CREW PRAMs and Decision Trees

Noam Nisan[1]

Laboratory of Computer Science
M.I.T.
Cambridge, MA 02139

ABSTRACT

This paper gives a full characterization of the time needed to compute a boolean function on a CREW PRAM with an unlimited number of processors.

The characterization is given in terms of a new complexity measure of boolean functions: the "block sensitivity". This measure is a generalization of the well know "critical sensitivity" measure (see [W], [CDR], [Si]). The block sensitivity is also shown to relate to the boolean decision tree complexity, and the implication is that the decision tree complexity also fully characterizes the CREW PRAM complexity. This solves an open problem of [W].

Our results imply that changes in the instruction set of the processors or in the capacity of the shared memory cells do not change by more than a constant factor the time required by a CREW PRAM to compute any boolean function. Moreover, we even show that a seemingly weaker version of a CREW PRAM, the CROW PRAM ([DR]), can compute functions as quickly as a general CREW PRAM. This solves an open problem of [DR].

Finally, our results have implications regarding the power of randomization in the boolean decision tree model. We show that in this model, randomization may only achieve a polynomial speedup over deterministic computation. This was known for Las-Vegas randomized computation; we prove it also for 1-sided error computation (a quadratic bound) and 2-sided error (a cubic bound).

1. Introduction

1 This work was done while the author was a student in U.C. Berkeley, supported by a grant from Digital Equipment Corporation

1.1. CREW PRAMs

The PRAM (Parallel Random Access Machine) is the "standard" model for parallel computation. In the PRAM model the processors communicate via shared memory cells. We will be interested in the inherent limitations of this model that are due to its basic communication mechanism, and will thus consider the "ideal" PRAM, a model that has no other constraints.

An *Ideal* PRAM consists of an unbounded number of processors and an unboounded number of common memory cells which can be read and written by any processor. Each processor has it's own local memory and possibly unlimited computational power. A PRAM computes a function in the following manner: The input is placed in the common memory cells, and then the computation proceeds in cycles, at each cycle each processor may read one memory location, do any computation using the information it knows, and write any information into one memory cell. Several variants of the PRAM model have been defined, which differ from each other in the way they handle memory access conflicts. Perhaps the most natural variant, and the one we will be considering is the CREW PRAM. For a CREW (Concurrent Read Exclusive Write) PRAM several processors may read from the same location at the same time, but two or more processors may *never* attempt writing into the same location at the same time.

A key result bounding the power of Ideal CREW PRAMs is the following theorem by Cook, Dwork and Reischuk [CDR]: Computing the OR function on n variables by a CREW PRAM takes $\Omega(\log n)$ parallel time. This result is tight since any function can be computed in $\log n$ time on this model. Actually, [CDR] proved a more general result as they give a lower bound on the time needed to compute a function on a CREW PRAM in terms of the function's "sensitivity".

We consider a generalization of the "sensitivity" measure: the "block sensitivity". We show that the [CDR] lower bound can be extended to block sensitivity, and, moreover, that the block sensitivity fully characterizes the complexity on the Ideal CREW PRAM model. We achieve this result by further relating the block sensitivity to the decision tree complexity, and thus we alternatively characterize the CREW complexity in terms of the decision tree complexity.

For a boolean function f, let $CREW(f)$ denote the CREW PRAM complexity of f (i.e. the time needed to compute f on an Ideal CREW PRAM with an unlimited number of processors and memory cells), let $bs(f)$ denote the block sensitivity of f, and let $D(f)$ denote the boolean decision tree complexity of f.

Theorem 1:

$$CREW(f) = \Theta(\log bs(f)) = \Theta(\log D(f))$$

Moreover, the lower bound holds for an Ideal PRAM, with unlimited computational power for each processor, and unlimited capacity of the common memory cells; while the upper bound requires only reasonable power for each processor and 1-bit memory cells. As a corollary we get that the computational power of a single processor, and the capacity of the common memory cells does not make a difference in this model (as long as we do not limit the number of processors, and ignore uniformity questions).

