Pebbling, Entropy and Branching Program Size Lower Bounds

Balagopal Komarath*[1] and Jayalal Sarma M N[2]

1 **Department of Computer Science & Engineering, IIT Madras Chennai, India**
baluks@cse.iitm.ac.in

2 **Department of Computer Science & Engineering, IIT Madras Chennai, India**
jayalal@cse.iitm.ac.in

Abstract

We contribute to the program of proving lower bounds on the size of branching programs solving the Tree Evaluation Problem introduced in [4]. Proving an exponential lower bound for the size of the non-deterministic thrifty branching programs would separate NL from P under the thrifty hypothesis. In this context, we consider a restriction of non-deterministic thrifty branching programs called *bitwise-independence*. We show that any bitwise-independent non-deterministic thrifty branching program solving $\mathsf{BT}_2(\mathsf{h},\mathsf{k})$ must have at least $\frac{1}{2}k^{h/2}$ states. Prior to this work, lower bounds were known for general branching programs only for fixed heights $h = 2, 3, 4$ [4]. Our lower bounds are also tight (up to a factor of k), since the known[4] non-deterministic thrifty branching programs for this problem of size $O(k^{h/2+1})$ are bitwise-independent. We prove our results by associating a fractional pebbling strategy with any bitwise-independent non-deterministic thrifty branching program solving the Tree Evaluation Problem. Such a connection was not known previously even for fixed heights.

Our main technique is the entropy method introduced by Jukna and Zak[6] originally in the context of proving lower bounds for read-once branching programs. We also show that the previous lower bounds known[4] for deterministic branching programs for Tree Evaluation Problem can be obtained using this approach. Using this method, we also show tight lower bounds for any k-way deterministic branching program solving Tree Evaluation Problem when the instances are restricted to have the same group operation in all internal nodes.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Pebbling, Entropy Method, Branching Programs, Size Lower Bounds.

Digital Object Identifier 10.4230/LIPIcs.STACS.2013.622

1 Introduction

The question whether L is a proper subset of P is one of the central problems in complexity theory. One of the approaches to the problem was proposed as a program by Cook [3] by introducing a suitable computational problem, namely the solvable path systems. The program suggests to prove lower bounds for increasingly stronger models of computation solving the solvable path systems problem. Indeed, for the specific problem, the attempt is to discover the structure of that restricted variant of the underlying computation process that captures the most natural, and if possible the most general, algorithmic strategies for

* Supported by the TCS Research Fellowship.

30th Symposium on Theoretical Aspects of Computer Science (STACS'13).
Editors: Natacha Portier and Thomas Wilke; pp. 622–633

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

solving the problem. Cook [3] also proved super-logarithmic space lower bounds for marking machines solving the solvable path systems problem. Marking machines capture pebbling algorithms (which is a class of "natural" algorithms) solving this problem.

Barrington and Mckenzie [2] took a similar approach by considering the problem GEN and attempted to prove (upper and lower bounds) for increasingly stronger models of computation for solving GEN. Specifically, Barrington [2] considered "oracle" branching programs where each state of the branching program is allowed to ask a question about the input. For example, a general BP can ask queries of the form "What is the i^{th} bit of the input?" (This is called a branching program with BIT oracle.). Barrington [2] proved exponential size lower bounds for branching programs equipped only with certain "weak" oracles. Gal *et al* [5] considered incremental branching programs for solving GEN which can be thought of as branching programs trying to solve the GEN problem by incrementally finding the elements of the closure.

Cook *et al* [4] proposed the tree-evaluation problem for separating L and P and introduced thrifty branching programs as a model that captures "natural" algorithms solving the tree-evaluation problem. It is shown in [4] that deterministic thrifty branching programs exactly correspond to algorithms that implement black-pebbling. They also introduced the concept of fractional black-white pebbling and showed that non-deterministic thrifty branching programs can implement fractional black-white pebbling. It is also known that exponential size lower-bounds for deterministic semantic incremental branching programs solving the GEN problem follows from exponential size lower-bounds for deterministic thrifty branching programs solving a generalization of tree-evaluation problem called the DAG-evaluation problem [10].

Tree Evaluation Problem: We now briefly describe the tree-evaluation problem (see section 2 for a formal definition). An instance of the tree evaluation problem, $\mathsf{FT_d(h,k)}$, is a complete d-ary tree where each leaf is associated with an element from $[k]$ (which we think of as the value of the leaf node) and the i^{th} internal node is associated with a function $f_i : [k]^d \mapsto [k]$. The value of an internal node is obtained by applying this function to the values of its children. The output is the value of the root node. The corresponding Boolean version, $\mathsf{BT_d(h,k)}$, differs from $\mathsf{FT_d(h,k)}$ in that the function at the root node maps a value in $[k]^d$ to a value in $\{0,1\}$. An instance of $\mathsf{BT_d(h,k)}$ is called a "yes" instance if and only if the value of the root node is 1. Another variant of the tree-evaluation problem is the single function variant $\widehat{\mathsf{FT}}_\mathsf{d}(\mathsf{h},\mathsf{k})$ where the functions at all internal nodes are the same. A natural computational model for tree-evaluation problem is k-way branching program where each node queries the value of a single k-ary variable (i.e., the query is either i, where i is a leaf node, or $f_j(r,s)$, where j is an internal node and $r,s \in [k]$.). As observed in [4], any size lower bound of the form $\Omega(k^{r(h)})$ for k-way branching programs, where $r(h)$ is an unbounded function, would prove that $\mathsf{L} \neq \mathsf{P}$. We only consider k-way branching programs in this paper. Here we think of the parameter d as fixed and are interested in how the size of the branching programs solving $\mathsf{FT_d(h,k)}$ increases with h and k.

