Perfect Hashing, Graph Entropy, and Circuit Complexity (preliminary version)

Ilan Newman
Hebrew University

Prabhakar Ragde *
University of Waterloo

Avi Wigderson †
Hebrew University

Abstract

We present two simple and general techniques, both of which should find further application. First, we show that approximate compaction can be efficiently performed in constant parallel time using perfect hash functions. This allows us to show that polylogarithmic-threshold functions are in linear AC^0.

Next, we show that the information-theoretic notion of graph entropy captures some aspect of the difficulty of computing Boolean functions. We use this to derive superlinear lower bounds on the formula size of threshold and other simple Boolean functions.

1 Introduction

Hash functions have a long history in computer science, but their applications in circuit complexity are less well known. As a simple example of their use in this domain, consider that a family of k-separating hash functions naturally yields monotone formulae for the threshold function Th_k^n [17]. Th_k^n is the Boolean function of n variables whose value is 1 if and only if at least k of the variables have value 1.

The notion of graph entropy was introduced by Korner [18] as a generalization of Shannon's entropy, to capture encoding of words over an alphabet in which only a certain subset of pairs of letters has to be distinguished. Korner [19] related this notion (and its generalization to hypergraphs) to hash functions by using it to rederive, in a simple way, the Fredman-Komlos lower bound on the size of a k-separating family of hash functions [12]. Boppana [3] has used the concept of graph entropy to obtain optimal lower bounds for CRCW PRAMs.

In this paper, we reveal more connections between hash functions, graph entropy, and the complexity of computing Boolean functions. We will apply them to obtain upper and lower bounds on circuits and formulae for threshold and other functions.

In the first part, we show that by means of a deterministic application of perfect hash functions, approximate compaction of a sparse array can be done in constant time by a CRCW PRAM with a linear number of processors. As an application, we describe a simple uniform family of **linear-size** unbounded fan-in constant-depth circuits for the threshold functions Th_k^n, for every $k = (\log n)^{O(1)}$. It was not previously known that these could be computed in linear size, and indeed this is somewhat surprising, since even the fact that

*Supported by an operating grant from the National Sciences and Engineering Research Council of Canada.

†Partially supported by American-Israeli Binational Science Foundation Grant No. 87-00082.

CH2899-3/90/0000/0091$01.00 © 1990 IEEE

these functions are in AC^0 (**polynomial-size** constant-depth circuits) is non-trivial.

Placing these functions in AC^0 was first done by probabilistic arguments [1]; uniform AC^0 circuits were described by Denenberg, Gurevich, and Shelah [8]. They also used a family of hash functions, though they were not interested in optimizing size, and their constructions do not compete with subsequent polynomial-size constructions such as in Fagin, Klawe, Pippenger, and Stockmeyer [9]. Independently, Wegener, Wurm and Yi [30] proved a result similar to ours, using a similar method; their construction can be made monotone, and we briefly describe how this can be done for our construction as well.

In the second part we consider the information theoretic notion of graph entropy. Very informally, graph entropy captures the information that a random vertex gives about the independent sets containing it, and thus in some sense measures the complexity of a graph. There are natural ways to associate graphs and hypergraphs with Boolean functions, and we use this measure to capture a complexity aspect of the function, which is related to its formula size.

This entropy bound is useful for simple functions, for which previously known techniques are too crude to apply. We obtain a general lower bound for quadratic Boolean functions, based on the entropy of the associated graph. We use it to prove that any formula that tests if a graph on n nodes contains a cycle of length 4 must have size $\Omega(n^2 \log n)$. We also use this connection to rederive, in a simple way, two known lower bounds on the formula size of threshold functions:

1. The $n \log n$ lower bound of Krichevskii [21] for the size of any formula computing the function Th_2^n.

2. The $\Omega(nk \log(n/k))$ lower bound of Snir [28] for the size of any monotone $\vee \wedge \vee$ formula computing Th_k^n.

It is interesting to observe that even though graph entropy is a monotone function on graphs, it yields non-monotone lower bounds.

