

Highlights

Linear threshold functions in decision lists, decision trees, and depth-2 circuits

Yogesh Dahiya, Vignesh K, Meena Mahajan, Karteek Sreenivasaiah

- Polynomial-size constant-rank linear decision trees can be converted to polynomial-size depth-2 threshold circuits.
- For decision lists that query conjunctions of linear threshold functions, constant-arity conjunction is equivalent to conjunction of two functions.
- Polynomial-size exact linear decision lists are equivalent to decision lists querying the conjunction of two linear threshold functions, and strictly more powerful than polynomial-size linear decision lists.

Linear threshold functions in decision lists, decision trees, and depth-2 circuits

Yogesh Dahiya^a, Vignesh K^b, Meena Mahajan^a, Karteek Sreenivasaiah^b

^a*The Institute of Mathematical Sciences, (a CI of Homi Bhabha National Institute, Mumbai), CIT Campus, Taramani, Chennai, 600113, TN, India*

^b*Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Hyderabad, 502285, Telangana, India*

Abstract

We show that polynomial-size constant-rank linear decision trees (LDTs) can be converted to polynomial-size depth-2 threshold circuits $\text{LTF} \circ \text{LTF}$. An intermediate construct is polynomial-size decision lists that query a conjunction of a constant number of linear threshold functions (LTFs); we show that these are equivalent to polynomial-size exact linear decision lists (ELDLs) i.e. decision lists querying exact threshold functions (ELTFs).

Keywords: linear threshold functions; decision lists; decision trees; depth-2 circuits; tree rank.

1. Introduction

Understanding the power of linear threshold functions LTFs as a primitive operation is a significant question in complexity theory. At the frontier of circuit lower bounds is the class TC^0 : polynomial-size constant-depth circuits using LTF gates. Currently we do not know explicit lower bounds even for polynomial-size depth-2 threshold circuits. We denote this class $\text{LTF} \circ \text{LTF}$. Restricting this circuit class further to polynomial-weight LTFs (MAJ gates) at either one or both of the two levels gives the classes $\text{MAJ} \circ \text{LTF}$, $\text{LTF} \circ \text{MAJ}$, and $\text{MAJ} \circ \text{MAJ}$. The exact relationship among these classes, and non-trivial lower bounds, are known — $\text{MAJ} \circ \text{MAJ}$ equals $\text{MAJ} \circ \text{LTF}$ and is strictly weaker

Email addresses: yogeshdahiya@imsc.res.in (Yogesh Dahiya), vigneshkrishnan2000@gmail.com (Vignesh K), meena@imsc.res.in (Meena Mahajan), karteek@cse.iith.ac.in (Karteek Sreenivasaiah)

than $\text{LTF} \circ \text{MAJ}$ [1], which is strictly weaker than $\text{LTF} \circ \text{LTF}$ [2]. Another way to use LTFs as a primitive is as a query, in linear decision lists LDLs and linear decision trees LDTs, see [3, 4]. Here, the usual complexity measure is query complexity, but one can also consider size, which is a measure of the space required to store the function in this representation. In our work, the size measure will be of particular interest. It is known that the class LDL of functions with polynomial-size LDLs is contained in $\text{LTF} \circ \text{LTF}$ [4], but the same is not known for the class LDT of functions computable by polynomial-size LDTs. For the classes $\text{LTF} \circ \text{LTF}$ and LDT, a common upper bound is $\text{MAJ} \circ \text{MAJ} \circ \text{MAJ}$. While we do not know even quadratic size lower bounds against $\text{LTF} \circ \text{LTF}$ computing an explicit function, we know exponential lower bounds against LDTs computing Inner Product; this follows from a rank-depth trade-off obtained in [4]; further discussed below. This makes LDT rank a very interesting measure from the lower bounds point of view.