These results also apply to a weaker model than the CREW PRAM: the CROW PRAM (Concurrent Read *Owner* Write) introduced by Dymond and Ruzzo [DR]. For a CROW PRAM, each memory cell is preassigned to some processor who is said to "own" the memory cell. The only processor who may write into a memory cell is the owner, although all processors may read it. This model is clearly a special case of the

CREW model, but we show that the parallel time needed to compute a function on this model is equal (up to a constant factor) to the time needed by a CREW PRAM. Let $CROW(f)$ be the parallel time need to compute f by an Ideal CROW PRAM.

Theorem 2:

$$CROW(f) = \Theta(CREW(f))$$

This result is particularly surprising as it is not achieved by simulation, and only applies to functions on a *full* domain. We actually show that a simulation result is impossible by giving a function on a *partial domain* that separates these two models by a factor of $\log n$.

1.2. Boolean decision trees

The Boolean Decision Tree model is perhaps the the simplest computational model for general boolean functions. A deterministic boolean decision tree computes a boolean function by repeatedly reading input bits until the function can be determined from the bits accessed. The decision of which bit to read at any time may depend on the previous bits read, and it determined by them. The only cost associated with this computation is the number of bits read; all other computation is free. The cost of an algorithm is the number of bits read on the worst case input, and the deterministic complexity of a function is the cost of the best deterministic algorithm for this function.

A randomized decision tree algorithm is also allowed to flip coins in order to determine the next input bit to be accessed. The cost of an algorithm is the expected number of locations examined on the worst case input. The complexity of a function is the cost of the best algorithm for this function. We distinguish between three kinds of probabilistic algorithms: zero error, one-sided error and two-sided error. In zero error computation of f, the algorithm must always be correct, no errors are allowed. For one-sided error computation the algorithm must always reject any string not in the language, and must accept any string in the language with probability of at least 1/2 (Here we identify the function f with the language $\{ x \mid f(x)=1 \}$). A two-sided error algorithm may err in both cases, but must give the correct answer with probability of at least 3/4.

This model has been studied extensively in several contexts. The complexity of *graph* properties in this model has been investigated ([Ro], [RV], [KK], [KSS]). The relation to Oracle Turing machines has been pointed out several times ([BI], [Ta], [IN]). It is also related to sublinear time Turing Machine computations ([IN]). The randomized complexity in this model has also been studied ([Y1], [Y2], [Sn], [K], [SW]).

In this paper we deal with the power of randomization in the boolean decision tree model. Snir, ([Sn], see also [SW]), first showed that randomization "helps": he exhibited a function with deterministic complexity n, but Las-Vegas (zero-error) randomized complexity of only $O(n^{0.753...})$. Saks and Wigderson ([SW]) conjecture that this is the optimal speedup possible by Randomization, and were able to prove it for a particular subclass of functions; for general functions the conjecture is still open. A more general result by Blum implies some bounds on the speedup possible by *zero-error* randomization: zero-error randomized decision tree algorithms can give at most a quadratic speedup. In this paper we show that even allowing error, randomization may only give a polynomial speedup.

Let $R_1(f)$ and $R_2(f)$ denote the one sided error and two sided error probabilistic complexities of f respectively, and $D(f)$ the deterministic complexity in the boolean decision tree model. We show that for any func-

tion f:

Theorem 3:

$$D(f) \leq 2R_1(f)^2$$

Theorem 4:

$$D(f) \leq 8R_2(f)^3$$

Impagliazzo and Naor [IN] have considered the uniform analogue of decision trees. Using our results, and paralleling some results in [BI], they showed that, if P=NP, then DTIME(poly-log)=RTIME(poly-log). Here RTIME(t) is the class of problems that can be solved in time t on a randomized TM, even allowing bounded 2-sided error.