2 Upper Bounds on Circuit Size

The class AC^0 (functions computable by polynomial-size constant-depth unbounded fan-in circuits) has been the subject of much interest. In the interests of efficient computation, it is natural to consider which functions are in LC^0, that is, computable by constant-depth unbounded fan-in circuits with a linear number of gates. An even more restrictive class is WLC^0, in which only a linear number of wires is allowed. As an example of a WLC^0 function, consider the n-output function that determines the position of the input bit of lowest index with value 1 ([10, 5]). Addition of two n-bit numbers has been shown to be in LC^0 but not WLC^0 [5]; there is no 1-output function known to separate AC^0 from LC^0 or LC^0 from WLC^0.

Threshold functions of n variables, like all symmetric functions, can be computed by bounded fan-in circuits of size $O(n)$ and depth $O(\log n)$ [24]. Chandra, Stockmeyer, and Vishkin [6] give a construction for unbounded fan-in circuits which achieves size $2^{cn^{1/d}}$ and depth $O(d)$, for any $d > 2$ and some constant c; Hastad [16] has proved that, up to the constant c, this is the smallest size achievable for $k = n/2$.

This does not rule out the possibility that for smaller values of k, smaller circuits may exist; in fact, it is easy to construct circuits of size $O(n^k)$ for Th_k^n. Ajtai and Ben-Or [1] demonstrate a general technique for converting probabilistic AC^0 circuits into deterministic AC^0 circuits, and use this technique to show that Th_k^n is in AC^0, provided that $k \leq (\log n)^r$. Their method proves the existence of circuits of size $O(n^r)$ and depth

$O(r)$. Note that Hastad's result implies that if $k = \omega(\log^r n)$ for every $r > 0$, Th_k^n is not in AC^0.

It is natural to conjecture that polylog threshold functions are candidates for separating AC^0 and LC^0; this paper shows such a conjecture to be false, by giving a simple uniform construction of circuits of size $O(n)$ and depth $O(r)$ for Th_k^n when $k \leq (\log n)^r$. The first part of the circuit performs what we may call approximate compaction: it either finds that there are more than k 1's on the input wires, or compresses the 1's onto at most $n' = k^4$ wires. Then, since n' is now small relative to n, the construction of [6] can be applied.

The approximate compaction is achieved through the use of perfect hash functions, as described in the following theorem.

Theorem 2.1 *For any k, there exists an unbounded fan-in circuit of size $o(n)$ and depth $O(1)$ which has n inputs $x_1, x_2, \ldots, x_n$ and k^4+1 outputs $z, y_1, y_2, \ldots y_{k^4}$ such that for any input, $z = 0$ if and only if $\sum_{i=1}^{n} x_i = \sum_{i=1}^{k^4} y_i$.*

Proof: It suffices to consider $k \leq n^{1/4}$. For clarity, we first describe an algorithm running on a CRCW PRAM with n processors; if several processors simultaneously write into a cell, an arbitrary processor succeeds. The algorithm first divides the input bits into $\sqrt{n}$ groups of size $\sqrt{n}$ and takes the OR of each group. It then attempts to map the $\sqrt{n}$ group bits into space k^2. This is accomplished by means of the set of perfect hash functions provided by the following lemma from [13]:

Lemma 2.1 *Let p be the smallest prime greater than m. For every $S \subseteq \{1, 2, \ldots, m\}$, $|S| \leq k$, there exists a value α, $1 \leq \alpha \leq p-1$, such that the mapping $x \rightarrow (\alpha x \bmod p) \bmod k^2$ is one-to-one when restricted to S.* ∎

In this case $m = \sqrt{n}$; the algorithm devotes $\sqrt{n}$ processors to each possible value of α. The $\sqrt{n}$ processors each take one input bit and attempt to map it; if a collision is detected, that value of α can be discarded. In $O(1)$ time, then, the algorithm either finds a value of α that achieves the one-to-one mapping, or finds that no such value exists (in which case there are more than k 1's in the group bits and thus in the input). This value of α suffices to map the original input bits into k^2 blocks of size $\sqrt{n}$, or space $k^2\sqrt{n}$. The process is now repeated, with the groups being of size k^2 in the second phase, resulting in compression of the original input bits into k^2 groups of size k^2, or total space k^4.

