Electronic Colloquium on Computational Complexity, Report No. 61 (2003)

ECCC

Free Binary Decision Diagrams for Computation of EAR_n

Jan Kára Daniel Král'*

Abstract

Free binary decision diagrams (FBDDs) are graph–based data structures representing Boolean functions with a constraint (additional to binary decision diagrams) that each variable is tested at most once during the computation. The function EAR_n is the following Boolean function defined for $n \times n$ Boolean matrices: $\mathrm{EAR}_n(M) = 1$ iff the matrix M contains two equal adjacent rows. We prove that each FBDD computing EAR_n must have size at least $2^{0.63 \log_2^2 n - O(\log n \log\log n)}$ and we present a construction of such diagrams of size $2^{1.89 \log_2^2 n + O(\log n)}$.

1 Introduction

Graph–based data structures representing Boolean functions are important both from the practical (verification of circuits) and from the theoretical (combinatorial properties of Boolean functions) point of view. The sizes of minimal representations of different Boolean functions in a certain class of data structures and the relation between the sizes in different classes are intensively studied. We refer the reader to a recent monograph [7] on the topic by Wegener. In this paper, we show that the optimal size of representation of a certain natural matrix-based Boolean function is neither polynomial nor exponential.

A binary decision diagram (BDD) is a directed graph where vertices are labelled with input variables and each vertex except for so-called sinks has outdegree two. The two outgoing arcs at each vertex are labelled with values 0 and 1 (0 is understood to be false and 1 to be true). The computation is started in a special vertex called a source and guided in the natural way by the input to one of the two special vertices called sinks — one of them is an accepting sink (the 1-sink) and the other one is a rejecting sink (the 0-sink). We refer the reader to a more formal definition in the next section. So-called free binary decision diagrams (FBDDs) are studied in this paper. These are just BDDs with an additional constraint that each variable is tested at most once during

*Department of Applied Mathematics and Institute for Theoretical Computer Science, Charles University, Malostranské náměstí 25, 118 00 Prague 1, Czech Republic. E-mail: {kara,kral}@kam.mff.cuni.cz. Institute for Theoretical Computer Science (ITI) is supported by Ministry of Education of Czech Republic as project LN00A056.

ISSN 1433-8092

the computation. FBDDs were introduced by Masek in [4] (he called them read–once branching programs) already in 1976. Lots of upper and lower bounds on the sizes of FBDDs have been proved since: The first exponential lower bound was proved in [8, 9] and further ones were proved later, e.g., [1, 3, 5, 6].

The function EAR_n is defined on $n \times n$ Boolean matrices as follows: The value of $\mathrm{EAR}_n(M)$ is 1 iff M contains two adjacent equal rows, i.e. if there exists $1 \leq i_0 < n$ such that $M[i_0, j] = M[i_0 + 1, j]$ for all $1 \leq j \leq n$ (throughout the paper, following the usual notation, the first coordinates always correspond to the rows of the matrix). The problem to decide whether the function EAR_n has FBDDs of a polynomial size was mentioned as an open problem in [7] (Problem 6.17). We prove that the size of optimal FBDDs for the function $\mathrm{EAR}_n(M)$ is $2^{\Theta(log^2 n)}$, namely it is between $2^{0.63 \log^2 n - O(\log n \log\log n)}$ and $2^{1.89 \log^2 n + O(\log n)}$ (in the whole paper, the bases of all logarithms are equal to 2). This settles the original problem of Wegener. The interest in the size of FBDDs for EAR_n is amplified by the fact that the size of its optimal FBDDs is neither polynomial nor exponential.

The paper is based on a conference paper [2] of the authors. The conference paper [2] contains a proof that the size of optimal FBDDs for the function EAR_n is $2^{\Theta(log^2 n)}$ without computing the multiplicative constants of the factors $\log^2 n$ in the exponents. If the constants are computed, the lower bound on the size of FBDDs computing EAR_n presented in [2] is $2^{\log^2 n/(2 \log 3) - O(\log n)} \approx 2^{0.315 \log^2 n - O(\log n)}$ and the upper bound is $2^{2.5 \log^2 n + O(\log n)}$. Thus both the bounds have been refined compared to the conference version of the paper.

The paper is structured as follows: We recall basic definitions related to (free) binary decision diagrams in Section 2. Next, in Section 3, the upper bound of $2^{1.89 \log^2 n + O(\log n)}$ on the size of FBDDs computing the function EAR_n is proved (Theorem 1). The asymptotically matching lower bound on the size of such FBDDs, the bound of $2^{0.63 \log^2 n - O(\log n \log\log n)}$, is proved in Section 4 (Theorem 3). We also state a little more general version of Theorem 3, namely Theorem 2, which is used to deduce some lower bounds for modifications of the problem in the final Section 5 where we also pose several related open problems.

2 Definitions and Notation

A *binary decision diagram (BDD, branching program)* $\mathcal{B}$ is an acyclic directed graph with three special vertices: a *source*, a 0*–sink* and a 1*–sink*. The vertices of $\mathcal{B}$ are called *nodes*. Each node except for the sinks has out–degree exactly two and it is assigned one of the input variables: One of the two arcs leading from such a node is labelled with 0 and the other with 1. The out–degrees of the two sinks are zero. The *size* of a BDD is the number of its nodes.

The *computation path* in $\mathcal{B}$ for the input $x_1, \ldots, x_n$ is the (unique) path $v_0, \ldots, v_k$ from the source v_0 to a sink v_k with the following property: If the node $v_i, 0 \leq i \leq k-1$, is assigned a variable x_j, then v_{i+1} is the unique node to which an arc labelled with the value of x_j leads from v_i to. The value of the

function $f_{\mathcal{B}}(x_1, \ldots, x_n)$ is equal to 1 (true) if the last node of the computation path for $x_1, \ldots, x_n$ is the 1–sink. We also say that $\mathcal{B}$ *accepts* the input $x_1, \ldots, x_n$. Otherwise, the function is equal to 0 (false) and we say that $\mathcal{B}$ rejects the input. The function $f_{\mathcal{B}}$ is *computed* by $\mathcal{B}$ and the diagram $\mathcal{B}$ *represents* the function $f_{\mathcal{B}}$. We say that $\mathcal{B}$ is *reduced* if each its node is on a computation path for some choice of values of the input variables and there are no parallel arcs in $\mathcal{B}$. It is straightforward (cf. [7]) to prove that for each binary decision diagram there exists one which is reduced and which computes the same function. We say that a binary decision diagram $\mathcal{B}$ is a *free binary decision diagram (FBDD, read–once branching program)* if for any choice of values of the input variables, the computation path for them does not contain two vertices with the same variable assigned, i.e., during the computation each variable is tested at most once.