A natural algorithm to solve $\mathsf{FT_d(h,k)}$ is to evaluate the tree in a bottom-up fashion. This can be captured by the concept of black pebbling $\mathsf{T}_\mathsf{d}^\mathsf{h}$ (The complete d-ary tree of height h.). A black pebble on a node indicates that the value of the node is known. A black pebble can be placed on an internal node only if both its children are black pebbled. As a special case, a black pebble can be freely placed on any leaf node. It can be shown that h pebbles are necessary and sufficient for black pebbling T_2^h. Since a value in $[k]$ can be represented using $\log k$ bits. This corresponds to a size bound of $\Theta(k^h)$ for branching programs. Similarly, fractional black-white pebbling captures natural non-deterministic algorithms solving $\mathsf{FT_d(h,k)}$. A

white pebble can be freely placed on any node and corresponds to guessing the value of that node. However, to remove a white pebble from a node (this corresponds to verifying the guessed value) both its children have to be pebbled. Moreover, a branching program may compute or guess a fraction of bits of the values of nodes and this results in fractional black and white pebbles respectively.

A deterministic thrifty branching program is one in which the branching program is only allowed to query $f_j(r, s)$ when r and s are the values of the children of node j. Cook *et al.* [4] showed that deterministic thrifty branching programs solving $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$ require $\Omega(k^h)$ states by showing that such branching programs implement exactly a black pebbling strategy. Cook *et al.* [4] also proved tight lower bounds for non-deterministic thrifty branching programs for $h = 2, 3, 4$. They also show an upper-bound of $O(k^{h/2+1})$ for non-deterministic thrifty branching programs solving $\mathsf{FT}_2(\mathsf{h}, \mathsf{k})$. This shows that the non-deterministic variant is more powerful.

Our Results: In this paper, we show that computation of non-deterministic thrifty branching programs with an additional restriction that we call *bitwise-independence* can be associated with a fractional black-white pebbling sequence and therefore requires exponential size. The additional restriction of bitwise-independence is not too severe since all known upper-bounds using non-deterministic thrifty branching programs can be achieved using those with bitwise-independence property. In particular, the branching program described in [4] that achieve $O(k^{h/2+1})$ upper-bound satisfy bitwise-independence. Our main result is the first exponential lower bound for the above restriction of non-deterministic thrifty branching programs.

▶ **Theorem 1** (Main Theorem). *If B is a bitwise-independent non-deterministic thrifty branching program solving $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$, then B has at least $\frac{1}{2}k^{h/2}$ states.*

We associate these branching programs with fractional pebbling. Cook *et al.* [4] showed that if the tree $\mathsf{T}^{\mathsf{h}}_{\mathsf{d}}$ can be fractionally pebbled using p pebbles, then the corresponding (binary) Tree evaluation problem can be solved by a non-deterministic thrifty branching program of size $O(k^p)$. However, the converse direction is far from clear. We make progress in this direction and prove our lower bound by connecting bitwise-independent non-deterministic thrifty branching programs to fractional black-white pebbling sequences. We use the known result[8] (see also [4]) that $h/2 + 1$ pebbles are necessary and sufficient to pebble $\mathsf{T}^{\mathsf{h}}_2$ using fractional black-white pebbling, to derive our lower bounds. We note that the lower bounds for $h = 2, 3, 4$ in [4] were not shown by associating it with fractional black-white pebbling.

Our main technique is a method proposed by Jukna and Zak [6] for proving size lower bounds for branching programs which they call the *entropy method*. Briefly, the method is to distribute a large set of inputs among the states of the branching program such that only a small number of inputs get mapped to any particular state. To achieve this, Jukna and Zak[6] proposed to consider the set F of inputs reaching that state and show that we can uniquely determine an input in F by a decision tree of low average depth (equivalently, the set F has low entropy.). It follows that there are a large number of states.

As our next contribution, we show that the lower bound proofs in [4] for k-way branching programs solving $\mathsf{FT}^3_2(\mathsf{k})$, $\mathsf{Children}^4_2(\mathsf{k})$ and thrifty branching programs solving $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$ can be obtained using this framework. Thus we get new simplified and unified views of the proofs for the following theorems.

▶ **Theorem 2.** *Any deterministic k-way branching program solving $\mathsf{FT}^3_2(\mathsf{k})$ must have at least k^3 states.*

Any deterministic k-way branching program solving $\mathsf{Children}^4_2(\mathsf{k})$ must have at least k^4 states.

We then apply our method in a restricted setting where the functions at all internal nodes are given to be the same.

