JOURNAL OF COMPUTER AND SYSTEM SCIENCES 32, 91–96 (1986)

Time–Space Trade-offs for Branching Programs

INGO WEGENER*

FB 20-Informatik, Johann Wolfgang Goethe-Universität, 6000 Frankfurt a.M., Federal Republic of Germany

Received October 1, 1984; revised July 9, 1985

Branching program depth and the logarithm of branching program complexity are lower bounds on time and space requirements for any reasonable model of sequential computation. In order to gain more insight to the complexity of branching programs and to the problems of time–space trade-offs one considers, on one hand, width-restricted and, on the other hand, depth-restricted branching programs. We present these computation models and the trade-off results already proved. We prove a new result of this type by presenting an effectively defined Boolean function whose complexity in depth-restricted one-time-only branching programs is exponential while its complexity even in width-2 branching programs is polynomial. © 1986 Academic Press, Inc.

1. INTRODUCTION

We assume that the reader is familiar with the basic concepts of Boolean networks and formulae for the computation of Boolean functions. Here we consider branching programs.

DEFINITION 1. A branching program is an acyclic labelled graph with one source and arbitrarily many sinks. Each node has out-degree 0 (sink) or 2 (computation node). One successor of each computation node can be reached via a 0-edge (an edge labelled 0) and the other via a 1-edge. The sinks are labelled by Boolean constants and the computation nodes by Boolean variables. The computation for the input vector a starts at the source. At a computation node labelled x_i we use the edge labelled by a_i. The program computes the Boolean function f, where $f(a)$ equals the label of the sink we reach for input a. The depth of a program is the length of the longest path and the complexity is the number of computation nodes. The proper complexity measures are denoted by $\mathrm{BPD}(f)$ and $\mathrm{BP}(f)$.

The logarithm of the complexity of branching programs is often called capacity. It has been shown already by Cobham [4] that depth and capacity are lower bounds on time and space requirements for any reasonable model of sequential computation. Pudlák and Zák [9] proved more tight connections between capacity and space requirements.

* Supported in part by DFG Grant We 1066/1-1.

0022-0000/86 $3.00

Many results are known on the depth of branching programs (see, e.g., Bollobás [1]) but only few results on the complexity of branching programs. Although most of the Boolean functions have exponential BP-complexity the largest known lower bound for effectively defined Boolean functions is an $\Omega(n^2/\log^2 n)$ bound by Nechiporuk [7]. Nechiporuk proved his result for contact schemes but one may easily translate it to branching programs.

In order to gain more insight to the problem of proving lower bounds and time–space trade-offs one has investigated more restricted models. This has been done at first for Boolean networks by the pioneering paper of Furst, Saxe, and Sipser [5] who introduced depth restricted circuits and proved nonpolynomial lower bounds for the parity function, the exactly-half function and other functions.

In order to examine space (or capacity) restricted branching programs one could investigate branching programs whose complexity is bounded by $\mathrm{BP}(f)$ of k $\mathrm{BP}(f)$ for some constant k. This seems to be very difficult since it is already hard to compute $\mathrm{BP}(f)$. Borodin, Dolev, Fich, and Paul [2] introduced width restricted branching programs. These are levelled branching programs where the number of nodes on each level is bounded by some constant k, in [2] $k=2$. In the following sense width-restricted branching programs may be considered as capacity restricted branching programs. While the capacity of any branching program of depth d is at least $\log d$ the capacity of a width-k branching program of depth d is at most $\log(kd)$. Further papers on width restricted branching programs (even for $k>2$) are Chandra, Furst, and Lipton [3], Pudlák [8], and Yao [12].