Fix a (free) binary decision diagram $\mathcal{B}$, input variables $x_1, \ldots, x_n$ and the node v of the computation path for $x_1, \ldots, x_n$ in $\mathcal{B}$. The variable assigned to v is said to be *scanned* at the node v and the variables assigned to the nodes before v on the path are said to *have been scanned* by the computation of the diagram $\mathcal{B}$.

3 Upper Bound

Fix a size n of the input matrix throughout this section. We design a recursive procedure `test(row1,row2,column,bit)` which computes the function EAR_n. The procedure tests whether the submatrix formed by the rows from `row1` to `row2` and the columns from `column` to n contains two equal adjacent rows. The procedure assumes that the entry at the row `row1` and the column `column` is equal to the value of `bit`. The procedure starts sweeping the entries of the column `column` from the row `row1` to the row `row2`. If it finds two non-equal adjacent entries (for the first time) in the same column, let us say `matrix[i,column]<>matrix[i+1,column]`, then it sets `row2` to be `i` and it newly defines `row3` to be `i+1` and `row4` to be the original value of `row2`. The procedure now continues testing whether the submatrix formed by the rows from `row1` to `row2` and the columns from `column+1` to n **or** the submatrix formed by the rows from `row3` to `row4` and the columns from `column` to n contain two equal adjacent rows.

If the procedure finds another pair of non-equal adjacent entries, the submatrix which the procedure was called with is now split into three parts (their borders are the two pairs of rows with non-equal adjacent entries). The procedure now recursively calls itself to the part with the least number of rows. If no two non-equal adjacent entries are found, the value of `column` is increased by one and the procedure continues sweeping the next column. The procedure continues in the just described fashion until both the parts vanish, i.e., they collapse to a single row, or `column` exceeds n. In the latter case, the original submatrix contains a pair of equal adjacent rows and the procedure accepts.

A C-like pseudocode of the procedure `test` follows (see also some comments

to it below):

Algorithm 1

```
Initial call: test(1, n, 1, matrix[1,1])

int i,j;

void test(int row1, int row2, int column, int bit) {
  if (row1==row2) return;
  if (column>n) accept;
one_block:
  if (row1==row2) return;
  for (i=row1; i<row2; i++)
    if (matrix[i+1,column]!=bit) {
      row3=i+1; row4=row2; row2=i; bit=!bit;
      goto two_blocks_second;
      }
  if (column++==n) accept;
  bit=matrix[row1,column]; goto one_block;
two_blocks:
  if (row1==row2) {
    row1=row3; row2=row4; bit=matrix[row3,column];
    goto one_block;
    }
  for (i=row1; i<row2; i++)
    if (matrix[i+1,column]!=bit) {
      if ((i-row1<=row2-(i+1))&&(i-row1<=row4-row3)) {
        j=row1; row1=i+1; bit=!bit;
        test(j,i,column+1,matrix[j,column+1]);
        goto two_blocks;
        }
      if ((row2-(i+1)<=i-row1)&&(row2-(i+1)<=row4-row3)) {
        j=row2; row2=i; test(i+1,j,column,!bit);
        bit=matrix[row3,column]; goto two_blocks_second;
        }
      if (row4-row3<=i-row1)&&(row4-row3<=row2-(i+1))) {
        j=row4; row4=row2; row2=i; i=row3; row3=row2+1;
        test(i,j,column,matrix[i,column]);
        bit=!bit; goto two_blocks_second;
        }
    }
  bit=matrix[row3,column];
two_blocks_second:
  if (row3==row4) {
    if (column++==n)
      if (row1<row2) accept; else return;
```

```
    bit=matrix[row1,column];
    goto one_block;
    }
  for (i=row3; i<row4; i++)
    if (matrix[i+1,column]!=bit) {
      ...
      }
  if (column++==n) accept;
  bit=matrix[row1,column]; goto two_blocks;
  }
```

The pseudocode of the algorithm consists of three parts delimited by the labels `one_block:`, `two_blocks:` and `two_blocks_second:`. The omitted code in the third part is analogical to the corresponding code in the second one. In the first part, the algorithm sweeps a submatrix formed by the rows from the row `row1` to the row `row2`. In the second and the third part, the submatrix is split into two blocks, one formed by the rows from the row `row1` to the row `row2` and the other by the rows from the row `row3` to the row `row4`. The first block is swept in the part of the pseudocode after the label `two_blocks:`, while the second block in the part of the pseudocode after the label `two_blocks_second:`.

The following two propositions can be proved by induction on $\texttt{column} = n, \ldots, 1$ and $\texttt{row2} - \texttt{row1} = 1, \ldots, n-1$ (for the values at the time of the procedure call). We leave their very straightforward proofs to the reader:

Proposition 1 *The procedure* `test` *from Algorithm 1 accesses only the entries of the matrix with the coordinates* $[x, y]$ *which satisfy one of the following conditions:*

$$\texttt{row1} < x \leq \texttt{row2} \text{ and } y = \texttt{column}$$

$$\texttt{row1} \leq x \leq \texttt{row2} \text{ and } \texttt{column} < y \leq n$$

Moreover, the procedure `test` *accesses each such entry at most once.*

Proposition 2 *The procedure* `test` *accepts iff there are two equal adjacent rows in the submatrix of the input matrix formed by the rows from the row* `row1` *to the row* `row2` *and by the columns from the column* `column` *to the column* n*.*

Since when a new recursive call is made, the number of rows of the new submatrix is at most one third of the rows of the original submatrix, we have the following:

Proposition 3 *The depth of recursion of the procedure* `test` *from Algorithm 1 is at most* $\log_3 n = \log n / \log 3$.

Theorem 1 *There is a free binary decision diagram* $\mathcal{B}$ *computing* EAR_n *of size* $2^{(3/\log 3)\log^2 n + O(\log n)} \approx 2^{1.89 \log^2 n + O(\log n)}$.