A naive conversion of this algorithm results in a constant-depth circuit with $O(n)$ gates and $O(n^2)$ wires; with care, this can be reduced to $o(n)$ gates and $O(k^2 n)$ wires. For convenience in describing the construction, we assume that $\sqrt{n} = r$ is an integer.

Let us rename the input bits $x_1, x_2, \ldots, x_n$ as $\{x_{i,j}\}$, where $x_{i,j}$ is the jth bit within block i. The circuit forms the bits $a_i = \vee_{j=1}^{r} x_{i,j}$ for $i = 1, 2, \ldots, r$; a_i is the ith block bit. To test whether a particular hash function f_i is good or not, we need threshold-2 circuits Th_2^r; for this we can use the results of Friedman [14] though in the case of threshold-2 there is a particularly simple construction. Let $S_{i,j} = \{m \mid f_i(m) = j\}$; then $b_{i,j} = Th_2^r(\{a_m \mid m \in S_{i,j}\})$ (for $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, k^2$) computes whether function i causes a collision in location j when the block bits are hashed to space k^2. This is the only place where $O(k^2 n)$ wires are needed. Computing $c_i = \vee_{j=1}^{k^2} b_{i,j}$ for $i = 1, 2, \ldots p$ determines if function f_i is bad. The situation where all functions are bad is captured by $z_1 = \wedge_{i=1}^{p} c_i$.

At this point there may be several good hash functions; one is selected by computing $d_i = (\wedge_{j=1}^{i-1} c_j) \wedge \bar{c}_i$. The bit d_i is 1 if and only if function i is selected. Determining whether or not block i maps to location j is done by computing $e_{i,j} = \vee\{d_m \mid i \in S_{m,j}\}$

for $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots k^2$. Finally, the first phase of the compression is completed by computing the jth bit of the new block i, namely $f_{i,j} = \vee_{m=1}^{r}(e_{m,i} \wedge x_{m,j})$ for $i = 1, 2, \ldots, r$, $j = 1, 2, \ldots k^2$.

For the second phase of the compression, we switch the roles of the indices (that is, we consider k^2 blocks, each of r bits) and repeat the previous development. Letting $x'_{i,j} = f_{j,i}$, we have: $a'_i = \vee_{j=1}^{r} x'_{i,j}$ for $i = 1, 2, \ldots, r$; $b'_{i,j} = Th_2^r(\{a'_m \mid m \in S_{i,j}\})$; $c'_i = \vee_{j=1}^{k^2} b'_{i,j}$; $z_2 = \wedge_{i=1}^{p} c'_i$; $d'_i = (\wedge_{j=1}^{i-1} c'_j) \wedge \bar{c}'_i$; $e'_{i,j} = \vee\{d'_m \mid i \in S_{m,j}\}$; $f'_{i,j} = \vee_{m=1}^{r}(e'_{m,i} \wedge x'_{m,j})$. Renaming the $\{f_{i,j}\}$ as $y_1, y_2, \ldots y_{k^4}$ and letting $z = z_1 \vee z_2$ concludes the formal description of the circuit.

Theorem 2.2 *If $k \leq (\log n)^d$ for some integer $d > 0$, then Th_k^n is in LC^0.*

Proof: We first use the approximate compression circuit to compress the input bits onto k^4 wires. The construction in [6] gives a circuit for $Th_k^{k^4}$ of depth $O(d)$ and size at most $2^{c(\log^{4d} n)^{1/4(d+1)}}$, which is $o(n)$. Connecting the outputs $y_1, y_2, \ldots y_{k^4}$ of the approximate compression circuit to the inputs of the $Th_k^{k^4}$ circuit and taking the conjunction of the output of the $Th_k^{k^4}$ circuit with z yields a Th_k^n circuit with $o(n)$ gates and $O(k^2 n)$ wires. ■

The non-monotonicity of the circuit comes from the need to select a unique hash function at each level. Wegener, Wurm, and Yi, in independent work [30], construct monotone circuits of the same complexity; for completeness, we briefly describe how our construction can be made monotone by their methods. The need to select a single hash function is obviated by trying them all in parallel. The construction of [6] is not monotone, but it can be replaced by the only slightly more complicated monotone construction of Boppana [2] The resulting circuit is monotone, but its size is superlinear due to the large number of hash functions in our perfect family. Since we have r functions per level and two levels, we are basically trying $O(n)$ hash functions in parallel. Fortunately, smaller families exist: Mehlhorn [23] gives a probabilistic construction that in our case gives a family of size $O(k \log n)$ per level, and a deterministic construction of size $O(k^2 \log n)$ per level. These allow us to construct monotone circuits with $o(n)$ gates and $O(n \log^{O(1)} n)$ wires.