Already Masek [6] introduced for $k=1$ k-times-only branching programs (BP_k-s), where each variable may be tested on each path of computation only k times. For BP_1-s one may compute efficiently the complexity of symmetric functions (Wegener [10]). One can prove exponential lower bounds for clique functions (Wegener [11]) and functions related to clique functions (Pudlák and Zák [9]). We motivate this restriction in Chapter 2 where we also refer the trade-off results obtained until now. Our main purpose is to compare width an depth restricted branching programs. On the one hand it is already known (Yao [12]) that some functions may have polynomial BP_1-complexity and non-polynomial width-2 BP-complexity. On the other hand we define effectively in Chapter 3 a Boolean function whose BP_1-complexity is exponential but whose width-2 BP-complexity is polynomial.

2. A Discussion of Trade-offs

For width-restricted branching programs one tries to minimize the depth of branching programs, where the capacity is minimum with respect to the depth. For BP_1-s and BP_k-s we try to minimize the capacity, where the depth is bounded. For $\mathrm{BP}_k-\mathrm{s}$ the depth is bounded by kn, where n is the number of variables. One may ask why one does not investigate branching programs whose depth is restric-

ted by BPD(f) or k BPD(f). This seems to be more natural. But similarly to the discussion of width restricted branching programs BPD(f) is often not known. More significant is the following argument. The depth may be large since we have to test all variables on some branch of the program, but all other branches may be short. On these paths the depth restriction is not important. In $BP_1 - s$ each path has an individual depth restriction. One is not allowed to gather computation paths if one is forced to separate them again by repeating an old test. We give an example for this effect. Let f^n_{exhcl} be the function on $N = \binom{n}{2}$ variables corresponding to the possible edges of an n-vertex graph which computes 1 iff the graph consists of an $n/2$-clique and $n/2$ isolated vertices. Pudlák and Zák [9] have shown an exponential lower bound on the BP_1-complexity of the function while Wegener [11] proved a polynomial upper bound on the BP_2-complexity of this function. Furthermore Wegener [10] proved that for this and many other functions the models of branching programs of minimum depth and $BP_1 - s$ coincide. The following function on $2N+1$ variables is essentially an exactly half clique function and should behave like it. It does so for $BP_1 - s$ but not for branching programs of minimum depth; $g_{2N+1}(x_1,..., x_N, y_1,..., y_N, z)$ computes the conjunction of the x and y variables if $z = 1$ and computes $f^n_{\text{exhcl}}(x_1,..., x_N)$ if $z = 0$. It is easy to prove that $\text{BPD}(g_{2N+1}) = 2N+1$. Its BP_1-complexity is exponential since f^n_{exhcl} is a subfunction. In branching programs of minimum depth we may test at first z. If $z = 1$, $2N$ nodes are sufficient to compute g_{2N+1}. If $z = 0$, we have to compute f^n_{exhcl} and the allowed depth is $2N$. By the efficient BP_2 mentioned above we compute g_{2N+1} with a polynomial number of nodes.

The following property should be fulfilled for any significant complexity measure C. If f' is a subfunction of f, i.e., f' results from f by replacing some variables by constants, then $C(f') \leqslant C(f)$. This property is indeed fulfilled for the complexity of $BP_k - s$, but as we have seen by the example above it is not fulfilled for branching programs of minimal depth. Thus $BP_1 - s$ are the proper model of depth restricted branching programs.

We mention now two trade-offs for branching programs proved earlier.

THEOREM 1. *The BP_1-complexity of the exactly half clique function is exponential (Pudlák and Zák [9]) while its BP_2-complexity is polynomial (Wegener [11]).*

This result leads to the open problem whether the classes $BP_k(P)$ of funtions with polynomial BP_k-complexity build a proper hierarchy.

The majority function computes 1 iff the number of ones in the input is at least $n/2$.

THEOREM 2. *The widh-2 branching program complexity of the majority function is not polynomial (Yao [12]) while its BP_1-complexity is polynomial (see, e.g., Wegener [10]).*

3. A New Time–Space Trade-off

We prove the counterpart of Theorem 2. We define effectively a function whose complexity with respect to width-2 branching programs but also to Boolean formulae and depth-2 circuits is polynomial while its BP_1-complexity is exponential. We have no theory for the construction of efficient width-2 branching programs. But the width-2 branching program complexity is bounded by the length of a minimal polynomial for f. Therefore we choose for the proposed trade-off a function where the number of prime implicants is polynomial.