Proof: The number of recursive calls of Algorithm 1 is at most $\log n / \log 3$ by Proposition 3. At each call, the variables `row1`, `row2`, `row3`, `row4`, `column` and `bit` need to be stored. Note that the variables `i` and `j` are global ones. There are $\log n$ bits and 1 bit needed to store the variables `column` and `bit`, respectively. The number of bits needed to store each of the variables `row1`, `row2`, `row3` and `row4` at the first call of the procedure is $\log n$, at the second one $\log n - \log 3$, at the third one $\log n - 2\log 3$, etc. because at each call the number of rows of the submatrix is (at least) three times smaller. Hence the overall number of bits needed for all instances of the procedure is $\log n / \log 3 \cdot (4 \log n/2 + \log n + O(1)) = 3\log^2 n / \log 3 + O(\log n)$. In addition, $O(\log n)$ bits are needed to store values of the global variables `i` and `j`. The space complexity of Algorithm 1 is hence $3\log^2 n / \log 3 + O(\log n)$ (measured in the number of used bits). The number of different states which may be reached by Algorithm 1 is thus at most $2^{3\log^2 n / \log 3 + O(\log n)}$. Note that the state also includes, in addition to the content of variables, the pointer to the instruction to be executed — since the code of the algorithm is finite, this adds only a finite number of bits per recursion level and this can be estimated by $O(\log n)$.

We create a BDD $\mathcal{B}$ with $2^{3\log^2 n / \log 3 + O(\log n)}$ nodes which simulates the computation of Algorithm 1: The nodes of $\mathcal{B}$ correspond to the states of Algorithm 1 just before accessing an entry of the input matrix and depending on the value of the entry the computation (in the diagram) continues to one of the consequent nodes. The computation reaches the 1–sink if Algorithm 1 accepts. $\mathcal{B}$ computes the function EAR_n by Proposition 2. By Proposition 1 applied to the initial call of the procedure, the diagram $\mathcal{B}$ scans each entry of the matrix at most once and hence $\mathcal{B}$ is a free binary decision diagram.

■

4 Lower Bound

The lower bound proof proceeds as follows: We fix a FBDD $\mathcal{B}$ computing the function EAR_n. An adversary constructs on-line an input matrix for the diagram based on the actual computation and the computation is stopped at a certain moment. The way in which the adversary constructs the input and when it stops the computation depends on parameters which are fixed in advance. Then, it is showed that a single node of the diagram can be final only for a limited number of combinations of adversary parameters and the lower bound is derived.

4.1 Adversary Parameters

The adversary strategy is described by a quadruple (ρ, τ, B, c). For the sake of brevity, the quadruple (ρ, τ, B, c) is called *adversary strategy* in the rest. The first element of the quadruple, ρ, is a positive number less than $1/2$. Fix R now

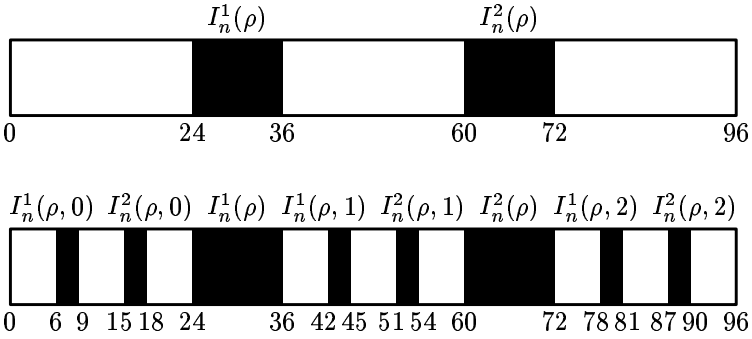

Figure 1: Intervals $I_n^k(\rho)$ and $I_n^k(\rho, a_1)$ for $n = 96$, $\rho = 1/8$, $k = 1, 2$ and $a_1 = 0, 1, 2$.

as follows:

$$R = \left\lfloor \frac{\log \rho n}{\log \frac{3}{1-2\rho}} \right\rfloor$$

We always assume that n is at least $n \geq \frac{3}{(1-2\rho)\rho}$, in particular, $R \geq 1$. Observe that by the choice of R, we have:

$$n \left(\frac{1-2\rho}{3}\right)^R \geq 1/\rho > 2 \tag{1}$$

In the rest, the actual values of n and ρ which determine R are always clear from the context and hence we decided not to add any subscript to R to emphasize this dependence.

A *ternary sequence* of length l is a sequence of l integers between 0 and 2. Let $I_n^k(\rho, a_1, \ldots, a_l)$ for $k = 1, 2$ and a ternary sequence $a_1, \ldots, a_l$, $0 \leq l \leq R-1$, be the following interval (cf. Figure 1):

$$I_n^k(\rho, a_1, \ldots, a_l) =$$

$$\left\langle n \cdot \left(\frac{k(1+\rho)}{3} - \rho\right) \cdot \left(\frac{1-2\rho}{3}\right)^l + \sum_{i=1}^{l} n \cdot \left(\frac{a_i(1+\rho)}{3}\right) \cdot \left(\frac{1-2\rho}{3}\right)^{i-1}, \right.$$

$$\left. n \cdot \left(\frac{k(1+\rho)}{3}\right) \cdot \left(\frac{1-2\rho}{3}\right)^l + \sum_{i=1}^{l} n \cdot \left(\frac{a_i(1+\rho)}{3}\right) \cdot \left(\frac{1-2\rho}{3}\right)^{i-1} \right\rangle$$

Observe that the length of $I_n^k(\rho, a_1, \ldots, a_l)$ is equal to $n\rho \left(\frac{1-2\rho}{3}\right)^l$. By (1), the length is at least 1 and hence $I_n^k(\rho, a_1, \ldots, a_l)$ contains at least one integer. Note that an interval $I_n^k(\rho, a_1, \ldots, a_l)$ can contain integers only between 1 and $n-1$

(inclusively). Observe also that all the intervals $I_n^k(\rho, a_1, \ldots, a_l)$ are disjoint. Moreover, the distance between any two of them is at least $n\left(\frac{1-2\rho}{3}\right)^R \geq 1/\rho$ which is more than 2 by (1).

We now describe the remaining three elements of the quadruple (ρ, τ, B, c). The second element τ is a pair (τ_1, τ_2) of mappings from all ternary sequences of length at most $R-1$ (including the empty sequence) to integers. A sequence $a_1, \ldots, a_l$ is mapped by τ_k to an integer from the interval $I_n^k(\rho, a_1, \ldots, a_l)$. The third element B is a binary matrix of size $R \times n$. There is no restriction on B. The last element c is a sequence $c_1, \ldots, c_R$ of length R which consists of positive integers such that $c_i \in \langle (i-1)\lfloor n/2R-1 \rfloor + L, i\lfloor n/2R-1 \rfloor \rangle$ where $L = \left\lfloor \frac{1}{\log 3} \log^2 n \right\rfloor$. In the rest, we always assume that n is so large that $n/2R - 1 \geq L$. Note also that $n/2 > c_R$. The value of L is not accidental, it was chosen in such a way that 2^L is (at least) the desired lower bound (cf. Lemma 3).