In general, small rank in decision trees of various types points to a certain simplicity of the function, making it easy to learn in certain settings, see for instance [5, 6]. In the context of LTF primitives, the use of small rank has been particularly fruitful. It was shown in [4] that LDLs (which are just rank-1 LDTs) and constant-rank LDTs require large depth to compute the Inner Product function. In fact, a rank-depth tradeoff was established (Corollary 8 in [4]); and this implies (as also observed in [7]) that LDTs of any rank computing the Inner Product function must be of exponential ($\exp(\Omega(\sqrt{n}))$) size¹. Also in [4], a simple construction showed that polynomial-size LDLs can be transformed to polynomial-size depth-2 threshold circuits, and hence $\text{LDL} \subseteq \text{LTF} \circ \text{LTF}$. In [7], this was pushed further a bit: polynomial-size constant-rank LDTs with MAJ queries were also transformed to polynomial-size depth-2 threshold circuits.

Our main result takes this one step further by removing the polynomial-weight restriction. We show the following.

Theorem 1. *If a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is computed by an LDT of rank r and size s , then f can be computed by depth-2 threshold circuits of size $O(s \cdot n^{3r} \log^r n)$.*

¹It is not hard to see that these lower bounds, presented in [4] for Inner Product, hold for any function with no large monochromatic squares; for any such function, the depth-rank tradeoffs and size lower bounds hold. In particular, the functions $\text{MAJ} \circ \text{XOR}$, $\text{OR} \circ \text{EQ}$, $\text{SINK} \circ \text{XOR}$, are all hard for LDT.

In particular, at polynomial size, $O(1)$ -rank-LDT is contained in $\text{LTF} \circ \text{LTF}$.

In proving the above, an intermediate (between small-rank LDTs and depth-2 circuits) computation model used is decision lists where each query is a conjunction of LTFs. The arity of the conjunctions is crucial; in our construction, this is the rank of the LDT we start out with. Our method also relates such lists, with queries that are constant-arity conjunctions of LTFs, to a related model of linear decision lists with equality queries ELDLs; these are decision lists where each query is an exact linear threshold function ELTF. This is an important model of computation as it can compute, in linear size, a function that witnesses the separation between $\text{LTF} \circ \text{MAJ}$ and $\text{LTF} \circ \text{LTF}$ as shown in [2]. The relation we show is formally the following:

Theorem 2. *The following are equivalent:*

1. *ELDLs of polynomial length.*
2. *Decision lists of polynomial length, querying functions in $\text{AND}_r \circ \text{LTF}$ for some fixed $r \geq 2$.*

That r is at least 2 is crucial; at $r = 1$ the decision lists are LDLs, which, by this theorem, are contained in the class ELDL of polynomial length ELDLs, but are known to be strictly weaker (the $\text{OR}_n \circ \text{EQ}_n$ function requires size $2^{\Omega(n)}$ in LDLs and super-polynomial size even in LDTs, but can easily be computed by an ELDL of length n).

Note that ELDL is another frontier class within $\text{LTF} \circ \text{LTF}$ for which no lower bounds are currently known. By Theorem 2, obtaining lower bounds for ELDLs is no easier than, and in fact equivalent to, showing lower bounds for decision lists with $\text{AND}_2 \circ \text{LTF}$ queries.

2. Some definitions, notation, and known results

We include here some basic definitions and notation; for more details, we follow standard notation as, for instance, in [8].

A linear threshold function, denoted LTF, is a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ expressible as $f(x) = 1 \Leftrightarrow \sum_i w_i x_i \geq w_0$ for some $w_0, w_1, \dots, w_n \in \mathbb{R}$. If f is an LTF, then so is $\neg f$. An exact linear threshold function, denoted ELTF, is a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ expressible as $f(x) = 1 \Leftrightarrow \sum_i w_i x_i = w_0$ for some $w_0, w_1, \dots, w_n \in \mathbb{R}$. From a

geometric point of view, LTFs are halfspaces, and ELTFs are hyperplanes. The class of ELTF functions is not closed under negations². For every ELTF f , there are LTFs g, h such that $h \implies g$ and $f = g \wedge \neg h = g - h$. We denote the class of functions that can be written as an LTF (ELTF) as LTF (ELTF respectively).