Definition 2. $f^n_{k(n)}$: $\{0,1\}^N \to \{0,1\}$, where $3 \leqslant k(n) \leqslant n$ and $N = \binom{n}{2}$ is a function where the variables correspond to the possible edges of an n-vertex graph. $f^n_{k(n)}$ computes 1 iff the graph specified by the variables contains a $k(n)$-clique on vertices $i_1, \ldots, i_{k(n)}$ such that $i_{j+1} \equiv i_j + 1 \bmod n$ for $1 \leqslant j \leqslant k(n) - 3$, that means $k(n) - 2$ vertices build a successive sequence mod n.

Theorem 3. *The complexity of $f^n_{k(n)}$ in width-2 branching programs and depth-2 circuits is at most $O(n^3 k(n)^2) = O(n^5)$ and therefore polynomial. The BP_1-complexity of $f^n_{k(n)}$ for $k(n) = \lfloor n^{1/3} \rfloor$ is at least $\Omega(2^{m(n)})$, where $m(n) = n^{2/3}/4 - o(n^{2/3})$ and therefore exponential.*

Proof. The cliques corresponding to prime implicants of $f^n_{k(n)}$ are called special cliques. There are n positions where the successive sequence of $k(n) - 2$ vertices may start. For the other two vertices of a special clique we have less than $\binom{n}{2}$ possibilities. Thus $f^n_{k(n)}$ has $O(n^3)$ prime implicants of length $\binom{k(n)}{2}$ each. This proves the upper bound of the theorem.

For the lower bound we show that after at most $m(n)$ arbitrary tests we do not know the value of the function and that two different nodes v and w cannot be merged in a BP_1 for $f^n_{k(n)}$ if some path from the source to v (resp. w) has length at most $\binom{k(n)}{2}/2 = m(n)$. Thus we can conclude that a BP_1 for $f^n_{k(n)}$ has exactly 2^l nodes in distance $l \leqslant m(n)$ from the source and the lower bound follows.

In order to know that $f^n_{k(n)}$ equals 1 we must have tested all $\binom{k(n)}{2}$ edges of a $k(n)$-clique positively. This is not the case after at most $m(n)$ tests. After at most $m(n)$ negative tests there are at most $n^{2/3}/2$ vertices where some adjacent edge is already tested negatively. Thus there is a subsequence of $(n - n^{2/3}/2)/(n^{2/3}/2) = 2n^{1/3} - 1 \geqslant k(n) - 2$ vertices, where no adjacent edge has been tested negatively. The existence of a special clique is still possible and we do not know that $f^n_{k(n)}$ equals 0.

Let v and w be different nodes such that some path $p(v)$ from the source to v and some path $p(w)$ from the source to w has length at most $\binom{k(n)}{2}/2 = m(n)$. It remains to prove the following claim. If we merge v and w the BP_1 does not compute $f^n_{k(n)}$. We may assume that v is not a successor or predecessor of w because in this situation we would create a cycle if we merge v and w. Let $G(v)$ and $G(w)$ be the partial graphs which contain the edges tested positively on $p(v)$ (resp. $p(w)$). The edges tested negatively are called forbidden, the edges not tested yet are called

variable. Since v and w do not lie on the same path there exists some edge e such that e belongs to one of the graphs, w.l.o.g. $G(v)$, and is forbidden in the other graph, $G(w)$. Since $f^n_{k(n)}$ is symmetric with respect to the vertices we assume w.l.o.g. that $e=(1, 2)$. Let us assume that we merge v and w. Having reached this node we are allowed to test only the edges variable in $G(v)$ *and* in $G(w)$. If we declare the edges not tested on $p(v)$ or $p(w)$ in an arbitrary way as existing or forbidden the following must hold if the BP_1 still computes $f^n_{k(n)}$. Either the completion of $G(v)$ and the completion of $G(w)$ contain a special clique (not necessarily the same) or in both completions any special clique has been destroyed by forbidden edges. We derive a contradiction by declaring the edges variable for $G(v)$ and $G(w)$ in such a way as existing and forbidden that the completion of $G(v)$ contains a special clique and the completion of $G(w)$ does not.