4.2 Description of Adversary Strategy

Let n and the quadruple (ρ, τ, B, c) be fixed in this subsection. The adversary creates the matrix M in a way which depends on the actual computation of the diagram $\mathcal{B}$. A column of the input matrix is *clear* if no entry of it was scanned by $\mathcal{B}$ so far. If $\mathcal{B}$ attempts to scan an entry from a clear column, then the adversary fixes the entries of the whole column for the rest of the computation. Such a column is said to be *fixed*. Let T_k for $k = 1, \ldots, R$ be the union of image sets of the mappings τ_1 and τ_2 restricted to ternary sequences of length at most $k-1$. Note that $|T_1| = 2$, $|T_2| = 2 + 3 \cdot 2 = 8$, $|T_3| = 2 + 3 \cdot 2 + 9 \cdot 2 = 26$, etc. We also have $T_1 \subset T_2 \subset T_3 \cdots \subset T_R$ by the definition of T_k. For the sake of brevity (in the rest of the paper), define T_0 to be the empty set. Let us define for $k = 1, \ldots, R$ and $l = 1, \ldots, n$ column vectors $m^{k,l}$ of size n as follows:

$$m_i^{k,l} = \begin{cases} 1 & |T_k \cap \langle 1, i-1 \rangle| \text{ is odd and } B_{k,l} = 1, \\ 1 & |T_k \cap \langle 1, i-1 \rangle| \text{ is even and } B_{k,l} = 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, n$. As observed in the previous subsection, the distance between any two intervals $I_n^k(\rho, a_1, \ldots, a_l)$ is at least $1/\rho > 2$ and hence the difference between any two elements of T_R is at least 3. The first c_1 columns are fixed by the adversary to be $m^{1,1}, \ldots, m^{1,c_1}$. The next $c_2 - c_1$ columns are fixed by the adversary to be $m^{2,1}, \ldots, m^{2,c_2-c_1}$, the next $c_3 - c_2$ columns to be $m^{3,1}, \ldots, m^{3,c_3-c_2}$, etc. In this way, the adversary creates (a part of) an input matrix M. Note that some columns, e.g. m^{1,c_1+1}, are not used by the adversary at all.

The k-th and $(k+1)$-th row form the *k-th pair of rows*. The computation *discovered* difference of the k-th pair of rows if there exists i such that $\mathcal{B}$ scanned the entries $M_{k,i}$ and $M_{k+1,i}$ of the input matrix M and $M_{k,i} \neq M_{k+1,i}$. Observe that if $\mathcal{R}$ is the set of k's such that difference of the k-th pair of rows was discovered, then $\mathcal{R} \subseteq T_R$. A column of M is said to be *open* if it is fixed and there is $i \notin \mathcal{R}$ such that exactly one of the entries of the column in the i-th and $(i+1)$-th row was scanned. A column of the input matrix is said to be

k-scanned, $1 \leq k \leq R$, if there are $i \in T_k \cup \{0\}$ and $j \in T_k \cup \{n\}$, $i < j$, such that all the entries of the column between the rows $i+1$ and j (inclusively) were scanned during the computation.

The adversary *stops* the computation when $\mathcal{B}$ attempts to scan the (c_R+1)-th column. The adversary can also (prematurely) abort the computation: The adversary *aborts* the computation if there are (at least) L open columns. Let $v_{\rho,\tau,B,c}$ be the node of $\mathcal{B}$ at which the computation is stopped or aborted for the strategy determined by the quadruple (ρ, τ, B, c). In the case that the computation is stopped, $v_{\rho,\tau,B,c}$ is the node at which $\mathcal{B}$ attempts to scan the $(c_R + 1)$-th column. In the case that the computation is aborted, $v_{\rho,\tau,B,c}$ is the node at which there are L open columns before $\mathcal{B}$ scans the entry of M determined by the node $v_{\rho,\tau,B,c}$. If some elements of the quadruple (ρ, τ, B, c) are clear from the context, we omit them in the subscript of $v_{\rho,\tau,B,c}$.

4.3 Analysis of the Strategy

Let $\mathcal{B}$ be a fixed free binary decision diagram computing the function EAR_n throughout this subsection.

Lemma 1 *Let (ρ, τ, B, c) and (ρ', τ', B', c') be two different adversary strategies. Let $\mathcal{R}$, $\mathcal{R}'$, respectively, be the set of k's such that difference of the k-th pair of rows was discovered before reaching the node $v_{\rho,\tau,B,c}$, $v_{\rho',\tau',B',c'}$, respectively. If $\mathcal{R} \neq \mathcal{R}'$, then $v_{\rho,\tau,B,c} \neq v_{\rho',\tau',B',c'}$.*

Proof: Assume for the sake of contradiction that $\mathcal{R} \neq \mathcal{R}'$ and $v_{\rho,\tau,B,c} = v_{\rho',\tau',B',c'}$. Since $\mathcal{R} \neq \mathcal{R}'$, we can assume that there is $i_0 \in \mathcal{R}' \setminus \mathcal{R}$. We show that $\mathcal{B}$ does not compute the function EAR_n. Since $n/2 > c_R$ and $n/2 > c'_R$, there is k such that the k-th column is clear for both the computations. Let M be the part of the input matrix scanned at the node $v_{\rho,\tau,B,c}$. Add to M as the k-th column such a vector that only the i_0-th pair of the rows of the matrix can be equal and then fill M in such a way that the i_0-th pair of the rows is equal. This can be done by the choice of k and because $i_0 \notin \mathcal{R}$. Let M_0 be the whole resulting matrix.

If the computation continues from the node $v_{\rho,\tau,B,c}$, the diagram $\mathcal{B}$ must accept because M_0 contains a pair of equal adjacent rows (the i_0-th pair).

Let M' be the part of the input matrix scanned by the diagram at the node $v_{\rho',\tau',B',c'}$. Set missing entries of M' to be equal to the corresponding entries of M_0. Let M'_0 be the resulting matrix. By the choice of k, the k-th columns of M_0 and M'_0 are equal. Since $i_0 \in \mathcal{R}'$, M'_0 does not contain two equal adjacent rows. If the computation continues from the node $v_{\rho',\tau',B',c'}$, the diagram $\mathcal{B}$ can scan only the entries of M'_0 not contained in M' (and these entries are equal to the corresponding entries of M_0) and hence $\mathcal{B}$ accepts. But M'_0 does not contain two equal adjacent rows as noted above — contradiction.