Further applications of approximate compaction to the problems of approximate counting and exact compaction are given in [26].

3 Lower Bounds On Formula Size

In this section, we give a general method for deriving lower bounds for some simple Boolean functions where the previously known general methods fail [11, 22, 25, 29].

Let X be a finite set, interpreted as Boolean variables. A **formula** is a rooted tree whose leaves are labeled with members of X or their negations, and whose internal nodes are labeled with the Boolean operations AND, OR. The root of the tree computes a Boolean function $f : \{0,1\}^X \mapsto \{0,1\}$ in the natural way. The formula and the function computed are both monotone if no negations appear. The size of a formula is the number of leaves in the tree. (An equivalent way of looking at formulae is as fan-in two, fan-out one Boolean circuits, where size is the number of wires carrying input bits.)

The **formula size** of a Boolean function f, denoted $L(f)$, is the minimum size of a formula for f. For a monotone function f, the minimum size of the monotone formula for f is denoted by $L_M(f)$. For a Boolean function f on n variables and for $T \subseteq \{1, 2, \ldots n\}$, we identify $f(T) = f(x)$, where x is the characteristic vector of T.

A brief review of the notion of entropy is helpful; further information can be found in [7]. Let Y, Z be random variables in some probability space. The entropy of Y is defined as $H(Y) = \sum_y p(y)\log(1/p(y))$. The mutual information between Y, Z is defined as $I(Y,Z) = H(Y) + H(Z) - H((Y,Z))$ and it may also be written as $I(Y,Z) = H(Y) - \sum_{y,z} p(y,z)\log(1/p(y|z))$.

Korner ([18, 19]) defined the notion of graph entropy. For a graph $G(V,E)$, let P be a probability distribution on V and let $A(G)$ be the collection of all maximal independent sets of G. Let X be a random variable on V and Y a random variable on $A(G)$. Define $P(G)$ to be the set of all probability distributions Q_{XY} on $V \times A(G)$ such that for every $v \in V$, $Q_{XY}(v,I) = 0$ if $v \notin I$, and $Q_X(v) = P(v)$.

The graph entropy $H(G,P)$ is defined as $\min\{I(X,Y)|Q_{XY} \in P(G)\}$. Hereafter we consider only uniform distributions on V, so we refer to the graph entropy of G as H(G). We shall need the following basic properties of H(G), as shown in [18, 19, 20].

E1: For two graphs $G = (V, E_G)$, $F = (V, E_F)$ on the same vertex set, let $K = G \cup F$ be the graph on V with $E = E_G \cup E_F$. Then $H(K) \leq H(G) + H(F)$.

E2: The graph entropy is monotone; deleting an edge can only decrease the graph entropy.

E3: If K_n is the complete graph on n vertices, then $H(K_n) = \log n$.

E4: The entropy of a bipartite graph on m (out of n) vertices does not exceed m/n.

We first give a lower bound for the size of a monotone formula for monotone functions. We do that by assigning each monotone function a cost. We prove that the cost of a function computed by $\vee$ and $\wedge$ gates is no more than the sum of costs of the inputs. Hence, by computing the cost of the output function, and the cost of a single variable, one gets a lower bound.

For any Boolean function g on n variables (which we take to be the set {1,2,...,n}), let $(g)_k$ be the set of minterms of g of size k. That is, $(g)_k = \{S|\ S \subseteq \{1,2,\ldots,n\}, |S| = k,\ g(S) = 1,\ \forall T \subset S\ g(T) = 0\}$.

