are done. Otherwise we have $c \|_{i_2} d$ and $c \|_{i_3} d$, so that $\{c, d\} = \{2, 8\}$ and $\{i_2, i_3\} = \{0, 2\}$ or $\{c, d\} = \{0, 9\}$ and $\{i_2, i_3\} = \{1, 3\}$. We shall discuss the first alternative, the second one being symmetrical. We have $c \not<_{i_1} d$ and we assume w.l.g. that $c <_{i_1} d$. Also, either $c <_{i_2} d$ or $c <_{i_3} d$, so we can assume w.l.g. that $c <_{i_2} d$. Let $x_e : x_f$ be the comparison at the node reached by the branch labelled with $x_c < x_d$.

If $e \not\prec_{i_1} f$ or $e \not\prec_{i_2} f$ we are done (since $c \not\prec_{i_1} d$ and $a \not\prec_{i_2} b$). But $i_1 \in \{1, 3\}$ and $i_2 \in \{0, 2\}$ so that no pair of numbers (different from $\{4, 5\}$ and $\{6, 7\}$) fulfil both conditions $e \mid_{i_1} f$ and $e \mid_{i_2} f$.

Case 2. $a <_i b$ for exactly two indices i. Thus either $\{a, b\} = \{0, 2\}$ or $\{a, b\} = \{1, 3\}$. Assume that $\{a, b\} = \{0, 2\}$. We have $0 <_i 2$ but $0 \not<_i 2$ for $i = 0, 3$. We are done unless comparison $c : d$ reached by the branch labelled with $x_0 < x_2$ fulfils $c \mid_i d$ for $i = 0, 3$. But no pair of numbers (different from $\{4, 5\}$ and $\{6, 7\}$) fulfil these conditions. The case $\{a, b\} = \{1, 3\}$ is handled similarly.

4. Modified counter-examples

Once the original hypothesis has been disproved, two questions come naturally to mind:

(i) can the hypothesis be saved by not unduly restricting its field of application (the monster barring approach) and

(ii) can the hypothesis be shown to be approximately true, by bounding the maximal gap between sorting type complexity and linear complexity.

We shall partially vindicate in this section a negative answer to the first question by refuting two plausible restricted conjectures, and proceed to explore the second question in the next section.

The first restricted subclass of decision problems we consider consists of problems with two outcomes, that is Yes/No problems.

It turns out that our original counter-example can be modified to belong to this class. This follows from the next theorem, which is interesting in its own right. The terminology used is taken from [6].

Theorem 1. *Let P be the decision problem defined in $\mathbb{R}^n$ by the sets $\{E_1, \ldots, E_r, F_1, \ldots, F_s\}$, and let P' be the problem defined by the sets $\{\bigcup_{i=1}^{r} E_i, F_1, \ldots, F_s\}$. Assume that the (affine) dimension of $\mathrm{cl}\, E_i \cap \mathrm{cl}\, E_j$ is less than $n-1$ for every $1 \leq i < j \leq r$. Then a linear decision tree solves P' iff it solves P.*

Theorem 1 asserts that we cannot isolate, using hyperplanes, a set which is the union of "essentially disjoint" components, without separating these components.

Lemma 1. *Let B be an open convex set and let B_i be a finite family of closed sets such that*

(i) $B \subset \cup B_i$;

(ii) $\dim B_i \cap B_j < n-1$ *for any* $i \neq j$.

Then $B \subset B_i$ for some i.

Proof. We can assume w.l.g. that for each i, $B \cap B_i \not\subset \bigcup_{j \neq i} B_j$. Assume that $B \not\subset B_1$. Let $y \in (B \cap B_1) \setminus \bigcup_{j \neq 1} B_j$ and let $S = \bigcup_{j \neq 1} (B_1 \cap B_j)$. The set S is contained in the

union of finitely many $n-2$ dimensional flats and $y \notin S$, so that the cone yS, consisting of the rays originating from y and going through S, is contained in the union of finitely many $n-1$ dimensional flats, and has therefore an empty interior. If $x \in B \backslash B_1$, then the segment xy is contained in B, and therefore in $\bigcup B_i$. It follows that this segment intersects S. Thus $B \backslash B_1$ is contained in yS, and has an empty interior. It follows that $B \backslash B_1$ is empty, and $B \subset B_1$.