■

Lemma 2 *Let (ρ, τ, B, c) and (ρ', τ', B', c') be two different adversary strategies. Let $\mathcal{M}$, $\mathcal{M}'$, be the set of entries of the input matrix scanned by the computation before reaching the node $v_{\rho,\tau,B,c}$, $v_{\rho',\tau',B',c'}$, respectively. If $\mathcal{M} \neq \mathcal{M}'$, then $v_{\rho,\tau,B,c} \neq v_{\rho',\tau',B',c'}$.*

Proof: Assume for the sake of contradiction that $\mathcal{M} \neq \mathcal{M}'$ and $v_{\rho,\tau,B,c} = v_{\rho',\tau',B',c'}$. We show that $\mathcal{B}$ does not compute the function EAR_n. Let $\mathcal{R}$, $\mathcal{R}'$, be the set of k's such that difference of the k-th pair of rows was discovered before reaching the node $v_{\rho,\tau,B,c}$, $v_{\rho',\tau',B',c'}$, respectively. By Lemma 1, we have $\mathcal{R} = \mathcal{R}'$.

Let i_0 and j_0 be such that the entry of the i_0-th row and the j_0-th column is contained in $\mathcal{M}$ but not in $\mathcal{M}'$. Since the difference between any two numbers of $\mathcal{R}$ is at least 3, $i_0 - 1 \notin \mathcal{R} = \mathcal{R}'$ or $i_0 \notin \mathcal{R} = \mathcal{R}'$. Assume that $i_0 \notin \mathcal{R}$; the other case is symmetric. Since $n/2 > c_R$ and $n/2 > c'_R$, there is k_0 such that the k_0-th column is clear with respect to both $\mathcal{M}$ and $\mathcal{M}'$. Let M be the part of the input matrix scanned at the node $v_{\rho,\tau,B,c}$. Add to M as the k_0-th column such a vector that only the i_0-th pair of the rows of the matrix can be equal and then fill M in such a way that the rows of the i_0-th pair of the rows are equal. This can be done by the choice of k_0 and because $i_0 \notin \mathcal{R}$. Let M_0 be the whole resulting matrix.

If the computation continues from the node $v_{\rho,\tau,B,c}$, the diagram $\mathcal{B}$ must accept because M_0 contains a pair of equal adjacent rows (namely, the i_0-th pair).

Let M' be the part of the input matrix scanned by the diagram at the node $v_{\rho',\tau',B',c'}$. Set missing entries of M' to be equal to the corresponding entries of M_0. Let M'_0 be the resulting matrix. Let M''_0 be obtained from M'_0 by complementing its entry at the i_0-th row and the j_0-th column. By the choice of k_0, the k_0-th columns of M_0, M'_0 and M''_0 are equal. Hence, if M'_0 or M''_0 contain two equal adjacent rows, then this is the i_0-th pair of the rows. On the other hand, they differ at the entry at the i_0-th row and the j_0-th column, and thus at least one of them does not contain a pair of equal adjacent rows. If the computation continues from the node $v_{\rho',\tau',B',c'}$, the diagram can scan only the entries contained in neither M nor M' (and these entries are equal to the corresponding entries of M_0 both for M'_0 and M''_0) and hence $\mathcal{B}$ accepts both the matrices M'_0 and M''_0 — contradiction.

■

Lemma 3 *If there exists ρ, τ and c such that for each matrix B, the adversary following the strategy (ρ, τ, B, c) aborts the computation, then the size of $\mathcal{B}$ is at least $2^L = 2^{\left\lfloor \frac{1}{\log 3} \log^2 n \right\rfloor}$.*

Proof: Let ρ, τ and c be fixed throughout the proof. Let V be the set of nodes of $\mathcal{B}$ at which the computation is aborted for the strategy (ρ, τ, B, c) with some B. We show that $|V| \geq 2^L$. Assume for the sake of contradiction the opposite, i.e., $|V| < 2^L$. Consider a node $v_0 \in V$ and let B_0 be the set of all matrices B

such that the computation is aborted at the vertex v_0. Since $|V| < 2^L$, there is a node v_0 such that $|B_0| > 2^{Rn-L}$. Fix such a node v_0 for the rest of the proof.

Let $\mathcal{R}$ be the set of k's such that difference of the k-th pair of rows was discovered when the computation was aborted at v_0 for some $B \in B_0$. By Lemma 1, $\mathcal{R}$ is the same for all choices $B \in B_0$. Let $\mathcal{M}$ be the set of entries scanned by the computation before abortion at v_0 for some $B \in B_0$. Again, $\mathcal{M}$ does not depend on the actual choice of $B \in B_0$ (by Lemma 2). Observe that it can be determined whether a column is open from the sets $\mathcal{R}$ and $\mathcal{M}$. Hence, for each choice of $B \in B_0$, the same L columns are open when the computation is aborted. Let now $j_1, \ldots, j_L$ be indices of the open columns at v_0. By the definition of an open column, there exists i_k for each j_k, such that exactly one of the entries with the coordinates $[i_k, j_k]$ and $[i_k + 1, j_k]$ is contained in $\mathcal{M}$ and $i_k \notin \mathcal{R}$.

Fix k to be an integer between 1 and L. Assume that the entry with the coordinate $[i_k, j_k]$ is contained in $\mathcal{M}$ and the entry with the coordinate $[i_k+1, j_k]$ is not (the other case is symmetric). We show that the value of the entry at the i_k-th row and at the j_k-th column is the same for all $B \in B_0$. Let B_1 and B_2 be two matrices of B_0. Let M_1 and M_2 be the parts of the input matrix scanned for B_1 and B_2, respectively. We now complete the matrix M_1. Since $n > n/2 > c_R$, there is at least one clear column. Fix this column in such a way that the only pair of rows which may be equal is the i_k-th pair and complete the rest of the matrix in such a way that the rows of the i_k-th pair are equal (this is possible because $i_k \notin \mathcal{R}$). Let M_1' be the resulting matrix. If the computation continues with the matrix M_1', the diagram $\mathcal{B}$ accepts (the rows of the i_k-th pair are equal). Now set missing entries of M_2 to be equal to corresponding entries of M_1'. Let M_2' be the obtained matrix. Since both the matrices are equal outside $\mathcal{M}$, the diagram $\mathcal{B}$ also accepts M_2'. The only two equal adjacent rows of M_2' can be the i_k-th pair of rows and since $\mathcal{B}$ accepts M_2', they are equal. The entries of M_1' and M_2' with the coordinate $[i_k + 1, j_k]$ are the same (they are outside $\mathcal{M}$) both in M_1' and M_2'. Hence, the entries with the coordinate $[i_k, j_k]$ must be also the same.