For any function classes $\mathcal{C}_1, \mathcal{C}_2$, the function class $\mathcal{C}_1 \circ \mathcal{C}_2$ is defined as : $\mathcal{C}_1 \circ \mathcal{C}_2 = \{f(g_1, g_2, \dots, g_m) \mid f \in \mathcal{C}_1; g_1, \dots, g_m \in \mathcal{C}_2\}$. That is, these are functions computable by depth-2 circuits with a \mathcal{C}_1 gate on top and \mathcal{C}_2 gates below it. The class $\mathcal{C}_1 \circ \mathcal{C}_2 \circ \mathcal{C}_3$ is analagous to the above — depth-3 circuits with a \mathcal{C}_1 gate on top, \mathcal{C}_2 gates at the middle layer, and \mathcal{C}_3 gates at the bottom layer.

Of special interest in this note are the classes $\text{LTF}_s \circ \text{LTF}$, $\text{AND}_r \circ \text{LTF}$, $\text{OR}_k \circ \text{ELTF}$, and $\text{LTF}_m \circ \text{AND}_r \circ \text{LTF}$, where the subscript denotes the arity/in-degree of the functions/gates. (We drop the subscript where the arity is not important.)

For function class \mathcal{C} , a \mathcal{C} -decision tree is a decision tree where each query computes some function from \mathcal{C} on the inputs. The size of such a decision tree is the number of query nodes (or the number of leaves, which is just one more), the depth of the tree is the maximum number of query nodes in any root-to-leaf path, and the rank of the tree is the largest d such that a complete binary tree of depth d can be embedded in it. (Formally, the rank of a leaf is 0, and the rank of an internal node is the maximum rank of its children if they have unequal rank, and is one more than the rank of its children if they have the same rank.) Of special interest to us are LTF-decision trees, also referred to as Linear Decision Trees (LDTs).

A \mathcal{C} -decision tree of rank one is a \mathcal{C} -decision list, and the depth of the tree in this case is called the length of the list. Thus a \mathcal{C} -decision list of length ℓ has the form

If f_1 then b_1 ; elseif f_2 then b_2 ; \dots ; elseif f_ℓ then b_ℓ ; else $\neg b_\ell$,

where each f_i belongs to the class \mathcal{C} , and each $b_i \in \{0, 1\}$. For brevity, we shall often write such a decision list as simply a tuple $((f_1, b_1), \dots, (f_\ell, b_\ell))$. Of special interest to us are (all of polynomial size) LTF-decision lists denoted LDL, ELTF-decision lists denoted ELDL, and $(\text{AND}_r \circ \text{LTF})$ -decision lists for some fixed constant r .

²It is easy to show that the function that is 1 on precisely the unit vectors is an ELTF, but the negation is not.

(Throughout this article, we shall use the standard convention, see for example [9], of denoting the complexity class of functions computable *efficiently* by some computation model using sans-serif bold-face font. For example, the class of functions computable by polynomial size LDLs is denoted LDL.)

We collect some known facts that will be used in proving our results.

- Proposition 1.**
1. Any \mathcal{C} -decision list of length s can be converted to a depth-2 circuit of the form $\text{LTF}_s \circ \mathcal{C}$.
 2. An LDT of size s and rank r can be converted to an $(\text{AND}_r \circ \text{LTF})$ -decision list of length s . ([10]; see also Fact 1 in [7].)
 3. Any linear threshold function on n variables can be computed as a disjoint OR of $O(n^3 \log n)$ exact threshold functions. Thus $\text{LTF} \subseteq \text{OR}_{O(n^3 \log n)} \circ \text{ELTF}$. ([9](Theorem 7)].)
 4. $\text{AND} \circ \text{ELTF} = \text{ELTF}$; any conjunction of ELTFs is also an ELTF ([9](Proposition 6 item 2)].)
 5. An LDT of size s and depth d can be converted to an $(\text{OR}_s \circ \text{AND}_d \circ \text{LTF})$ circuit. In particular, $\text{LDT} \subseteq \text{MAJ} \circ \text{MAJ} \circ \text{MAJ}$.

Proof. Items 2,3,4 are shown in the references cited.

Item 1 is the folklore binary coding technique; $((f_1, b_1), \dots, (f_\ell, b_\ell))$ evaluates to 1 on input x if and only if $\sum_{i=1}^{\ell+1} (-1)^{1-b_i} 2^{\ell+1-i} [f_i(x) = 1?] \geq 0$, where $f_{\ell+1}$ is the constant 1 function and $b_{\ell+1} = 1 - b_\ell$.