For the proofs given in this abstract, we will be interested only in $(g)_1, (g)_2$. Observe that $(g)_1$ is a subset of $\{1,2,\ldots n\}$ and $(g)_2$ is a set of unordered pairs on $\{1,2,\ldots,n\}$. $(g)_2$ will be identified with the graph $G(g) = (V,E)$, $V = \{1,2,\ldots n\}$, $E = (g)_2$. The cost μ of a function g will be defined as $\mu(g) = H(G(g)) + (|(g)_1|/n)$.

Theorem 3.1 *If g is a monotone Boolean function on n variables, then $L_M(g) \geq n\mu(g)$.*

Proof: We note that for a variable x_i (a leaf of a formula), $(x_i)_1 = \{i\}$, $G(x_i) = \phi$, (the empty graph), and so $\mu(x_i) = 1/n$. Furthermore, the cost function is monotone with respect to inclusion; if $(g)_1 \subseteq (h)_1$ and $(g)_2 \subseteq (h)_2$ then $\mu(g) \leq \mu(h)$. We show that, for any internal node of a formula, the cost of the subfunction computed at that node is at most the sum of the costs of the subfunctions computed at the two children. The result follows by an easy induction.

To show sub-additivity for an $\vee$ node, let $g = h \vee f$. We have that $(g)_1 = (h)_1 \cup (f)_1$, and $(g)_2 \subseteq G(h) \cup G(f)$, thus

$$\begin{aligned}\mu(g) &\leq (|(h)_1 \cup (f)_1|/n) \\ &\quad + H(G(h) \cup G(f)) \\ &\leq (|(h)_1|/n) + (|(f)_1|/n) \\ &\quad + H(G(h)) + H(G(f)) \\ &= \mu(h) + \mu(f).\end{aligned}$$

The first inequality is by the monotonicity of μ. The second is by property E1 of graph entropy.

To show sub-additivity for an $\wedge$ gate, let $g = h \wedge f$, $A = (h)_1$, and $B = (f)_1$. We have that $(g)_1 = A \cap B$, and $(g)_2 \subseteq G(h) \cup G(f) \cup G_\Delta(A, B)$, where $G_\Delta(A, B)$ denotes the complete bipartite graph G with bipartition $(A - B, B - A)$. Thus

$$\begin{aligned}
\mu(g) &\leq H(\, G(h) \cup G(f) \cup G_\Delta(A,B)) \\
&\quad +(|A \cap B|/n) \\
&\leq H(G(h)) + H(G(f)) \\
&\quad + \frac{|A-B| + |B-A|}{n} + \frac{|A \cap B|}{n} \\
&\leq H(G(h)) + H(G(f)) + (|A| + |B|)/n \\
&= \mu(h) + \mu(f).
\end{aligned}$$

The first inequality is by the monotonicity of μ. The second is by E1 and E4. ■

Note that, since $\mu(g) \leq \log n + 1$, this method cannot prove lower bounds greater than $\Omega(n \log n)$. The next theorem allows us to apply our monotone lower bound to non-monotone formula size.

Theorem 3.2 *Let f be any function with $f(S) = 0$ for every S, $|S| = 1$. Then $L(f) \geq n\mu(f)$.*

Proof: Krichevskii [21] proved the following lemma by induction on the number of variables.

Lemma 3.1 *Let $f(x_1, .., x_n)$ be any Boolean function for which $f(S) = 0$ for every S, $|S| = 1$. Then, there is a monotone function ψ_f such that*

1. *$\psi_f(S) = 0$ for any S, $|S| = 1$.*
2. *$\psi_f(S) \geq f(S)$ for every S, $|S| = 2$.*
3. *$L_M(\psi_f) \leq L(f)$.* ■

By the lemma there is a monotone function ψ_f for which $L(f) \geq L_M(\psi_f)$; by theorem 3.1 we have $L_M(\psi_f) \geq n\mu(\psi_f)$. Since $G(\psi_f) \supseteq G(f)$, the theorem follows by monotonicity of the cost function μ. ■

Theorem 3.2 gives a general lower bound for quadratic Boolean functions in terms of the entropy of the associated graph, although it is applicable to any function. For our first application, we give a simple proof of a theorem originally due to Krichevskii ([21], see also [15]).