▶ **Theorem 3.** *Any deterministic k way branching program solving $\widehat{\mathsf{FT}}_2(\mathsf{h},\mathsf{k})$ with the functions at internal nodes restricted to a group operation must have at least $2^{h-2}k$ states.*

We observe that the above lower bound is tight. Indeed, when the internal operation is that of a group, the associativity property can be used to design branching programs of size $O(2^h k)$, when the function at the internal nodes is fixed. When the function at the internal nodes is also a part of the input, the upper bound is off by a factor of k, namely $O(2^h k^2)$.

The rest of the paper is organized as follows: In Section 2 we introduce the preliminaries needed for the paper. We prove the main result in section 3. Further applications of the entropy method are described in Section 4.

2 Preliminaries

For definitions of basic notions in complexity theory, we refer the reader to a standard textbook [1, 9]. We give the formal definition of the Tree Evaluation Problem first. In the following discussion, we label the nodes of the tree using usual heap numbering. The root node is labelled 1 and for each internal node i, its children are labelled $d(i-1)+2,\ldots,di+1$. We use v_i to denote the value of the i^{th} node in the input tree. When we are talking about a specific input I, we use v_i^I to denote the value of node i of the input I.

We now define the function and Boolean versions of the tree-evaluation problem.

▶ **Definition 4.** (Tree Evaluation Problems [4]) Input: The tree $\mathsf{T}^{\mathsf{h}}_{\mathsf{d}}$ where each leaf node is associated with a value from $[k]$ and each internal node i is associated with a function $f_i : [k]^d \mapsto [k]$, where $d, h, k \geq 2$

Output for $\mathsf{FT}_{\mathsf{d}}(\mathsf{h},\mathsf{k})$: The value $v_1 \in [k]$ of the root node, where in general $v_i = a$ if i is a leaf and a is the value associated with i^{th} node in the input and $v_i = f_i(v_{j_1},\ldots,v_{j_d})$ if i is a non-leaf node with nodes $j_1,\ldots,j_d$ as children.

Output for $\mathsf{BT}_{\mathsf{d}}(\mathsf{h},\mathsf{k})$: The value $v_1 \in \{0,1\}$ of the root node. The evaluation rules are the same as for $\mathsf{FT}_{\mathsf{d}}(\mathsf{h},\mathsf{k})$.

It is known that tree-evaluation problems are in LOGDCFL [4] (For definition of LOGDCFL see [7].). Since we think of the parameter d as a constant, the input size is $O(d^h k^2 \log k)$. Since all the values in the definition of tree-evaluation problems are k-ary, a general model to solve tree-evaluation problem is a branching program that queries k-ary variables. Such branching programs are called k-way branching programs, since each query has k possible outcomes (depending on the value of the queried variable.). We define these models formally now. Throughout the technical sections of this paper, we use BP as short form for branching program.

▶ **Definition 5** (k-way BPs[4])**.** A non-deterministic k-way BP B for $\mathsf{FT}_{\mathsf{d}}(\mathsf{h},\mathsf{k})$ is a directed rooted multigraph. It consists of k final states labelled $1,\ldots,k$. All other states of the BP are query states. A query state is labelled either i where i is a leaf node or labelled $f_i(x_1,\ldots,x_d)$ where i is an internal node, $x_1,\ldots,x_d \in [k]$, and each outgoing edge is labelled by an element from $[k]$. A computation path on input x is a directed path from the root (the start state) and each edge in the path is consistent with x. At least one such computation must end in a

final state. The BP B is deterministic if and only if each query state has exactly k outgoing edges labelled $1, \ldots, k$.

A non-deterministic k-way BP B for $\mathsf{BT_d(h,k)}$ is defined similarly except that each query state labelled $f_1(x_1, \ldots, x_d)$ where $x_1, \ldots, x_d \in [k]$ has all of its outgoing edges labelled by either 0 or 1. There are two final states labelled 0 and 1. The BP B is deterministic if and only if each query state labelled $f_1(x_1, \ldots, x_d)$ has exactly two outgoing edges labelled 0 and 1 and every other query state has exactly k outgoing edges labelled $1, \ldots, k$. The BP B solves $\mathsf{BT_d(h,k)}$ if and only if for every "yes" instance has at least one accepting computation path and every "no" instance has no accepting computation path.

By a sub-BP B' of B obtained by restricting input set E to E', we refer to the BP obtained from B by removing edges not used by inputs in E' and by shortcutting states for which only one outgoing edge can be active when we consider computation on instances in E'.

The size of binary binary for solving tree-evaluation problems differ from the size of k-way BPs by a factor of at most k. Therefore, a size lower bound of $\Omega(k^{r(h)})$ for k-way branching programs, where $r(h)$ is an unbounded function, would separate L from $\mathsf{LOGDCFL}$.

▶ **Definition 6** (Thrifty BPs [4]). A non-deterministic BP solving $\mathsf{BT_d(h,k)}$ is called *thrifty* if and only if for any accepting computation path on instance I any query state labelled $f_i(x_{d(i-1)+2}, \ldots, x_{di+1})$ satisfies $x_j = v_j^I$ for $d(i-1)+2 \leq j \leq di+1$ (i.e., the internal nodes are queried only at the correct values of its children.).