Proof of Theorem 1. Obviously, a decision tree solving P solves P' as well. Conversely, let T be a linear decision tree solving P'. For each leaf u of T the set S_u of inputs reaching u is an open convex set contained in the closure of one of the sets defining P'. But if $S_u \subset \bigcup_{i=1}^{r} \operatorname{cl} E_i$ we have from the assumption on the sets E_i, and from lemma 1, that $S_u \subset \operatorname{cl} E_i$ for some i. Each leaf of T is reached by inputs belonging to the closure of a unique set out of the sets defining P, so that T solves P.

Corollary 1. *There exist a Yes/No sorting-type decision problem with linear complexity* $\leqslant 10$ *and sorting complexity* >11.

Proof. Let P be the problem defined in $\mathbb{R}^{10}$ by the sets D_i, $0 \leqslant i \leqslant 4$, of (1). The permutations defining the sets D_i, $i = 0, \ldots, 3$ differ in the relative ordering of more than one pair of numbers, so that in each of the sets $\operatorname{cl} D_i \cap \operatorname{cl} D_j$, $0 \leqslant i < j \leqslant 3$, there is more than one equality between inputs which is identically satisfied, and $\dim \operatorname{cl} D_i \cap \operatorname{cl} D_j < 9$. It follows, by Theorem 1, that a linear decision tree solves P iff it solves the problem P' defined by the two sets $\{\bigcup_{i=0}^{3} D_i, D_4\}$. P' cannot therefore be solved by a sorting type decision tree of depth 10 but can be solved by a linear decision tree of this depth.

Theorem 1 can be used to yield other similar equivalences, most of which are pretty obvious in the restricted case of sorting-type decision trees. We bring three examples:

(i) A linear decision tree which finds the kth largest input out of a list of n inputs, finds also which $k-1$ inputs are greater than it, and which $n-k$ are less than it (see [8, 5.3.3, ex. 2]).

(ii) A linear decision tree which checks whether n inputs are ordered according to an odd or an even permutation, finds also this permutation (see [8, §5.3.1, ex. 29]).

(iii) A linear decision tree which finds the maximal points out of a set of n points in the plane (see [9]), finds these points in order (maximal points are ordered in inverse order according to the x coordinate and the y coordinate). This remark yields immediately the lower bound of $n \lg n$ proven in [4].

Theorem 1 can in fact be viewed as a tool for "normalizing" linear decisions problems by decomposing them into their "true" components.

Another natural subclass of sorting-type decision problems consists of the problems defined by convex sets. The convexity requirement, which occurs in Rabin's

original definition of decision problems [1, p. 645] is fulfilled by many interesting problems, such as sorting or finding the first k out of n elements. Many other problems are equivalent to convex problems, by Theorem 1.

Once more, our counterexample can be modified to fall within this class. This is done by partitioning the set D_4 into convex components, and refining the linear decision tree of Fig. 1 to yield a solution for the new problem. Thus we have

Theorem 2. *There exists a convex sorting-type decision problem with linear complexity* ≤ 10 *and sorting complexity* >11.

The reader is referred to [12] for the details of the straightforward, if tedious, construction of the problem.

5. Composite problems

The gain achieved by using linear comparisons is not limited to one step; greater gaps can be exhibited by composing the counter-example with itself.

Definition. Let P_i, $i = 1, \ldots, k$, be problems defined in $\mathbb{R}^{n_i}$ by the sets $\{D_j^i\}_{j=1}^{m_i}$. The *Cartesian product* $P_1 \times \cdots \times P_k$ of these problems is the problem defined in $\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} = \mathbb{R}^{n_1+\cdots+n_k}$ by the sets $\{D_{j_1}^1 \times \cdots \times D_{j_k}^k : 1 \leq j_i \leq m_i\}$; the nth *power* P^n of a problem P is defined to be the n-fold cartesian product of P.