To see item 5, observe that we can check if an input reaches any specific leaf of an LDT by a conjunction of the queried LTFs, or their negations, that appear on the path from root to that leaf. Computing an OR over all leaves labelled 1 gives an $\text{OR}_s \circ \text{AND}_d \circ \text{LTF}$ circuit. We know that $\text{OR} \circ \text{AND} \circ \text{LTF} \subseteq \text{MAJ} \circ \text{MAJ} \circ \text{LTF} \subseteq \text{MAJ} \circ \text{MAJ} \circ \text{MAJ}$ by [1]. \square

3. Proving Theorem 1

At a high level, the conversion of a small-rank LDT to a depth-2 threshold circuit proceeds in stages. We first convert the tree to a decision list using Proposition 1(2), then the list to a depth-3 circuit using Proposition 1(1). The next crucial and new step, which we describe below, is to replace each sub-circuit feeding into the top gate by a disjoint OR of ELTFs. This generalisation of Proposition 1(3), that we show below, allows us to obtain an equivalent $\text{LTF} \circ \text{ELTF}$ circuit. Expressing the ELTFs as the difference of LTFs completes the construction.

We now describe the details of the conversion.

Proposition 1(3) tells us that a function described by a halfspace can also be seen as a union of disjoint hyperplanes. The lemma below extends this to the intersection of half-spaces $\text{AND}_r \circ \text{LTF}$:

Lemma 2. *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be computed by a circuit of the form $\text{AND}_r \circ \text{LTF}_n$. Then there exists a set $\mathcal{A}_f = \{f^1, f^2, \dots, f^\ell\}$ where $\forall i, f^i \in \text{ELTF}$ and $\ell \in O(n^{3r} \log^r n)$ such that $\forall x \in \{0, 1\}^n$:*

- *If $f(x) = 0$ then $\forall i \in [\ell]$ we have $f^i(x) = 0$.*
- *If $f(x) = 1$ then $\exists i \in [\ell]$ such that $f^i(x) = 1$ and $\forall j \neq i, f^j(x) = 0$.*

Proof. We first describe the proof for $r = 2$. Let $f = g \wedge h$ where $g, h \in \text{LTF}$. Using Proposition 1(3) on g , we obtain a set $S = \{g_1, g_2, \dots, g_s\}$ of ELTF functions with $|S| \in O(n^3 \log n)$ such that g is the disjoint-OR of all the g_i . Similarly, we obtain a set $T = \{h_1, h_2, \dots, h_t\}$ of ELTF functions with $|T| \in O(n^3 \log n)$ for the function h . Define the following set:

$$\mathcal{A} = \{g_i \wedge h_j \mid g_i \in S, h_j \in T\}$$

Note that since g (h) is a disjoint-OR of the functions in S (T respectively), if $(g \wedge h)(x) = 1$, then exactly one of the functions in S and exactly one of the functions in T will evaluate to 1 on x . Hence if $f(x) = 1$, then exactly one of the functions in \mathcal{A} evaluates to 1 on x . On the other hand if $f(x) = 0$, then every function in \mathcal{A} evaluates to 0. Thus f is the disjoint-OR of the functions in \mathcal{A} . As described above, \mathcal{A} consists of $\text{AND}_2 \circ \text{ELTF}$ functions. By Proposition 1(4), each such function is in fact an ELTF function, and hence the set \mathcal{A} has all the properties stated in the lemma. Note that $|\mathcal{A}| \in O(n^6 \log^2 n)$.

In general, for $r > 2$, we take \mathcal{A} to be the cartesian product with operator \wedge of all the r sets. This will give $|\mathcal{A}| \in O(n^{3r} \log^r n)$. \square

Using Lemma 2, we are now able to transform depth-3 circuits.

Theorem 3. *For all r, s , we have*

$$\text{LTF}_s \circ \text{AND}_r \circ \text{LTF}_n \subseteq \text{LTF}_m \circ \text{LTF}_n$$

where $m \in O(s \cdot n^{3r} \log^r n)$.

Proof. Let C be an $\text{LTF}_s \circ \text{AND}_r \circ \text{LTF}_n$ circuit computing a function f . Let the top LTF gate in C be $\sum_{i=1}^s w_i g_i \geq t$ where the g_i functions are computed by $\text{AND}_r \circ \text{LTF}_n$ circuits.