2.1 Pebbling

Pebbling sequences capture "natural" algorithms that solve the tree-evaluation problems that evaluate the values at nodes of the tree in a bottom-up fashion. A black pebble value at a node indicates the fraction of the value at the node that is known to the BP. Similarly, a white pebble indicates the fraction of the value at the node guessed by the BP (respectively, the fraction of value at the node that needs to be verified by the BP). For example, a black pebble value of 1 indicates that value is completely known and a white pebble value of 1 indicates that the value was guessed from $[k]$. In order to compute or guess (a fraction of) the value at any node, the BP must completely figure out (by computing or guessing) the values of its children.

▶ **Definition 7** (Fractional Black-White Pebbling [4]). A fractional pebbling configuration on a rooted d-ary tree T is an assignment of a pair of real numbers $(b(i), w(i))$ to each node i of the tree. The values $b(i)$ and $w(i)$ are called the black and white pebble values, respectively, of node i. We have for every i

$$\begin{aligned} b(i) + w(i) &\leq 1 \\ 0 \leq b(i), w(i) &\leq 1 \end{aligned} \tag{1}$$

The legal pebble moves are as follows.

1. For any node i, decrease $b(i)$ arbitrarily.
2. For any node i, increase $w(i)$ arbitrarily.
3. For any node i, if each child of i has pebble value 1, then decrease $w(i)$ to 0.
4. For any node i, if each child of i has pebble value 1, then increase $b(i)$ arbitrarily and simultaneously decrease the black pebble value of children of i.

The number of pebbles in a configuration is the sum over all nodes i of $b(i) + w(i)$. A fractional pebbling of T using p pebbles is a sequence of (legal) fractional pebbling moves on nodes of T which starts and ends with each node having pebble value 0 and at some point the root node has black pebble value 1, and no configuration has more than p pebbles.

A black pebbling is a fractional black-white pebbling such that for all i the black pebble value $b(i)$ always takes values from $\{0, 1\}$ and $w(i) = 0$.

It is known that $h/2+1$ pebbles are necessary and sufficient to pebble $\mathsf{T}_2^{\mathsf{h}}$ using fractional black-white pebbling [8].

2.2 Entropy Method

We now formally describe the entropy method introduced in [6]. We specialize the definition slightly to suit our application of the method. Let B be a BP computing the characteristic function of language L. Let A be a particular set of inputs. Define a "distribution" function $g : A \mapsto States(B)$. Now consider an arbitrary state s in the range of g and let $F = g^{-1}(s)$. Define a decision tree D such that each $a \in F$ reaches a unique leaf in D. Such a decision tree is called a 'splitting tree' for F in [6]. The next step is to prove that D has low depth which will imply that the entropy of F, $h(F) = \log |F|$, is small. Then we have $Size(B) \geq 2^{|A|-h(F)}$. In defining A and g, we may use properties of L and any restrictions imposed on the structure of B. The goal is to minimize the maximum value of $h(F)$ over all choices of F by using an A that is as large as possible.

For the rest of our discussion, we fix $d = 2$. However all our arguments can be easily generalized to arbitrary constant d.

2.3 Bitwise-independent Non-deterministic Thrifty BPs

We use N to denote the total number of non-root nodes in $\mathsf{T}_2^{\mathsf{h}}$. Let B be a non-deterministic thrifty BP for $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$. Let s be a state of B. We define

$$F_s = \{(v_2^I, \ldots, v_{N+1}^I) \colon \exists I \text{ and a computation path } C(I) \text{ such that } s \in C(I)\}$$
$$A_s = \{(v_2^I, \ldots, v_{N+1}^I) \colon \exists I \text{ and an accepting computation path } C(I) \text{ such that } s \in C(I)\}$$

We use $\pi(S, i)$ to denote the set of all i^{th} component of the tuples in S (typically, S is either F_s or A_s for some s.). That is, the set formed by projecting the i^{th} component of all tuples in S. For any encoding function $\phi : [k] \mapsto \{0, 1\}^{\lceil \log k \rceil}$, we use $(r)_i$ to denote the i^{th} bit of $r \in [k]$ when r is encoded using ϕ.

▶ **Definition 8** (Bitwise-independent Non-deterministic Thrifty BPs)**.** Let $k = 2^\ell$ and let B be a non-deterministic thrifty BP solving $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$. Then B is bitwise-independent if and only if there exists an encoding function $\phi : [k] \mapsto \{0, 1\}^\ell$ such that for every state s in B the following two conditions are satisfied.

$$F_s = \bigtimes_{i=2}^{N+1} \phi^{-1}\left(\bigtimes_{j=1}^{\ell} (\pi(F_s, i))_j\right)$$
$$A_s = \bigtimes_{i=2}^{N+1} \phi^{-1}\left(\bigtimes_{j=1}^{\ell} (\pi(A_s, i))_j\right)$$

Where we think of the first $\times$ as the normal Cartesian product and the second one (over all the bits) as concatenating all the bits after forming the Cartesian product. When k is not

a power of two, we consider the largest power of two smaller than k. Let this be 2^ℓ. Then B is bitwise-independent if and only if the above two conditions are satisfied with equality replaced by superset.