We now show that $|B_0| \leq 2^{Rn-L}$. When the computation starts scanning a column with the index j_k for $k = 1, \ldots, L$, the value of the entry of $B \in B_0$ which determines the j_k-th column of the input matrix is determined because the entry with the coordinates $[i_k, j_k]$ or $[i_k + 1, j_k]$ is the same for all $B \in B_0$. Hence, L out of Rn entries of the matrix B cannot be chosen arbitrarily and $|B_0| \leq 2^{Rn-L}$ — contradiction.

■

Lemma 4 *Let (ρ, τ, B, c) be a quadruple for which the adversary stops the computation. When the computation is stopped, the number of k_0-scanned columns is at least $c_{k_0} - L + 1$ for all $k_0 = 1, \ldots, R$.*

Proof: We show that at the time when the diagram attempts to scan the $(c_{k_0} + 1)$-th column, there are at least $c_{k_0} - L + 1$ k_0-scanned columns. Let $m^{k,l}$

be column vectors and T_k sets as in Subsection 4.2. Observe that if a column fixed to $m^{k,l}$ with $k \leq k_0$ is not k_0-scanned at the time when $\mathcal{B}$ attempts to scan the $(c_{k_0}+1)$-th column, it must be open. Since at that time, there are at most $L-1$ open columns (otherwise, the computation is aborted), there at least $c_{k_0} - L + 1$ k_0-scanned columns.

■

Lemma 5 *Let (ρ, τ, B, c) be a quadruple for which the adversary stops the computation. Let $\mathcal{R}$ be the set of i's such that difference of the i-th pair of rows was discovered before reaching the node $v_{\rho,\tau,B,c}$. Let T_k, $1 \leq k \leq R$, be as in Subsection 4.2. Then $|\mathcal{R} \cap (T_{k_0} \setminus T_{k_0-1})| \geq 1$ for each $k_0 = 1, \ldots, R$.*

Proof: Let $m^{k,l}$ be column vectors defined as in Subsection 4.2. Consider the moment when the diagram attempts to scan the $(c_{k_0}+1)$-th column. At that moment, at least one of the columns fixed to be $m^{k_0,l}$ for some l is not open — otherwise the computation is aborted. Then, by the definition of $m^{k_0,l}$, there is $i \in T_{k_0} \setminus T_{k_0-1}$ such that difference of the i-th pair of rows was discovered.

■

Lemma 6 *Let (ρ, τ, B, c) be a quadruple for which the adversary stops the computation and let k_0 be an integer between 2 and R. Let $\mathcal{R}$ be the set of i's such that difference of the i-th pair of rows was discovered before reaching the node $v_{\rho,\tau,B,c}$. Let T_k, $1 \leq k \leq R$, be as in Subsection 4.2. If the number of (k_0-1)-scanned columns is at least $c_{k_0-1}+1$, then $|\mathcal{R} \cap (T_{k_0} \setminus T_{k_0-1})| \geq 2$.*

Proof: Let $m^{k,l}$ be column vectors defined as in Subsection 4.2. Since at least $c_{k_0-1}+1$ columns are (k_0-1)-scanned, there is a (k_0-1)-scanned column fixed to $m^{k_0',l}$ for some $k_0' \geq k_0$ and some l. Then, there are integers $i \in T_{k_0-1} \cup \{0\}$ and $j \in T_{k_0-1} \cup \{n\}$, $i < j$, such that all the entries of the column $m^{k_0',l}$ between the rows $i+1$ and j were scanned. Observe now that $|[i+1, j-1] \cap (T_{k_0} \setminus T_{k_0-1})| \geq 2$. Hence, $|\mathcal{R} \cap (T_{k_0} \setminus T_{k_0-1})| \geq 2$.

■

4.4 The Bound

Theorem 2 *Fix a number ρ and assume that for each τ and c, there exists a matrix B such that the computation is stopped by the adversary for the quadruple (ρ, τ, B, c). Let S be the following number:*

$$S = \prod_{i=0}^{R-1} \min \left\{ \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor^2, \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor \cdot \frac{(n/2R) - L - 1}{L} \right\}$$

If a FBDD $\mathcal{B}$ computes the function EAR_n, then its size must be at least S.

Proof: Fix a matrix B for each τ and c such that the computation is stopped by the adversary for the quadruple (ρ, τ, B, c). Let $v_{\tau,c}$ be the node at which the computation is stopped and let V be the set of all such nodes $v_{\tau,c}$. Let $\mathcal{R}_v$ and $\mathcal{M}_v$ for $v \in V$ be the set of discovered differences of pairs of rows and scanned entries of the input matrix, respectively, before reaching the node v. By Lemmas 1 and 2, the values of sets $\mathcal{R}_v$ and $\mathcal{M}_v$ do not depend on the choice of τ and c for which $v = v_{\tau,c}$. Let c_i^v, $1 \leq i \leq R$, be the number of i-scanned columns at v (this is determined by $\mathcal{R}_v$ and $\mathcal{M}_v$). Let T_i^τ for $1 \leq i \leq R$ be a set T_i corresponding to τ defined as in Subsection 4.2. Let further I_i, $i = 1, \ldots, R$, be the union of all intervals $I_n^j(\rho, a_1, \ldots, a_l)$ where the union is taken over all possible ternary sequences of length at most $i-1$ (i.e., $l \leq i-1$) and over $j = 1, 2$. In addition, set I_0 to the empty set. Clearly, we have $T_i \subseteq I_i$.

We say that a node v is *consistent* with τ and c if the following conditions are satisfied:

1. $\mathcal{R}_v \subseteq T_R^\tau$,
2. it holds that $|\mathcal{R}_v \cap (I_{k+1} \setminus I_k)| \geq 2$ for each $k = 1, \ldots, R-1$ with $c_k^v > c_k$ and
3. $c_k - L + 1 \leq c_k^v$ for each $k = 1, \ldots, R$.