2. Sensitivity and block sensitivity

In this section we will discuss the relationships between several complexity measures of boolean functions. The relationships we show here will then have implications regarding CREW PRAM complexity and boolean decision trees. The complexity measures we consider are the "*sensitivity*", the "*block sensitivity*", and the "*certificate*" complexity. We will also mention the relation of these to the boolean decision tree complexity.

Notation: let w be a boolean string of length n, let S be any subset of indices, $S \subset \{1 \cdots n\}$, then $w^{(S)}$ means the string w, with all bits in S flipped. I.e. $w^{(S)}$ differs from w exactly on S.

Definition: Let f be a boolean function, and w any input string, and i any index. We say f is *sensitive* to x_i on w if $f(w) \neq f(w^{(i)})$. The *sensitivity* of f on w, $s_w(f)$, is the number of locations i such that f is sensitive to x_i on w. The *sensitivity* of f, $s(f)$ is the maximum over all w of the sensitivity of f on w.

The sensitivity of a boolean function has been discussed in the literature: Simon [Si] shows that every function that depends on all its variables has sensitivity of at least $\Omega(\log n)$. Turan [Tu] showed that all *graph* properties have sensitivity of at least $\Omega(v)$. Cook, Dwork, and Reishuk [CDR], use the sensitivity of a function to give lower bounds for the CREW PRAM complexity. What we do here is consider a generalization of the sensitivity by allowing several bits to be flipped together to change the value of the function.

Definition: Let f be a boolean function, w any boolean string, and S any subset of indices. We say that f is sensitive to S on w if $f(w) \neq f(w^{(S)})$. The *block sensitivity* of f on w, $bs_w(f)$ is the largest number t such that there exists t *disjoint* sets $S_1, S_2, \cdots, S_t$ such that for all $1 \leq i \leq t$, f is sensitive to S_i on w. The *block sensitivity* of f, $bs(f)$ is the maximum over all w of the block sensitivity of f on w.

Our main lemma will be the relation between the block sensitivity and the *certificate* complexity:

Definition: Let f be a boolean function, and w any input string. A *1–certificate* (0-certificate) for f is an assignment to some subset of the variables that forces the value of f to 1 (0). The *certificate complexity of f on w*, $C_w(f)$, is the size of the smallest certificate that agrees with w. The *certificate complexity of f*, $C(f)$, is the maximum over all w of $C_w(f)$.

The certificate complexity of a function describes how many bits of the input must be revealed to you (by someone who knows all the input bits) in order to convince you of the value of the function. It may also be viewed as the nondeterministic complexity in the boolean decision tree model. The 1-certificates of f are the terms of f, and the 0-certificates of f are the terms of the com-

plement of f.

We will first mention the obvious relations between them (see [W]):

Proposition 2.1: For any f:

$$s(f) \leq bs(f) \leq C(f)$$

Proof: The left inequality follows directly from the definitions, The right inequality follows from the fact that for any input w, any certificate for w must include at least one variable from each set f is sensitive to on w. ■

It turns out that for a large subclass of functions these three measures of functions are really equal:

Proposition 2.2: For all *monotone* functions f:

$$s(f) = bs(f) = C(f)$$

Proof: It is enough to show that $C(f) \leq s(f)$. Consider a minimal certificate of size $C(f)$, w.l.o.g. assume it is a 1-certificate. The string which has 1 in every bit of the certificate and 0 in all other places, will have sensitivity of $C(f)$. The reason is that turning off any of the 1-bits will change the value of function to 0. ■

The following example shows that for general, non-monotone, functions the inequalities may be strict:

Example 2.3: Let f be the symmetric function on n variables defined to be true iff exactly $n/2$ or $n/2+1$ of the inputs are 1 (for simplicity assume that 4 divides n.) for this function we get that:

$$s(f) = \frac{n}{2} \qquad bs(f) = \frac{3n}{4} \qquad C(f) = n - 1$$

[Ru] exhibits a function with a quadratic gap between the sensitivity and the block sensitivity; it is still an open problem whether the gap may be bigger (superpolynomial?). [WZ] exhibit a function with a polynomial (but subquadratic) gap between the block sensitivity and the certificate complexity. Our main lemma shows that the certificate complexity may only be polynomially bigger than the block sensitivity.