The intuition is that at any state the bits of values of non-root nodes can be partitioned into "fixed" bits and "unfixed" bits and the sets F_s and A_s are such that all possible combinations of unfixed bits are in the set. i.e., the BP cannot store implicit information about bits (such as, the second bit is the complement of the first bit).

If we only consider minimal bitwise-independent non-deterministic thrifty BPs, then we have $|F_s|, |A_s| \geq 1$ for any query state s. This is because any query state s that does not have any accepting computation path passing through it can be merged with the reject state. Also note that by the definition of bitwise independence, for any i and s, we have $\pi(F_s, i)$ and $\pi(A_s, i)$ are always powers of two when k is a power of two.

We now define the pebbling values assigned to a non-root node i at state s of the BP. These pebbling values are referred to as "actual" pebble values.

$$\begin{aligned} b(i,s) &= 1 - \log_k |\pi(F_s, i)| \\ w(i,s) &= \log_k \frac{|\pi(F_s, i)|}{|\pi(A_s, i)|} \end{aligned} \tag{2}$$

3 Lower Bounds for Bitwise-independent Non-deterministic Thrifty BPs

In this section, we prove exponential size lower bounds for bitwise-independent non-deterministic thrifty BPs. The proof uses the entropy method. We consider the input set E for the problem $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$, where each leaf node is allowed to take values from $[k]$ and for each instance I and each internal node i, we allow $f_i(v^I_{2i}, v^I_{2i+1})$ to take any value from $[k]$ and restrict it to 1 elsewhere. We also fix $f_1(v^I_2, v^I_3) = 1$ and 0 elsewhere so that $|E| = k^N$. Note that all instance $I \in E$ are "yes" instances. The idea is to map an accepting computation path on $I \in E$ into a valid fractional black-white pebbling sequence. To this end, we define a set of critical states for each node on an accepting computation path of $I \in E$. The distribution function then maps each input to the state where the pebble value is maximum (which will imply that the number of inputs is small).

We now show that our definition of pebble values satisfy the restrictions imposed on pebble values by (1).

▶ **Claim 9.** *For any non-root node i and state s, $0 \leq b(i,s), w(i,s) \leq 1$.*

▶ **Claim 10.** *For any non-root node i and state s, $b(i,s) + w(i,s) \leq 1$.*

The following claim establishes the fact that if the total pebble value of the tree (in non-root nodes) is high at a state, then there are only a few inputs on an accepting computation path reaching that state. In other words, if the pebble value at a point of the computation is high, then the entropy at that point is low.

▶ **Claim 11.** *If the total pebble value of the non-root nodes of the tree at a state s is p, then the number of "yes" instances reaching s on an accepting computation path is k^{N-p}.*

Proof. Consider a particular non-root node i. Assume that the total pebble value at i is p_i. From this we have $1 - \log_k |\pi(F_s, i)| + \log_k \frac{|\pi(F_s,i)|}{|\pi(A_s,i)|} = p_i$. Therefore $|\pi(A_s, i)| = k^{1-p_i}$. Now by simple counting the total number of inputs on an accepting computation path is $k^{\sum_{i=2}^{N+1}(1-p_i)} = k^{N-p}$. ◀

Consider an input $I \in E$ and an accepting computation path $C(I)$ for I. Our aim is to define a sequence of critical states $(\gamma_0 =)s_0, s_1, \ldots, s_{t+1}(= acc)$ (s_0 and s_{t+1} are always start and final states of the BP) and associate a fractional pebbling configuration with each critical state. The sequence thus obtained will be a valid fractional pebbling sequence and it will be defined such that the pebbling values at each node will underestimate the actual pebbling values defined by (2).

Critical States

We now define the sequence of critical states on the accepting computation path $C(I)$. The critical state for the root node is the last state querying the root node. Every non-root node j may have multiple critical states. Let s denote *a* critical state of parent of j. If $b(j, s) > 0$, then the last node querying node j before s is a critical state for j. If $w(j, s) > 0$, then the first node querying node j after s is a critical state for j.

Pebble Configurations

We now define the sequence of pebble configurations associated with critical states on an accepting computation path of input I. The black pebble value of the root node becomes 1 immediately after its critical state and remains so until the end of the computation where it becomes 0. Now we define the pebble values of an arbitrary non-root node j. Let s' be a critical state for j', the parent of j. If $b = b(j, s') > 0$, then this black pebble value must have increased from 0 to b at some point of computation. Now consider the critical state s for j before s' as defined before. The black pebble value of node j is increased from 0 to b at the critical state immediately following s. Similarly, if $w = w(j, s') > 0$, then this white pebble value must decrease from w to 0 at some point of computation. Now consider the critical state s for j after s' as defined before. The white pebble value is reduced from w to 0 from the critical state immediately following s.

The following claims about the validity of the starting and ending pebbling configurations are easily proved.

▶ **Claim 12.** *The start state has an empty pebbling configuration.*

▶ **Claim 13.** *The accepting state has an empty pebbling configuration.*

The following lemmas establish the fact that the pebbling sequence defined above is a valid pebbling sequence.