The problem $P = P_1 \times \cdots \times P_k$ is the composition of k independent problems, defined on disjoint sets of inputs. P can therefore be solved by solving separately each problem P_i. It follows that the Ω complexity of P is at most the sum of the Ω-complexities of its components $P_1, \ldots, P_k$ (we have to dispose of the technical requirement that Ω be closed under composition with projections, so that if $f: \mathbb{R}^n \to \mathbb{R}$ is in Ω, then $g(x_1, \ldots, x_m) = f(x_{i_1}, \ldots, x_{i_n})$ is also in Ω). For sorting type problems this bound is tight, as shown by the following theorem.

Theorem 3. *The sorting complexity of* $P_1 \times \cdots \times P_k$ *is equal to the sum of the sorting complexities of* $P_1, \ldots, P_k$.

Corollary 2. *For every* n *there is a sorting-type problem of sorting complexity* $>11n$, *and linear complexity* $\leq 10n$.

Proof. By theorem 3 and the discussion preceding it, the nth power P^n of the problem P defined in Section 3 has the required properties.

Proof of Theorem 3. We shall prove the theorem for $k = 2$, the general case following by induction. Let P_1 be defined on $x_1, \ldots, x_m$ and have sorting complexity

c_1, and P_2 be defined on $y_1, \ldots, y_n$, and have sorting complexity c_2. Let T be a sorting type decision tree solving $P_1 \times P_2$. We can delete from T any comparison of the form $x_i : y_j$ (by assuming the outcome to be $x_i < y_j$), and the resulting decision tree still solves $P_1 \times P_2$. We assume therefore w.l.g. that T does not contain "mixed" comparisons. There is an adversary (oracle) $\mathcal{A}_1$ that forces any decision procedure for P_1 to perform c_1 comparisons and an adversary $\mathcal{A}_2$ forcing c_2 comparisons for P_2. The adversary that answers comparisons involving only x_i's according to the $\mathcal{A}_1$ strategy and answers comparisons involving only y_j's according to the $\mathcal{A}_2$ strategy forces c_1 comparisons involving x_i's and c_2 comparisons involving y_j's, in any decision procedure for $P_1 \times P_2$ not containing "mixed" comparisons. The result now follows.

The reader familiar with the theory of games will no doubt have recognized in the last proof the notion of the Cartesian product of games [7].

The crux of the above proof is that "mixed" comparisons, involving both x_i's and y_j's, do not help in solving $P_1 \times P_2$. This is not true of every class of comparisons. An example is given in [12] of two problems defined by linear nonhomogeneous inequalities, that can be solved concurrently with less linear comparisons than the sum of the number of comparisons required to solve each one separately.

The result of Corollary 2 can be easily extended to the two restricted subclasses of problems we have considered.

Corollary 3. (i) *For every n there exists a Yes/No sorting type problem of sorting complexity $>11n$ and linear complexity $\leqslant 10n$.*

(ii) *For every n there exists a convex sorting type problem of sorting complexity $>11n$ and linear complexity $\leqslant 10n$.*

A problem satisfying the first claim can be built from P^n by using the technique of Corollary 1; the second claim follows from the remark that the product of convex problems is a convex problem.

We have shown that the use of linear comparisons can save an unbounded number of comparisons. Yet, the number of comparisons has been reduced only by a constant factor. As linear comparisons can be harder to perform, the overall computational complexity is not necessarily reduced. We believe that our methods can be used to exhibit larger ratios between the two complexity measures. However, lower bounds on sorting complexity that are nonlinear in the number of inputs seems to be achievable only through the use of information theoretical arguments, and are therefore valid for linear complexity as well. Proving a nonlinear speedup seems therefore to require new techniques.