Fix a pair τ and c and let us consider the node $v = v_{\tau,c}$. Since all intervals $I_n^j(\rho, a_1, \ldots, a_i)$ are disjoint, we have $|\mathcal{R}_v \cap (I_k \setminus I_{k-1})| \geq 1$ for all $k = 1, \ldots, R$ by Lemma 5. By Lemma 6, the inequality $c_k^v > c_k$ for $k = 1, \ldots, R-1$ implies $|\mathcal{R}_v \cap (I_{k+1} \setminus I_k)| \geq 2$. Finally, we have $c_k - L + 1 \leq c_k^v$ for each $k = 1, \ldots, R$ by Lemma 4. Hence, we can conclude that if $v = v_{\tau,c}$, then v is consistent with τ and c.

We now compute the number of all possible pairs τ and c. Functions τ_1 and τ_2 can be chosen in the following number of ways (let it be denoted by S_τ):

$$S_\tau := \prod_{\substack{(a_1,\ldots,a_l) \\ 0 \leq l \leq R-1}} \prod_{k=1,2} ||I_n^k(\rho, a_1, \ldots, a_l)||$$

where the outer product ranges over all ternary sequences with length at most $R-1$ and $||I_n^k(\rho, a_1, \ldots, a_l)||$ is the number of integers contained in the interval $I_n^k(\rho, a_1, \ldots, a_l)$. The sequence c can be chosen in $S_c := \lfloor n/2R - L \rfloor^R$ ways. Hence, the number of pairs τ and c is $S_\tau S_c$. We show that a single node $v \in V$ is consistent with at most $S_\tau S_c / S$ pairs of τ and c. This immediately implies that $|V| \geq S$ and the lower bound from the statement of the theorem follows.

Fix a node $v \in V$ of the diagram $\mathcal{B}$. The number of pairs of mappings τ with which the node v is consistent is equal to the following:

$$S'_\tau := \prod_{\substack{(a_1,\ldots,a_l) \\ 0 \leq l \leq R-1}} \prod_{\substack{k=1,2 \\ I_n^k(\rho,a_1,\ldots,a_l) \cap \mathcal{R}_v = \emptyset}} ||I_n^k(\rho, a_1, \ldots, a_l)||$$

This is because all the intervals $I_n^k(\rho, a_1, \ldots, a_l)$ are disjoint and $\mathcal{R}_v \subseteq T_R$. Since $||I_n^k(\rho, a_1, \ldots, a_l)|| \geq \lfloor |I_n^k(\rho, a_1, \ldots, a_l)| \rfloor = \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^l \right\rfloor$, we have that S'_τ is at

most:

$$\frac{S_\tau}{\prod\limits_{k=0}^{R-1}\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor^{|\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|}}$$

We establish an upper bound on the number of sequences c consistent with the node v. Recall that if $|\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|<2$, $k=1,\ldots,R-1$, and c is consistent with v, then $c_k-L+1\le c_k^v\le c_k$. In addition, the last inequality always holds for $k=R$ from trivial reasons. Thus, we have the following upper bound on the number of consistent sequences c:

$$L\cdot\prod_{\substack{k=1,\ldots,R-1\\ |\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|<2}} L\cdot\prod_{\substack{k=1,\ldots,R-1\\ |\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|\ge 2}}\lfloor n/2R-L\rfloor=$$

$$S_c\cdot\frac{L}{\lfloor(n/2R)-L\rfloor}\cdot\prod_{\substack{k=1,\ldots,R-1\\ |\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|<2}}\frac{L}{\lfloor(n/2R)-L\rfloor}\le$$

$$S_c\cdot\frac{L}{(n/2R)-L-1}\cdot\prod_{\substack{k=1,\ldots,R-1\\ |\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|<2}}\frac{L}{(n/2R)-L-1}$$

Thus, the node v is consistent with at most the following number of pairs τ and c (recall that $\mathcal{R}_v\cap(I_k\setminus I_{k-1})\ne\emptyset$ for each $k=1,\ldots,R$):

$$\frac{S_\tau\cdot S_c}{\prod\limits_{k=0}^{R-1}\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor^{|\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|}\cdot\frac{(n/2R)-L-1}{L}\cdot\prod\limits_{\substack{k=1,\ldots,R-1\\ |\mathcal{R}_v\cap(I_{k+1}\setminus I_k)|<2}}\frac{(n/2R)-L-1}{L}}\le$$

$$\frac{S_\tau\cdot S_c\cdot\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^0\right\rfloor^{-1}\cdot\frac{L}{(n/2R)-L-1}}{\prod\limits_{k=1}^{R-1}\min\left\{\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor^2,\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor\cdot\frac{(n/2R)-L-1}{L}\right\}}\le$$

$$\frac{S_\tau\cdot S_c}{\prod\limits_{k=0}^{R-1}\min\left\{\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor^2,\left\lfloor n\rho\left(\frac{1-2\rho}{3}\right)^k\right\rfloor\cdot\frac{(n/2R)-L-1}{L}\right\}}=\frac{S_\tau\cdot S_c}{S}$$

■

Theorem 3 *Let $\mathcal{B}$ be a FBDD computing the function* EAR_n. *Then, the size of $\mathcal{B}$ is at least* $2^{\frac{1}{\log 3}\log^2 n-O(\log n\cdot\log\log n)}\approx 2^{0.63\log^2 n-O(\log n\cdot\log\log n)}$.

Proof: In the proof, we assume that n is sufficiently large (the bound on n is implicitly stated at the end of this paragraph). Let $\rho=(\log n)^{-1}$. The number R is the following:

$$R=\left\lfloor\frac{\log\rho n}{\log\frac{3}{1-2\rho}}\right\rfloor=\left\lfloor\frac{\log\rho+\log n}{\log 3-\log(1-2\rho)}\right\rfloor=\left\lfloor\frac{\log n+\log\rho}{\log 3+2\rho+O(\rho^2)}\right\rfloor=$$

$$\frac{\log n + \log \rho}{\log 3} \cdot \left(1 - \frac{2\rho}{\log 3} + O(\rho^2)\right) + O(1) = \frac{\log n}{\log 3} - \frac{\log \log n}{\log 3} + O(1)$$

Assume in the rest that n is so large that $\rho < 1/2$, $R \geq 1$ and $(n/2R) - 1 \geq L$. The inequality $(n/2R) - 1 \geq L$ is satisfied for n sufficiently large, because $(n/2R) - 1 = \Theta(n/\log n)$ and $L = \Theta(\log^2 n)$.