▶ **Lemma 14.** *Let s be a critical state for node j, then both of j's children are fully pebbled at s.*

Proof. Let s query $f_j(u, v)$. We have $\pi(A_s, 2j) = \{u\}$ (and $\pi(A_s, 2j + 1) = \{v\}$) by the thrifty property. Then $b(2j, s) + w(2j, s) = 1 - \log_k |\pi(F_s, 2j)| + \log_k \frac{|\pi(F_s,2j)|}{|\pi(A_s,2j)|} = 1$ (and similarly for $2j + 1$). ◀

▶ **Lemma 15.** *If the black pebble value of node j is increased or the white pebble value of node j is decreased at state s, then both its children are fully pebbled at the critical state immediately before s.*

Proof. For a node j, the black pebble value is increased or the white pebble value is decreased only at the critical state immediately following a critical state for j. By Lemma 14 both children of node j are fully pebbled at this critical state. ◀

The following is our key technical lemma and establishes the fact that the pebbling values defined for the critical states never overestimate the actual pebbling values of nodes.

▶ **Lemma 16.** *Let b and w be the pebble values defined at state s for an arbitrary non-root node $2j$, then $b \leq b(2j, s)$ and $w \leq w(2j, s)$.*

Proof. The proof is divided into two parts. First, we show that the black pebble values are never overestimated. Then we show that white pebble values are never overestimated.

We consider an arbitrary state s at which the black pebble value of node $2j$ is defined as $b > 0$. Note that the black pebble value of a non-root node $2j$ is non-zero if and only if there exists a critical state for the parent of $2j$ at which the actual pebble value of $2j$ is b. Therefore, there exists a state s_{2j} that is a critical state for $2j$ before s and s_j that is a critical state for j, the parent of $2j$, after s (with $s = s_j$ possibly.). Now suppose that the actual black pebble value for node $2j$ at state s is $b(2j, s)$ and that $b(2j, s) < b$.

$$\begin{aligned} 1 - \log_k |\pi(F_s, 2j)| &< b \\ \implies |\pi(F_s, 2j)| &> k^{1-b} \end{aligned}$$

Now by the independence assumption we may conclude that there are more than k^{1-b} inputs that differ only at the value of node $2j$. By the definition of critical states, there does not exist any node querying $2j$ in $C(I)$ from s to s_j. All these inputs can follow the same path to the critical state s_j. Therefore, the black pebble value is $b(2j, s_j) < b$, a contradiction.

It remains to prove that white pebble values are never overestimated. We will prove that the white pebble value of a node $2j$ is at least the estimated value w between all states from s_j to s_{2j} (both inclusive). Here s_j is a critical state for j at which node $2j$ acquired a white pebble value of w and s_{2j} is the critical state for $2j$ after which this pebble value is removed. In order to prove this, it is sufficient to prove that the ratio $\frac{f'}{a'} = \frac{|\pi(F_{s'}, 2j)|}{|\pi(A_{s'}, 2j)|}$ for any state s' is greater than the corresponding ratio $\frac{f}{a}$ at state s_j, where s' is a state on $C(I)$ in the segment from s_j to s_{2j}. By the independence argument, we have $f' \geq f$ by taking projections of all f inputs that differ from I only at node $2j$. We will show that if $a' > a$, then $f' > f$ by an appropriate amount so that the ratio is not reduced.

Since the white pebble value is acquired at state s_j, we have $w(2j, s_j) = w$. Now consider a state s' (Possibly equal to s_{2j}) on the segment of the computation path $C(I)$ between s_j and s_{2j}. Our aim is to prove that $w \leq w(2j, s')$. Let $f = |\pi(F_{s_j}, 2j)|$, $f' = |\pi(F_{s'}, 2j)|$, $a = |\pi(A_{s_j}, 2j)|$ and $a' = |\pi(A_{s'}, 2j)|$. First of all note that $f' \geq f$ since there are no nodes querying $2j$ from s_j to s_{2j} and the independence property guarantees f inputs that differ from I only at node $2j$ will reach s'. Now we will show that whenever $a' > a$, f' is greater than f by the same multiplicative factor. Note that both f and a are powers of two. By the assumption of bit-wise independence, we can partition bits of node $2j$ into "fixed" bits and "unfixed" bits for any F_s (and A_s). The only way to add elements to these sets are by unfixing bits. Let us assume that exactly one more bit became unfixed in $\pi(A_{s'}, 2j)$. So $a' = 2a$.

Let r' be a value in $\pi(A_{s'}, 2j) \setminus \pi(A_{s_j}, 2j)$. We claim that $r' \notin \pi(F_{s_j}, 2j)$. We will prove this by contradiction. Suppose $r' \in \pi(F_{s_j}, 2j)$, then by the independence property there is an input J which is the same as I except that $v^J_{2j} = r'$ reaches s' through s_j. Since $r' \in \pi(A_{s'}, 2j)$, there is an accepting path for J through s_j. This accepting path is obtained by using the independence property of $A_{s'}$ and the fact that an accepting computation for I passes through s'. But this path makes a non-thrifty query at s_j. Therefore $r' \notin \pi(F_{s_j}, 2j)$

as claimed. Since $r' \in \pi(F_{s'}, 2j)$, at least one bit must have become unfixed. But this implies $f' \geq 2f$. This proof can be easily extended to the case where $a' = 2^m a$ for any m. ◀

We now prove our main result by associating an accepting computation on input $I \in E$ to a valid fractional black-white pebbling sequence.