Corollary 3.1 $L(Th_2^n) \geq n \log n$.

Proof: $(Th_2^n)_1 = \phi$ and $G(Th_2^n) = K_n$ (the complete graph on n vertices); thus by property E3, $\mu(Th_2^n) = \log n$. We note that it is simple to prove that there is a matching upper bound. ■

Let $C4_N$ be the Boolean function that has value 1 on a graph of N vertices if and only if the graph contains a cycle of length 4. Let $D2_N$ be the Boolean function that has value 1 on a graph of N vertices if the graph contains a vertex of degree at least 2. Note that $C4_N$ and $D2_N$ are Boolean functions on $n = \binom{N}{2}$ variables, where a variable denotes the presence or absence of an edge.

Corollary 3.2 *Any formula for $D2_N$ is of size $\Omega(N^2 \log N)$.*

Proof: $(D2_N)_1 = \phi$ and $(D2_N)_2 = \mathcal{L}(K_N)$, the line graph of K_N. By explicitly specifying distributions according to the definition of graph entropy, we will show an upper bound of $\log(N-1)$ and $\log N/2$ on the graph entropies of $\mathcal{L}(K_N)$ and its complement, respectively. But the sum of these two graph entropies must be at least $\log \binom{N}{2}$ (by E1 and E3), and so the upper bounds are in fact tight. It follows that $\mu(D2(n)) = \Omega(\log n)$, and by theorem 3.2 the lower bound follows.

The independent sets of $\mathcal{L}(K_N)$ are matchings (in K_n). The cliques in $\mathcal{L}(K_N)$ are stars and triangles (in K_n). (A star is a set of edges all of which are adjacent to one vertex). Let $\mathcal{M}$ denote the set of perfect matchings, $\mathcal{S}$ denote the set of maximal stars and E denote the edge set of K_n. Define the

probability distribution Q_1 on $E \times \mathcal{M}$ where $Q_1(e|M) = \frac{2}{n}$ for every matching $M \in \mathcal{M}$ and $e \in M$, and the induced probability on $\mathcal{M}$ is uniform. Let (X_1, Y_1) be a random variable on $E \times \mathcal{M}$ distributed according to Q_1. Define the probability distribution Q_2 on $E \times \mathcal{S}$ where $Q_2(e|S) = \frac{1}{n-1}$ for every star in $\mathcal{S}$, $e \in \mathcal{S}$ and the induced probability on $\mathcal{S}$ is uniform. Let (X_2, Y_2) be a random variable on $E \times \mathcal{S}$ distributed according to Q_2. Observe that Q_1 is a probability distribution in $P(\mathcal{L}(K_N))$ (from the definition of graph-entropy) and Q_2 is one for the complement of $\mathcal{L}(K_N)$.

It can be checked that $I_1(X_1, Y_1) = \log(n-1)$ and $I_2(X_2, Y_2) = \log n/2$. ■

Corollary 3.3 *Any formula for $C4_N$ is of size $\Omega(N^2 \log N)$.*

Proof: Let f be the restriction of $C4_{N+1}$ obtained by taking a special vertex z and setting all variables corresponding to edges adjacent to z to 1. It is easy to see that $(f)_1 = \phi$ and $f \geq D2(n)$. By monotonicity of the cost function and by corollary 3.2, the result follows. ■

For our final application of this technique, we use the natural generalization of graph entropy to hypergraphs as developed in [18], [19], [20]. Using these notions, we can give a simple proof to the following lower bound obtained by Snir [28].

A formula of the form $\vee_{i=1}^{p} \wedge_{j=1}^{k} \vee_{q \in S_{ij}} t_q$, where t_q is a variable or its complement, is called an $\vee \wedge \vee$ formula. Note that every $\wedge$ gate has fan-in k.