For each g_i , we use Lemma 2 to obtain a set $\mathcal{A}_{g_i} = \{g_i^1, g_i^2, \dots, g_i^{\ell_i}\}$ of $\ell_i \in O(n^{3r} \log^r n)$ hyperplanes.

In the top gate of C , we replace each g_i with $\sum_{j=1}^{\ell_i} g_i^j$ to obtain a $\text{LTF}_{s'} \circ \text{ELTF}$ circuit C' , where $s' = \sum \ell_i = O(s \cdot n^{3r} \log^r n)$. Circuit C' computes the same Boolean function as C since Lemma 2 guarantees that $\forall x, g_i(x) = \sum_j g_i^j(x)$. Hence $f \in \text{LTF}_{s'} \circ \text{ELTF}_n$ via C' .

The top gate of C' is $\sum_{j=1}^{s'} w'_j f_j \geq t$ where the f_j are the ELTFs described above. Replace each f_j in this expression with $f_{j,1} - f_{j,2}$ where $f_{j,1}, f_{j,2}$ are LTFs whose difference is f_j , to get circuit C'' . Circuit C'' also computes f , and has the form $\text{LTF}_m \circ \text{LTF}_n$, where $m = 2s' \in O(s \cdot n^{3r} \log^r n)$. \square

Now we can complete the proof of our main result.

Proof. (of Theorem 1.) Let f be an n -variate Boolean function computed by an LDT T with rank r , size s . By Proposition 1(2), f is computed by a decision tree of length s with queries in $\text{AND}_r \circ \text{LTF}_n$. By Proposition 1(1), f is computed by a circuit of the form $\text{LTF}_s \circ \text{AND}_r \circ \text{LTF}_n$. By Theorem 3, f is computed by an $\text{LTF}_m \circ \text{LTF}_n$ circuit, where $m \in O(s \cdot n^{3r} \log^r n)$. \square

4. Proving Theorem 2

We first observe an easy manipulation of decision lists of a particular form.

Lemma 3. *A Boolean function f computable by a $(\text{OR}_\ell \circ \mathcal{C})$ -decision list L of length s can also be computed by a \mathcal{C} -decision list of length $s\ell$.*

Proof. Let f and L satisfy the premise. Then L is an $(\text{OR}_\ell \circ \mathcal{C})$ -decision list that has the form: $((f_1, b_1), (f_2, b_2), \dots, (f_s, b_s))$. For all $i \in [s]$, let $f_i = g_i^1 \vee \dots \vee g_i^{\ell_i}$ for some $\ell_i \leq \ell$, where each g_i^j is in \mathcal{C} . Replacing each (f_i, b_i) in L with the sub-list $((g_i^1, b_i), (g_i^2, b_i), \dots, (g_i^{\ell_i}, b_i))$ gives a \mathcal{C} -decision list computing f . \square

Now the proof of Theorem 2 is straightforward.

Proof. (of Theorem 2) Since an ELTF is expressible as the conjunction of two LTFs, ELDLs of length s can be expressed as decision lists of length s with queries in $\text{AND}_r \circ \text{LTF}$ for any $r \geq 2$.

It remains to prove that for any fixed r , decision lists of length s querying $\text{AND}_r \circ \text{LTF}$ functions can be converted to ELDLs of length polynomial in s .

Let L be an $\text{AND}_r \circ \text{LTF}$ -decision list of length s . Using Lemma 2, we can reframe each query as an $\text{OR}_\ell \circ \text{ELTF}$ function, where $\ell \in O(n^{3r} \log^r n)$. Using Lemma 3, we can strip away the outer OR and obtain an ELDL with length $O(s\ell) = O(sn^{3r} \log^r n)$. \square

Remark. We can use Theorem 2, with more careful attention to parameters, to obtain another proof of Theorem 1 as follows. LDTs of rank r can be transformed to $(\text{AND}_r \circ \text{LTF})$ -Decision lists using Proposition 1(2), then to ELDLs using Theorem 2, then to $\text{LTF} \circ \text{ELTF}$ circuits using Proposition 1(1), and finally to $\text{LTF} \circ \text{LTF}$ circuits using the fact, proved in [9], that $\text{LTF} \circ \text{LTF} = \text{LTF} \circ \text{ELTF}$.