Lemma 2.4: For all boolean functions f:

$$bs(f) \geq \sqrt{C(f)}$$

Proof: Let w be an input achieving the certificate complexity, i.e. every certificate for w is of length of at least $C(f)$. Let S_1 be some *minimal* set of indices such that $f(w) \neq f(w^{(S_1)})$, let S_2 be another minimal set *disjoint* from S_1, such that $f(w) \neq f(w^{(S_2)})$, and in general we pick S_i to be a minimal set disjoint from all previous sets picked such that $f(w) \neq f(w^{(S_i)})$. We continue picking these sets until at a certain point no such set exists, say the last set was S_t.

The union of all sets has to be a certificate for w, since otherwise we could have picked yet another set that changes the value of the function when flipped. Thus we get that:

$$\sum_{i=1}^{t} |S_i| \geq C(f)$$

Now we can bound the block sensitivity of f in two ways:

(1) f is sensitive to each S_i on w, thus $bs_w(f) \geq t$.

(2) Since for each i, S_i is minimal then on $w^{(S_i)}$, f is sensitive to each element in S_i, thus $bs_{w^{(S_i)}}(f) \geq |S_i|$.

So if $t \geq \sqrt{C(f)}$ then (1) gives us the desired result, otherwise at least one of the sets has to be of size larger than $\sqrt{C(f)}$ and (2) will give us the result. ■

We conclude this section by mentioning the relation between the certificate complexity, and the boolean decision tree complexity. This result was independently discovered by several people, perhaps first by Blum. ([BI])

Lemma 2.5:

$$C(f) \leq D(f) \leq (C(f))^2$$

3. PRAM complexity

3.1. CREW PRAMs

Let $CREW(f)$ denote the parallel time needed to compute f on a CREW PRAM with an unbounded number of processors, each given arbitrary power. [CDR] gave a general lower bound for $CREW(f)$ in terms of the sensitivity of f. They showed that for all f:

$$CREW(f) \geq \log_{\alpha} s(f)$$

where α is some constant less than 5.

We first note that this result may be strengthened to give a bound in terms of the block sensitivity.

Lemma 3.1: for all f:

$$CREW(f) \geq \log_{\alpha} bs(f)$$

Proof: Let w be an input achieving the block sensitivity, and let $S_1, S_2, \cdots, S_t$ be the sets f is sensitive to on w. We define a new function $f'(X_1, X_2, \cdots, X_t)$ as follows: $f(X_1, \cdots, X_t)$ is equal to $f(w')$ where w' is derived from w by flipping all the bits in the set S_i for each i such that X_i=1. It is easy to see that f' instantly reduces to f on a CREW PRAM, and that the sensitivity of f' on the input 000...000 is t. thus

$$CREW(f) \geq CREW(f') \geq \log_{\alpha} s(f') \geq$$

$$\geq \log_{\alpha} t = \log_{\alpha} bs(f)$$

■

The surprising fact is that lemma 3.1 actually gives a tight lower bound for every function f! That will be shown using decision trees.

Lemma 3.2: A CREW PRAM can simulate a boolean decision tree of depth d in $\log_2 d$ time steps (using 2^d processors).

Proof: We will have a processor for each node of the decision tree. In the first step each processor will read the input variable that belongs to its node and set up a pointer to point to the node that should be followed by this node according to the value of the input. From now on in each step all the processors will use the standard "pointer doubling" method, and repeatedly copy the pointer of the node they're pointing to into their own pointer. It is easy to see that after $\log_2 d$ steps the root will point to the last leaf reached in the computation. ■

The only thing left to note is that the upper and lower bound that we gave are actually to within a constant factor from each other.