Theorem 3.3 *[28] The size of a $\vee \wedge \vee$ formula for T_k^n, where every $\wedge$ gate has fan in k, is at least $\frac{n \log(n/(k-1))}{\log(k/(k-1))}$.*

We first define hypergraph entropy, and state the properties we need for the lower bound proof. Let $G = (V, E)$ be an hypergraph on vertex set V (E is a collection of subsets of V called 'edges'). An independent set $S \subseteq V$ of G is a set of vertices that does not contain any 'edge' $e \in E$. The hypergraph entropy is defined the same as graph entropy, with the above definition of independent sets. An hypergraph for which every edge $e \in E$ has size k is called a k-uniform hypergraph. We will restrict ourselves to k-uniform hypergraphs.

Hypergraph entropy has properties similar to graph entropy:

H1: For two hypergraphs $G = (V, E_G)$, $F = (V, E_F)$ on the same vertex set, $H(K) \leq H(G) + H(F)$.

H2: The hypergraph entropy is monotone; deleting an edge can only decrease the hypergraph entropy.

H3: The complete k-uniform hypergraph, K_n^k, is the hypergraph on n vertices, whose edge set consists of all sets of size k. The entropy of the complete k-uniform hypergraph is $H(K_n^k) = \log n - log(k-1)$.

H4: Let G=(V,E) be a k-uniform hypergraph. We call G a k-partite hypergraph if there is a partition of V into k parts $V_1 \ldots V_k$, such that for every edge $e \in E$ and every V_i, $i \in$[k], $|e \cap V_i| = 1$. The entropy of a k-partite hypergraph on m (out of n) vertices is no more than $\frac{m}{n}(\log k - \log(k-1))$.

We now prove the theorem.

Proof: Consider a minimum size $\vee \wedge \vee$ formula for T_k^n. Let $g_i = \wedge_{j=1}^{k} \vee_{q \in S_{ij}} t_q$ be the function computed at the ith $\wedge$ gate. Clearly, the formula must be monotone (i.e., no negation), and for every fixed i ($1 \leq i \leq p$), the sets S_{ij} ($1 \leq j \leq k$) are pairwise disjoint.

Define the hypergraph G_i whose edge set is $(g_i)_k$. G_i is a k-partite hypergraph on vertex set $\bigcup_j S_{ij}$ (the 'parts' are S_{ij}, $1 \leq j \leq k$). Similarly, define T to be the hypergraph whose edge set is $(T_k^n)_k$. T is the complete k-regular hypergraph K_n^k.

Since $T_k^n = \vee_i g_i$ we get that $(T_k^n)_k = \cup_i (g_i)_k$. That is, a formula of this kind for T_k^n defines a way to decompose the complete k-regular hypergraph into a union of k-partite hypergraphs.

The size of the formula is

$$s = \sum_{i=1}^{p} \sum_{j=1}^{k} |S_{ij}| = \sum_{i=1}^{p} |V(G_i)|$$

By H3, the hypergraph entropy of K_n^k is $\log n - \log(k-1)$. By H4, we have $H(G_i) \leq \frac{|V(G_i)|}{n}(\log k - \log(k-1))$. Thus, by H1, it follows that

$$\begin{aligned} H(K_n^k) &= \log n - \log(k-1) \\ &\leq \sum_{i=1}^{p} H(G_i) \\ &\leq \frac{1}{n}(\log k - \log(k-1)) \sum_{i=1}^{p} |V(G_i)| \\ &= \frac{s}{n} \log(k/k-1) \end{aligned}$$

And we get the desired lower bound:

$$s \geq \frac{n \log(n/(k-1))}{\log(k/(k-1))}. \quad \blacksquare$$

Acknowledgements

We are grateful to Ravi Boppana, Aviad Cohen, and Mauricio Karchmer for enlightening discussions.

References

[1] M. Ajtai and M.Ben-Or, A theorem on probabilistic constant depth computation, Proc. 16th ACM STOC (1984), 471-474.

[2] R. Boppana, Threshold functions and bounded depth monotone circuits. Proc. 15th ACM STOC (1984), 475-479.

[3] R. Boppana, Optimal separations between concurrent-write parallel machines, Proc. 21st ACM STOC (1989), 320-326.

[4] L. Carter and M. Wegman, Universal classes of hash functions, J. Comp. Syst. Sci 18 (1979), 143-154.