▶ **Theorem 17.** *If B is a bitwise-independent non-deterministic thrifty BP solving $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$, then B has at least $\frac{1}{2}k^{h/2}$ states.*

Proof. Assume that k is a power of two. We now apply the entropy method described in subsection 2.2. Our input set is the set E described previously. We now describe our distribution function h. Recall that each instance I in E is a "yes" instance and therefore guaranteed to have an accepting computation path $C(I)$ in B. As we have already seen, we can identify a sequence of critical states in $C(I)$ and associate a fractional black-white pebbling configuration with each critical state such that the sequence of fractional black-white pebbling configurations form a valid fractional pebbling of $\mathsf{T}_2^{\mathsf{h}}$ (See Claims 9, 10, 12, 13, 14, and 15). But we know that any valid fractional black-white pebbling of $\mathsf{T}_2^{\mathsf{h}}$ must have a configuration with at least $h/2$ pebbles on non-root nodes [8]. Let s be the critical state in $C(I)$ that corresponds to this configuration. Our distribution function h maps I to s. Now consider an arbitrary state s in $range(h)$ and consider the set $G_s = h^{-1}(s)$. By Claim 11, we have $|G_s| \leq k^{N-h/2}$. It follows that B has at least $k^{h/2}$ states. (Intuitively, if we consider the splitting tree for G_s, we can determine the input by querying only those bits that are "unfixed" at s.)

When k is not a power of two, we consider the highest power of two (2^ℓ) smaller than k. Consider the sub-BP of B that solves $\mathsf{BT}_2(\mathsf{h}, \mathsf{k})$ when the values are from the set $[2^\ell]$. By definition of bitwise-independence and the lower bound when k is a power of two, we have that this sub-BP of B has at least $2^{\ell^{h/2}} > \frac{1}{2}k^{h/2}$ states. ◀

▶ Remark. We note that the lower-bound proof in [4] for deterministic thrifty BPs can be obtained by specializing our argument to deterministic thrifty BPs. Specifically, we define the black pebble value of a node as 1 if and only if its value is known. The critical state for the root node is the last state querying root and critical state for other nodes j are those states which query j and immediately precedes the critical state for j's parent. The fact that the computation follows a valid black pebbling can be argued using thriftiness (bitwise-independence is not required.). We then map each input to the state that has h or more pebbles. The lower bound follows.

4 Lower Bounds for Deterministic BPs Using Entropy Method

In this section, we show that many lower-bound proofs in [4] can be derived using the entropy method and derive some new applications of the method. We believe that reformulating it in terms of the entropy method makes the connection more explicit.

▶ **Theorem 18.** *([4]) Any deterministic k-way BP solving $\mathsf{FT}_2^3(\mathsf{k})$ has at least k^3 states.*

Proof. First, we will consider a k-way BP that takes two inputs $u, v \in [k]$ and computes $u +_k v$ where $+_k$ is addition modulo k (appropriately defined on $[k]$.). We will prove a size lower-bound of k states for this problem. Then we will use this result to prove the theorem.

Let B be a k-way BP solving the above problem. We apply the entropy method to prove the required size lower-bound. Our input set A consists of k^2 inputs (all inputs). Our distribution function maps each input in A to the last edge in the k-way BP B solving this

problem. Now consider an arbitrary edge e labelled r and connecting a state labelled (w.l.g.) u to the output state s. Now consider the set of inputs F_e reaching this edge. The only possible inputs are those with $u = r$ and $u +_k v = s$. But this implies that $v = s -_k r$. Therefore $F_e = \{(r, s -_k r)\}$ has cardinality one. Since the choice of e was arbitrary, we have $Edges(B) \geq k^2/1 = k^2$. Since we are considering k-way BPs where each state has exactly k outgoing edges $States(B) \geq k$.

Consider the sub-problem of $\mathsf{FT}^3_2(\mathsf{k})$ where $f_1 = +_k$, leaves are allowed to take arbitrary values, and for any input I, we allow $f^I_j(v^I_{2j}, v^I_{2j+1})$ for $j = 2, 3$ to take arbitrary values and restrict it to 1 elsewhere. Consider a k-way BP B solving this problem. Now consider the sub-BP b' obtained from B by fixing $(v_4, v_5) = (v_6, v_7) = (r, s)$ for some $r, s \in [k]$. Note that the sub-BP B' computes $u +_k v$ for $u = f_2(r, s)$ and $v = f_3(r, s)$ and therefore must have at least k states. Now the set of all states querying f_2 or f_3 in B is the disjoint union of all states querying $f_2(r, s)$ and $f_3(r, s)$ for k^2 (r, s) pairs. Therefore $States(B) \geq k^3$ as claimed. ◀

The $\mathsf{Children}^4_2(\mathsf{k})$ problem is the same as $\mathsf{FT}^4_2(\mathsf{k})$ problem except that the tree has no root node and the values at nodes 2 and 3 together is defined as the output.