6. Concluding remarks

The results obtained in previous sections are quite insensitive to variations in the decision problem/decision tree models. We can allow the use of weak

inequalities for defining the sets of a problem, allow the use of weak inequalities as labels in decision trees, or add to each comparison a third outcome, namely equality. The meaningful requirements are that the sets defining a problem have disjoint interiors, and that for internal points each comparison determines a unique outcome and the tree yields a correct answer.

The conjecture that linear comparisons do not help in solving sorting-type problems was disproved using an ad-hoc problem. One would like to know how it stands with respect to "natural" sorting type problems. In particular, can linear comparisons help in sorting? We conjecture it is not so.

As an interesting combinatorial question we mention the problem of finding a minimal counter-example to the conjecture we disproved. Yap [17] has shown that the conjecture is valid for decisions trees of depth ≤ 2.

Our result can be viewed as a trade off result in complexity theory: Using more complex comparisons, one can solve certain problems with less comparisons. The same questions can be asked anew for other classes of comparisons. For example: Can a linear problem be solved in less comparisons if quadratic comparisons are allowed? For similar results see [11].

Finally, this work contains several general results concerning linear decision problems; a more systematic treatment can be found in [13]. We hope that the material presented here may convey to the reader some of the mathematical nicety of this topic, with its interaction of geometrical and combinatorial arguments.

Acknowledgment

I am grateful to Martin Fürer who simplified my original counter-example to yield the current one and to Leslie Valiant for his useful suggestions.

References

[1] D.P. Dobkin, A nonlinear lower bound on linear search tree programs for solving knapsack problems, *J. Comput. System Sci.* **13** (1976) 69–73.

[2] D.P. Dobkin and R.J. Lipton, A lower bound of $1/2n^2$ on linear search programs for the knapsack problem, *J. Comput. System Sci.* **16** (1978) 413–417.

[3] 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.

[4] P. van Emde Boas, On the $\Omega(n \lg n)$ lower bound for convex hull and maximal vector determination, *Information Processing Lett.* **10** (1980) 132–136.

[5] M. Fredman and B. Weide, On the complexity of computing the measure of $\bigcup[a_i, b_i]$, *Comm. ACM* **21** (1978) 540–544.

[6] B. Grünbaum, *Convex Polytopes* (Wiley, London, 1967).

[7] J.C. Holladay, Cartesian products of termination games, in M. Dresher, A.W. Tucker and P. Wolfe, Eds., *Contribution to the Theory of Games Vol. 3*, Annals of Mathematics Studies **39** (Princeton University Press, Princeton, NJ, 1957) 189–200.

[8] D.E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching* (Addison-Wesley, Reading, MA, 1973).

[9] H.T. Kung, F. Luccio and F.P. Preparata, On finding the maxima of a set of vectors. *J. ACM* **22** (1975) 469–476.
[10] M.O. Rabin, Proving simultaneous positivity of linear forms, *J. Comput. System Sci.* **6** (1972) 639–650.
[11] E.M. Reingold, Computing the maxima and the median, *Proc. IEEE 12th Annual Symposium on Switching and Automata Theory* (1971) 216–218.
[12] M Snir, Comparison between linear functions can help, Research Report CSR-66-80, Computer Science Department, University of Edinburgh (1980).
[13] M. Snir, Proving lower bounds for linear decision trees, *Proc. 8th International Colloquium on Automata, Languages and Programming*, (1981), 305–315.
[14] A.C. Yao, On the complexity of comparison problems using linear functions, *Proc. IEEE 16th Annual Symposium on Foundations of Computer Science* (1975) 85–89.
[15] A.C. Yao and R.L. Rivest. On the polyhedral decision problem, *SIAM J. Comput.* **9** (1980) 343–347.
[16] C.K. Yap, On lifted problems, *Proc. IEEE 19th Annual Symposium on Foundations of Computer Science* (1978) 267–279.
[17] C.K. Yap, Private communication.