[5] A. Chandra, S. Fortune, and R. Lipton, Unbounded fan-in circuits and associative functions, J. Comp. Syst. Sci. 30 (1986), 222-234.

[6] A. Chandra, L. Stockmeyer, and U. Vishkin, Constant depth reducibility, SIAM J. Comp. 13 (1981), 423-429.

[7] I. Csiszar and J. Korner, Information theory: coding theorems for discrete memoryless systems, Academic Press, New York, 1982.

[8] L. Denenberg, Y. Gurevich, and S. Shelah, Definability by constant-depth polynomial-size circuits, Information and Control 70 (1986), 216-240.

[9] R. Fagin, M.M. Klawe, N.J. Pippenger, and L. Stockmeyer, Bounded-depth, polynomial-size circuits for symmetric functions, Theor. Comp. Sci 16 (1985), 239-250.

[10] F.E. Fich, P. Ragde, and A. Wigderson, Relations among concurrent-write PRAMs (preliminary version), Proc. 3rd ACM PODC (1984), 179-189.

[11] M. Fischer, A. Meyer, and M. Paterson, $\Omega(n \log n)$ lower bounds on length of Boolean formulas. SIAM J. of Computing, 11 (1982), 416-427.

[12] M. Fredman and J. Komlos, On the size of separating systems and perfect hash functions, SIAM J. Alg. Disc. Meth., 5(1984), 61-68.

[13] M. Fredman, J. Komlos, and E. Szemeredi, Storing a sparse table with O(1) worst-case access time, Proc. 23rd IEEE FOCS (1982), 165-169.

[14] J. Friedman, Constructing $O(n \log n)$ size monotone formulae for the kth elementary symmetric polynomial of n Boolean variables, Proc. 25th IEEE FOCS (1984), 506-515.

[15] G. Hansel, Nombre minimal de contacts de fermature necessaires pour realiser une function booleenne symetrique de n variables, C. R. Acad. Sci. Paris 258 (1964), 6037-6040.

[16] J. Hastad, Almost optimal lower bounds for small depth circuits, Proc. 18th ACM STOC, 1986, 6-20.

[17] L.S. Khasin, Complexity bounds for the realization of monotone symmetrical functions by means of formulas in the basis $+, \cdot, -$. Sov. Phys. Dokl. 14, 1149-1151.

[18] J. Korner, Coding of an information source having ambiguous alphabet and the entropy of graphs, Trans. 6-th Prague Conf. on Information Theory, Academia, Prague (1973), 441-425.

[19] J. Korner, Fredman-Komlos bounds and information theory, SIAM. J. Alg. Disc. Meth. 7 (1986), 560-570.

[20] J. Korner and K. Marton, New bounds for perfect hashing via information theory, to appear in European J. Comb.

[21] R.E. Krichevskii, Complexity of contact circuits realizing a function of logical algebra, Sov. Phys. Dokl. 8(1964), 770-772.

[22] V.M. Krapchenko, A method of obtaining lower bounds for the complexity of π-schemes. Math. Notes Acad. Sci. USSR 11 (1972), 474-479.

[23] K. Mehlhorn, On the program size of perfect and universal hash functions, Proc. 23rd IEEE FOCS (1982), 170-175.

[24] D.E. Muller and F.P. Preparata, Bounds to complexities of networks for sorting and for switching, J. ACM 22 (1975), 195-201.

[25] E.I. Nechiporuk, A Boolean function. Sov. Math. Dokl. 7 (1966), 999-1000.

[26] P. Ragde, The parallel simplicity of compaction and chaining, Proc. 17th ICALP (1990).

[27] J. Reif, An optimal parallel algorithm for integer sorting, Proc. 26th IEEE FOCS (1985), 496-504.

[28] M. Snir, The covering problem of complete uniform hypergraphs, a note, Disc. Math. 27 (1979) 103-105.

[29] I. Wegener, The complexity of Boolean functions. Willy-Teubner Series in Computer Science, 1987.

[30] I. Wegener, N. Wurm, and S.-Z. Yi, Symmetric functions in AC^0 can be computed in constant depth with very small size. Universität Dortmund, Forschungsbericht Nr. 326, 1989.