▶ **Theorem 19.** *([4]) Any deterministic k-way BP solving $\mathsf{Children}^4_2(\mathsf{k})$ has at least k^4 states.*

Proof. Consider a k-way BP that takes four inputs u, v, w, x and computes the tuple $(u +_k v, w +_k x)$. We will prove a size lower-bound of k^2 states for this problem and argue that the theorem follows from this result.

Let B be a deterministic k-way BP solving this problem. We now apply the entropy method. Our input set A is the set of all inputs and therefore $|A| = k^4$. Our distribution function will map each input in A to the last edge in its computation path on B. Consider an arbitrary edge e labelled r that connects a query state labelled u to the output state (s, t). Now consider the set of inputs F_e that get mapped to e. We have $u = r$, $v = s -_k r$, and $w +_k x = t$. Since there are exactly k inputs that satisfy these conditions $|F_e| \leq k$. Therefore $Edges(B) \geq k^4/k = k^3$ and it follows that $States(B) \geq k^2$.

Consider the sub-problem of $\mathsf{Children}^4_2(\mathsf{k})$ where $f_2 = f_3 = +_k$, leaves are allowed to take arbitrary values, and for any input I, we allow $f^I_j(v^I_{2j}, v^I_{2j+1})$ for $j = 4, 5, 6, 7$ to take arbitrary values and restrict it to 1 elsewhere. Consider a k-way BP B solving this problem. Now consider the sub-BP B' obtained from B by fixing values of sibling leaves to (r, s). Note that the sub-BP B' solves the problem discussed in the previous paragraph and hence requires k^2 states. As before, since the level 2 query states of B are the disjoint union of query states for k^2 distinct (r, s) pairs, we have $States(B) \geq k^4$. ◀

We now present a new lower-bound of $\Omega(2^h k)$ for $\widehat{\mathsf{FT}}_2(\mathsf{h}, \mathsf{k})$ problem when the function at internal nodes are restricted to a group operation. This forms a special case of the general problem.

▶ **Theorem 20.** *Any deterministic k way BP solving $\widehat{\mathsf{FT}}_2(\mathsf{h}, \mathsf{k})$ with the functions at internal nodes restricted to a group operation has at least $2^{h-2}k$ states.*

Proof. Assume without loss of generality that the functions at internal nodes are $+_k$. The leaf nodes are labelled $x_1 = 2^{h-1}, \ldots, x_{2^{h-1}} = 2^h - 1$. Let B be a deterministic k-way BP solving this problem. Now consider the sub-BP B' obtained from B by fixing $x_3, \ldots, x_{2^{h-1}}$ to 1. The sub-BP B' computes $x_1 +_k x_2$ and therefore has at least k states. A similar argument can be applied to each pair of leaves. Since there are 2^{h-2} disjoint pairs of leaves, the BP B must have at least $2^{h-2}k$ states. ◀

Upper Bounds: We observe upper bounds for the size of branching programs computing $\widehat{\mathsf{FT}}_2(\mathsf{h},\mathsf{k})$ problem when the function at internal nodes are restricted to a group operation. The associativity of the group operation implies upper bounds. We now briefly describe a BP for this problem. The BP is a layered BP of width k. The BP evaluates the group product in a left-associative fashion. In order to do this, the BP only has to remember the value of the product $v_1 \dots v_{i-1}$ in the i^{th} layer. This value is in $[k]$ and can be remembered using width k. Then, in the i^{th} layer, the BP reads v_i and moves to $i + 1^{\text{st}}$ layer updating the remembered value as required. There are two variations possible in this setting. In the first one, the function at the internal nodes is fixed. In this case the branching program described will be of size $2^h k$ and hence Theorem 20 is tight. In the second version, when the function at the internal node is also a part of the input, the described method will give an upper bound of $2^h k^2$ (since we also have to query the function values).

References

1. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
2. David A.Mix Barrington and Pierre McKenzie. Oracle Branching Programs and Logspace versus P. *Information and Computation*, 95(1):96 – 115, 1991.
3. Stephen A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9(3):308–316, 1974.
4. Stephen A. Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and Branching Programs for Tree Evaluation. *ACM Transactions on Computation Theory (TOCT)*, 3(2):4:1–4:43, 2012.
5. Anna Gal, Michal Koucky, and Pierre McKenzie. Incremental Branching Programs. *Theory of Computing Systems*, 43(2):159–184, April 2008.
6. S. Jukna and S. Žák. On Uncertainty versus Size in Branching Programs. *Theoretical Computer Science*, 290(3):1851–1867, January 2003.
7. I. H. Sudborough. On the Tape Complexity of Deterministic Context-free Languages. *Journal of ACM*, 25(3):405–414, July 1978.
8. Frank Vanderzwet. Fractional Pebbling Game Lower Bounds, December 2011. http://www.cs.toronto.edu/ fvan/mt11.pdf.
9. H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer New York Inc., 1999.
10. Dustin Wehr. Lower bound for Deterministic Semantic-incremental Branching Programs Solving GEN. *CoRR*, abs/1101.2705, 2011.