Theorem 1: For all f:

$$CREW(f) = \Theta(\log D(f)) = \Theta(\log bs(f))$$

Proof: Lemmas 2.4 and 2.5 show that $D(f)$ and $bs(f)$ are polynomially related to each other; thus the bounds given in lemmas 3.1 and 3.2 are within a constant factor of each other. ■

It should be noted that the upper bound simulation can be carried out by processors limited to a reasonable instruction set, and on memory cells that contain only 1 bit, while the lower bound derived in [CDR] holds regardless of the instruction set of the processors or the capacity of the memory cells. As a corollary we get that this model is insensitive to these issues as long as the number of processors is not limited.

3.2. CROW PRAMs

An extra bonus to be got from the previous proof is the equivalence in computation time between CREW PRAMs and the seemingly weaker CROW PRAMs. (The connection between CROW PRAMs and decision trees was also observed by [Ra]).

Let $CROW(f)$ be the time needed to compute f on an ideal CROW PRAM (with an unlimited number of processors) then we get:

Theorem 2: for any boolean function f:

$$CROW(f)=\Theta(CREW(f))$$

Proof: The simulation of decision trees described in the proof to lemma 3.2 can also be carried out by a CROW PRAM. ■

It is interesting to note that this result does *not* yield a simulation. The relation between CREW PRAM complexity and decision trees is a global one; there does not exist a correspondence between specific stages of the computation and parts of the decision tree. We can actually show that a CROW PRAM can not simulate a CREW PRAM step is a constant time! We exhibit a problem on a *partial* domain which separates these two models. Consider the following "promise" problem: Compute the OR of n input bits when you are "promised" that at most one input bit may be 1. A CREW PRAM with n processors can compute this function in 1 step; a CROW PRAM, however, can not compute this function quickly:

Lemma 2.3: A CROW PRAM requires time $\Omega(log\ n)$ to compute the OR function even on this partial domain.

Proof: We say a processor p *depends* on input location i at time t, if its state on the input consisting of all zeroes is different than its state on the input consisting of a 1 in location i and zeroes everywhere else, at time t. (The processors' state includes its private memory as well as all memory owned by it.) A simple induction shows that at time t each processor can depend on at most 2^t locations. this is so since in one time unit a processor may only read 1 memory cell, and after this time unit it can depend on at most anything it depended on in the last time step and anything the processor who owned this memory cell depended on last time step. Since the answer depends on all input bits, we get that $2^t \geq n$ when the algorithm terminates and the lemma follows. ■

This lemma does not contradict theorem 2, since theorem 2 only holds for functions on *full* domains.

4. Probabilistic vs. Deterministic decision trees

We first relate the one-sided error randomized complexity of a function to its deterministic complexity.

Theorem 3: For any boolean function f:

$$D(f)\leq 2(R_1(f))^2$$

This theorem will follow from the following more general lemma. Let $C^{(1)}(f)$ denote the certificate complexity of f limited to the 1-instances, i.e. the maximum of $C_w(f)$ over all $w \in f^{-1}(1)$.

Lemma 4.1: For any boolean function f:

$$D(f)\leq 2C^{(1)}(f)R_2(f)$$

Proof (Sketch): We will build a deterministic algorithm for f. The deterministic algorithm for f will start by picking any 1-certificate of f of size at most $C^1(f)$ and asking all the variables in it. After this first step we are left with an induced function on the remaining variables, and the algorithm will recursively solve this induced problem. Our claim is that after at most $2R_2(f)$ such stages the induced function would be a constant.

consider the randomized algorithm for f running on any 0-instance, w. The probability that this algorithm queries some variable in the 1-certificate must be at least 1/2. The reason is that without querying some bit in the 1-certificate, the algorithm can not dis-

tinguish between w, for which f is 0, and w with all the bits in the 1-certificate flipped to conform with the 1-certificate for which f is 1.