Using Proposition 1(2) and Theorem 2, we can convert polynomial-size $O(1)$ -rank LDTs to polynomial-size ELDLs. Using Theorem 2, Proposition 1(1), and Theorem 3, we can convert polynomial-size ELDLs to polynomial-size $\text{LTF} \circ \text{LTF}$ circuits. Thus we obtain the following corollary.

Corollary 4. *The class of functions computed by polynomial-size ELDLs contains all functions computed by $O(1)$ -rank polynomial-size LDTs and is contained in the class of functions computable by polynomial-size $\text{LTF} \circ \text{LTF}$ circuits. i.e., $O(1)$ -rank-LDT \subseteq ELDL \subseteq $\text{LTF} \circ \text{LTF}$.*

Acknowledgements

We thank the reviewers for several helpful remarks about the results and presentation. We also thank the reviewers for suggesting the alternate proof of Theorem 1 that now appears as a remark after proof of Theorem 2.

References

- [1] M. Goldmann, J. Håstad, A. A. Razborov, [Majority gates vs. general weighted threshold gates](#), Computational Complexity 2 (1992) 277–300. [doi:10.1109/SCT.1992.215375](#).
URL <https://doi.org/10.1109/SCT.1992.215375>

- [2] A. Chattopadhyay, N. S. Mande, [A short list of equalities induces large sign-rank](#), SIAM J. Comput. 51 (3) (2022) 820–848. doi:10.1137/19m1271348.
URL <https://doi.org/10.1137/19m1271348>
- [3] H. D. Gröger, G. Turán, [On linear decision trees computing boolean functions](#), in: Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings, 1991, pp. 707–718. doi:10.1007/3-540-54233-7_176.
URL https://doi.org/10.1007/3-540-54233-7_176
- [4] G. Turán, F. Vatan, [Linear decision lists and partitioning algorithms for the construction of neural networks](#), in: Foundations of Computational Mathematics, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 414–423. doi:10.1007/978-3-642-60539-0_34.
URL https://doi.org/10.1007/978-3-642-60539-0_34
- [5] A. Ehrenfeucht, D. Haussler, [Learning decision trees from random examples](#), Information and Computation 82 (3) (1989) 231 – 246. doi:[https://doi.org/10.1016/0890-5401\(89\)90001-1](https://doi.org/10.1016/0890-5401(89)90001-1).
URL <http://www.sciencedirect.com/science/article/pii/0890540189900011>
- [6] A. R. Klivans, R. A. Servedio, [Learning DNF in time \$2^{\tilde{O}\(n^{1/3}\)}\$](#) , J. Comput. Syst. Sci. 68 (2) (2004) 303–318. doi:10.1016/j.jcss.2003.07.007.
URL <https://doi.org/10.1016/j.jcss.2003.07.007>
- [7] K. Uchizawa, E. Takimoto, [Lower bounds for linear decision trees with bounded weights](#), in: SOFSEM 2015: Theory and Practice of Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 412–422. doi:10.1007/978-3-662-46078-8_34.
URL https://doi.org/10.1007/978-3-662-46078-8_34
- [8] A. Chattopadhyay, M. Mahajan, N. S. Mande, N. Saurabh, [Lower bounds for linear decision lists](#), Chic. J. Theor. Comput. Sci. 2020 (2020).
URL <http://cjtcs.cs.uchicago.edu/articles/2020/1/contents.html>

- [9] K. A. Hansen, V. V. Podolskii, [Exact threshold circuits](#), in: Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010, 2010, pp. 270–279. doi:[10.1109/CCC.2010.33](https://doi.org/10.1109/CCC.2010.33).
URL <https://doi.org/10.1109/CCC.2010.33>
- [10] A. Blum, [Rank- \$r\$ decision trees are a subclass of \$r\$ -decision lists](#), Information Processing Letters 42 (4) (1992) 183–185. doi:[10.1016/0020-0190\(92\)90237-P](https://doi.org/10.1016/0020-0190(92)90237-P).
URL [https://doi.org/10.1016/0020-0190\(92\)90237-P](https://doi.org/10.1016/0020-0190(92)90237-P)