If there is τ and c such that for each B, the computation of $\mathcal{B}$ is aborted, then the lower bound follows from Lemma 3. Otherwise, Theorem 2 can be used. Before we show that the lower bound on the size of the diagram $\mathcal{B}$ from Theorem 2 is at least $2^{\frac{1}{\log 3}\log^2 n - O(\log n \cdot \log \log n)}$, some auxiliary computations need to be performed. Recall that by the choice of R, we have $\rho n \left(\frac{1-2\rho}{3}\right)^R \geq \rho/\rho \geq 1$, in particular, $\log\left(\rho n \left(\frac{1-2\rho}{3}\right)^R\right) \geq 0$. Thus, the following inequality holds for each $i = 0, \ldots, R$:

$$\log \rho n \left(\frac{1-2\rho}{3}\right)^i = \log \rho n - i \cdot \log \frac{3}{1-2\rho} \geq \frac{R-i}{R} \cdot \log \rho n =$$

$$\frac{R-i}{R} \cdot (\log n - \log \log n + O(1)) \tag{2}$$

By Theorem 2, the size of $\mathcal{B}$ is at least S where:

$$S = \prod_{i=0}^{R-1} \min\left\{ \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor^2, \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor \cdot \frac{(n/2R) - L - 1}{L} \right\}$$

The bounds on logarithms of members of the product are computed using (2):

$$\log \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor^2 = \frac{2(R-i)}{R} \cdot (\log n - \log \log n + O(1)) \tag{3}$$

And:

$$\log \left\lfloor n\rho \left(\frac{1-2\rho}{3}\right)^i \right\rfloor + \log \frac{(n/2R) - L - 1}{L} =$$

$$\frac{(R-i)}{R} \cdot (\log n - \log \log n + O(1)) + \log n - \log R - \log L + O(1) =$$

$$\frac{(R-i)}{R} \cdot (\log n - \log \log n + O(1)) + \log n - 3 \log \log n + O(1) \tag{4}$$

Comparing (3) and (4) immediately yields:

$$\log S \geq \sum_{i=0}^{R-1} \frac{2(R-i)}{R} \cdot (\log n - O(\log \log n) + O(1)) =$$

$$\frac{2(R+1)}{2} \cdot (\log n - O(\log \log n) + O(1)) = \frac{1}{\log 3} \log^2 n - O(\log n \log \log n)$$

Hence, the size of $\mathcal{B}$ must be really at least $2^{\frac{1}{\log 3}\log^2 n - O(\log n \cdot \log \log n)}$.

■

5 Open problems

In this section, we pose two open problems. First of all, the multiplicative constants of the factor $\log^2 n$ in the exponent of our lower and upper bounds do not match. Hence, we have the following problem:

Problem 1 *Determine c such that the size of optimal FBDDs computing the function* EAR_n *is* $2^{c\cdot\log^2 n+o(\log^2 n)}$.

Theorems 1 and 3 imply that $\frac{1}{\log 3} \leq c \leq \frac{3}{\log 3}$.

As pointed out to us by Jiří Sgall, our lower bound proof also works for "narrow" matrices. More precisely, if we consider $n \times \Omega(\log^3 n)$ matrices instead of $n \times n$ matrices, we have also a lower bound on FBDDs of order $2^{\Omega(\log^2 n)}$: Just set ρ to a fixed constant and $L = \Theta(\log^2 n)$ both in Lemma 3 and Theorem 2. We find interesting to determine what is the size of optimal FBDDs for $n \times \Theta(\log^k n)$ matrices for $1 < k < 3$ (note that if $k \leq 1$, it is easy to construct FBDDs of size polynomial in n):

Problem 2 *Find a function $f(k)$ for $1 < k < 3$ such that the size of optimal FBDDs for $n \times (\log^k n)$ matrices is* $2^{\Theta(\log^{f(k)} n)}$.

Our lower bound proof can be modified for such narrow matrices: Set ρ to a fixed constant, $R = \Theta(\log^{(k-1)/2} n)$ and $L = \Theta(\log^{(k+1)/2} n)$. Then, we have a lower bound of $2^{\Omega(\log^{(k+1)/2} n)}$ by Lemma 3 and Theorem 2. On the other hand, we miss any non-trivial upper bounds: If $k \geq 2$, our only upper bound on the size of FBDDs for such $n \times (\log^k n)$ matrices is $2^{O(\log^2 n)}$ and if $1 < k < 2$, we have only a trivial upper bound of order $2^{O(\log^k n)}$. Thus, we know only the following estimates on the function $f(k)$ from Problem 2:

$$\begin{array}{ll} \frac{k+1}{2} \leq f(k) \leq k & \text{for } 1 < k < 2 \text{ and} \\ \frac{k+1}{2} \leq f(k) \leq 2 & \text{for } 2 \leq k < 3. \end{array}$$

Acknowledgement

The second author would like to thank Petr Savický for introducing binary decision diagrams during his outstanding lectures on the topic. Both the authors would like to thank Petr Savický and Jiří Sgall for their comments on the problem.

References

[1] Babai, L., Hajnal, P., Szemerédi, E., Turán, G.: A lower bound for read–once only branching programs. Journal of Computer and System Sciences **35** (1987) 153–162.

[2] Kára, J., Král', D.: Optimal Free Binary Decision Diagrams for Computation of EAR_n, Proc. 27th International Symposium on Mathematical Foundations of Computer Science 2002, LNCS vol. **2420** (2002) 411–422.

[3] Kriegel, K., Waack, S.: Lower bounds on the complexity of real–time branching programs. RAIRO — Theoretical Informatics and Applications **22** (1988) 447–459.

[4] Masek, W.: A fast algorithm for the string editing problem and decision graph complexity. M. Sc. Thesis, MIT (1976).

[5] Savický, P., Žák, S.: A large lower bound for 1–branching programs. ECCC report 96-036 (1996).

[6] Savický, P., Žák, S.: A read–once lower bound and $(1, +k)$–hierarchy for branching programs. Theoretical Computer Science **238(1-2)** (2000) 347–362.

[7] Wegener, I.: Branching Programs and Binary Decision Diagrams — Theory and Applications. SIAM Monographs on Discrete Mathematics and Applications 4 (2000).

[8] Wegener, I.: On the complexity of branching programs and decision trees for clique functions, Journal of the ACM **35** (1988) 461–471.

[9] Žák, S.: An exponential lower bound for one–time–only branching programs. Proc. 11th International Symposium on Mathematical Foundations of Computer Science 1984, LNCS vol. **176** (1984) 562–566.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'