This shows that the first stage had already "shed" away 1/2 of a query from the expected running time of any 0-instance. Thus, after $2R_2(f)$ stages the running time for any 0-instance must be 0, so the function must be a constant. ■

The previous techniques do not carry over to the 2-sided error case. In order to give results here we will need to use our results concerning the block sensitivity. We first show that the block sensitivity can serve as a lower bound for the two-sided error randomized complexity of a function.

Lemma 4.2: For all boolean functions f:

$$R_2(f) \geq \frac{bs(f)}{2}$$

Proof: Let w be the input that achieves the block sensitivity, and let $S_1, S_2, \cdots, S_t$ be disjoint sets s.t. f is sensitive to S_i on w. For each $1 \leq i \leq t$, any randomized algorithm running on w must query some variable in S_i with probability of at least 1/2, since otherwise it can not distinguish between w and $w^{(S_i)}$. Thus the total expected time has to be at least $t/2$. ■

At this point, by combining lemmas 2.4, 4.1 and 4.2, we immediately get the following theorem:

Theorem 4: For any function f:

$$D(f) \leq 8R_2(f)^3$$

5. Acknowledgements

I would like to thank Russell Impagliazzo for his contributions to the paper, Amos Fiat, Valerie King and Moni Naor for many helpful discussions and Avi Wigderson for pointing out the application to CROW PRAMs.

6. References

[BI] M. Blum and R. Impagliazzo, "Generic oracles and oracle classes", 28th FOCS, 1987.

[CDR] S. Cook, C. Dwork and R. Reischuk, "Upper and lower bounds for parallel random access machines without simultaneous writes", Siam J. on Computing, Feb 1986.

[DR] P.W. Dymond and W.L. Ruzzo, "Parallel RAMs with owned global memory and deterministic context-free language recognition", ICALP, 1986.

[IN] R. Impagliazzo and M. Naor, "Decision trees and Downward closures", Structure in complexity conference, 1988.

[K] V. King, "The randomized complexity of graph properties", Manuscript, 1987.

[KK] D. J. Kleitman and K. J. Kwiatkowski, "Further results on the Aanderaa-Rosenberg conjecture", J. Combinatorial Theory (B) 28, 1980.

[KSS] J. Kahn, M. Saks and D. Sturtevant, "A topological approach to evasiveness", Combinatorica 4, 1984.

[Ra] P. Ragde, personal communication.

[Ro] A. L. Rosenberg, "On the time required to recognize properties of graphs: a problem", SIGACT news 5 #4, 1973.

[Ru] D. Rubinstein, personal communication.

[RV] R. Rivest and S. Vuillemin, "On recognizing graph properties from adjacency matrices", Theor. Comp. Sci. 3, 1978.

[Si] H. U. Simon, "A tight $\Omega(\log \log n)$ bound on the time for parallel RAM's to compute nondegenerate boolean functions", FCT'83, Lecture notes in Comp. Sci. 158, 1983.

[Sn] M. Snir, "Lower bounds for probabilistic linear decision trees", Theor. Comp. Sci. 38, 1985.

[SW] M. Saks and A. Wigderson, "Probabilistic boolean decision trees and the complexity of evaluating game trees", 27th FOCS, 1986.

[Ta] G. Tardos, "Query complexity, or Why is it difficult to seperate $NP^A \cap coNP^A$ from P^A by a random oracle A", manuscript, 1987.

[Tu] G. Turan, "The critical complexity of graph properties", Information Processing letters, 1984.

[W] I. Wegener, "The complexity of Boolean functions", pp. 373-410, John Wiley & sons and B.G. Teubner, Stuttgard, 1987.

[WZ] I. Wegener and L. Azdori, "A note on the relation between critical and sensitive complexity", manuscript.

[Y1] A. Yao, Probabilistic computations: towards a unified measure of complexity", 18th FOCS, 1977.

[Y2] A. Yao, "Lower bounds to randomized algorithms for graph properties", 28th FOCS, 1987.