Let A be the set of vertices $z \notin \{1, 2\}$ adjacent to some edge tested on $p(v)$ or $p(w)$. By our assumption on the lengths of $p(v)$ and $p(w)$ we conclude that $|A| \leqslant n^{2/3}$. Thus $B := \{1,..., n\} - A$ has cardinality at least $n - n^{2/3}$. B consists of at most $n^{2/3}$ successive subsequences mod n. The longest subsequence has length at least $(n-n^{2/3})/n^{2/3} = n^{1/3} - 1$. Thus we may choose a sequence $i,..., i+k(n)-3 \pmod n$ of vertices all in B. Let D be any $k(n)$-subset of B containing the vertices 1 and 2 and $i,..., i+k(n)-3 \pmod n$. In $G(v)$ and in $G(w)$ all edges on D with the only exception of $e=(1, 2)$ are variable. The edges tested already in $G(v)$ and $G(w)$ and containing one node in D are incident with 1 or 2. We declare all edges on D with the only exception of $e=(1, 2)$ as existing and all other variable edges as forbidden.

$G^*(v)$, the completion of $G(v)$, contains the special clique consisting of the vertices in D. $G^*(w)$, the completion of $G(w)$, does not contain this clique since $e=(1, 2)$ is forbidden. Since $G(w)$ does not contain any special clique and since only edges on D are declared as existing a special clique of $G^*(w)$ has to contain a vertex $z \in D - \{1, 2\}$ and by our considerations above a vertex $z' \notin D$. But the edge (z, z') does not exist in $G^*(w)$. Thus $G^*(w)$ does not contain any special clique. Our assumption that we may merge v and w in a BP_1 for $f^n_{k(n)}$ leads to a contradiction.

Q.E.D.

REFERENCES

1. B. BOLLOBÁS, "Extremal Graph Theory," Academic Press, New York, 1978.
2. A. BORODIN, D. DOLEV, F. E. FICH, AND W. PAUL, Bounds for width two branching programs, *in* "15th Sympos. on Theory of Computing," 1983, pp. 87–93.
3. A. K. CHANDRA, M. L. FURST, AND R. J. LIPTON, Multiparty protocols, *in* "15th Sympos. on Theory of Computing," 1983, pp. 94–99.
4. A. COBHAM, The recognition problem for the set of perfect squares, *in* "7th Sympos. on Switching and Automata Theory," 1966, pp. 78–87.
5. M. L. FURST, J. B. SAXE, AND M. SIPSER, Parity, circuits, and the polynomial time hierarchy, *in* "22nd Sympos. on Foundations of Computer Science," 1981, pp. 260–270.
6. W. MASEK, "A Fast Algorithm for the String Editing Problem and Decision Graph Complexity," M.Sc. thesis, MIT, 1976.

7. E. I. NECHIPORUK, A Boolean function, *Sov. Math. Dokl.* 7 (1966), 999–1000.
8. P. PUDLÁK, A lower bound on complexity of branching programs, preprint, Univ. Prague, 1983.
9. P. PUDLÁK AND S. ZÁK, Space complexity of computations, preprint, Univ. Prague, 1983.
10. I. WEGENER, Optimal decision trees and one-time-only branching programs for symmetric Boolean functions, *Inform. and Control* **62** (1984), 129–143.
11. I. WEGENER, On the complexity of branching programs and decision trees for clique functions, *J. Assoc. Comput. Mach.*, in press.
12. A. C. YAO, Lower bounds by probabilistic arguments, *in* "24th Sympos. on Foundations of Computer Science," 1983, pp. 420–428.