August 1986 LIDS-P-1616

LOWER BOUNDS ON THE TIME TO COMPUTE A SIMPLE BOOLEAN FUNCTION ON A PARALLEL RANDOM ACCESS MACHINE[1]

Zhi–Quan Luo[2]
John N. Tsitsiklis[3]

ABSTRACT

We consider a parallel random access machine (PRAM) in which information is communicated via a shared memory. We strengthen a lower bound of Cook, Dwork and Reischuk for the time required to compute the Boolean "OR" function under some additional restrictions on the type of algorithms allowed. Then, we consider a related model in which a limited number of simultaneous writes is allowed and provide an algorithm for computing the Boolean "OR" which is optimal within the class of "non–adaptive" algorithms.

1. Research supported by the Army Research Office under Contract DAAAG–29–84–K–0005.
2. Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139.
3. Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139.

I. A LOWER BOUND FOR A RESTRICTED CLASS OF ALGORITHMS.

Consider a parallel random access machine in which concurrent reads but no concurrent writes are allowed (CREW–PRAM). (The reader is refered to [1–2] for a precise definition of this model. In fact the exact nature of the model used does not really matter, as long as simultaneous writes are prohibited.) Cook, Dwork and Reischuk [1–2] have proved that the time to compute the Boolean "OR" function of n arguments is at least $A \log_b n$, where $b = (5+21^{1/2})/2 \approx 4.8$ and A is a positive constant. We prove below that under some additional restrictions on the allowed algorithms, the lower bound may be improved to $A \log_a n$, where $a = 2 + 3^{1/2} \approx 3.732$.

We start with a few definitions. Let $\Sigma = \{0,1\}$, let n be a positive integer (which will be fixed from now on) and let $f : \Sigma^n \mapsto \Sigma$ be some function. For any $I = (x_1, ..., x_n) \in \Sigma^n$, let us denote by $I(j)$ the input $(y_1, ..., y_n) \in \Sigma^n$ satisfying $y_i = x_i$, $\forall i \neq j$, and $y_j = 1 - x_j$. We say that f is *n–sensitive somewhere* if there exists an input $I \in \Sigma^n$ such that for every $j \in \{1, ..., n\}$ we have $f(I(j)) \neq f(I)$. By letting $I = (0, ..., 0)$, we notice that the Boolean "OR" function is n–sensitive somewhere.

We say that an index $i \in \{1, ..., n\}$ ***influences processor p at time t with input $I \in \Sigma^n$*** if the state of processor p in the beginning of the t-th stage of the computation, given that the input is I, is different from the state of the same processor at the same time when the input is $I(i)$. We also define a corresponding concept of an index ***i influencing a memory cell c with input I*** in the obvious way.

Let $K(p,t,I)$ (respectively, $L(p,t,I)$) be the set of indices which influence processor p (respectively, cell c) at time t with input I and let $K(p,t) = \cup_{I \in \Sigma^n} K(p,t,I)$, $L(p,t) = \cup_{I \in \Sigma^n} L(p,t,I)$.

An algorithm on the CREW–PRAM which computes f is called ***proper*** if it has the following two properties:

a) For any processor p and any time t, the cell (if any) read by p at time t is the same for all $I \in \Sigma^n$.

b) For any cell c and any time t, there exists an input $I(t,c) \in \Sigma^n$ such that $L(p,t,I) \subset L(p,t,I(t,c))$, $\forall I \in \Sigma^n$.

Properness is the additional condition to be imposed on the class of algorithms that we study in this section. These conditions may seem restrictive; however, any parallel algorithm that has been proposed for the Boolean "OR" function (including the algorithms in [1–2]) does satisfy them, with $I(t,c) = (0, ..., 0)$.)

Theorem 1: A proper computation on a CREW–PRAM of a n–sensitive somewhere function requires at least $A \log_a n$ time, where $a = 2 + 3^{1/2} \approx 3.7$ and A is a positive constant.

Proof: We define P_t and C_t by the following recursions:

$$P_{t+1} = P_t + C_t \quad , P_0 = 0. \tag{1}$$

$$C_{t+1} = C_t + 2P_{t+1} \quad , C_0 = 1. \tag{2}$$

From these recursions, we easily obtain that $C_t \geq A(2+3^{1/2})^t$, where A is a positive constant.

We will show by induction that $|K(p,t)| \leq P_t$ and $|L(c,t,I)| \leq C_t$, for all p, t, I, c. These inequalities are trivially true for $t=0$ and we henceforth assume that they are valid for some t.
Step 1: Since for the PRAM model each processor may read at most one cell at each time and by part (a) of the properness assumption, it follows that for any t, p, there exists some cell $c(p,t)$ such that $K(p,t+1,I) \subset K(p,t,I) + L(c(p,t),t,I)$, $\forall I$. Therefore, using part (b) of the properness assumption and the induction hypothesis,

$$|K(p,t+1)| = |\cup_I K(p,t+1,I)| \leq |\cup_I K(p,t,I)| + |\cup_I L(c(p,t),t,I)|$$

$$\leq |K(p,t)| + |L(c(p,t),t,I(t,c(p,t)))| \leq P_t + C_t = P_{t+1}.$$

Step 2: Let t, I and a cell c be fixed. We distinguish two cases:
a) If, with input I, some processor p writes into cell c at time $t+1$, then, for that particular processor p we have $L(c,t+1,I) \subset K(p,t+1,I)$, from which we obtain $|L(c,t+1,I)| \leq |K(p,t+1,I)| \leq P_{t+1} \leq C_{t+1}$, as desired.
b) Assume now that, with input I, no processor writes into c at time $t+1$. We define $Y(c,t+1,I)$ as the set of indices $i \in \{1,...,n\}$ such that, with input $I(i)$, some processor writes into cell c at time $t+1$. We then obtain $|L(c,t+1,I)| \leq |L(c,t,I)| + |Y(c,t+1,I)| \leq C_t + |Y(c,t+1,I)|$. Thus, it only remains to show that $|Y(c,t+1,I)| \leq 2P_{t+1}$.

Let $r = |Y(c,t+1,I)|$ and let $Q_1,...,Q_k$ be the set of processors for which there exists some $i \in Y(c,t+1,I)$ such that processor Q_j writes into cell c at time $t+1$. Clearly, $0 \leq k \leq r$. For $j=1,...,k$, let Y_j be the set of indices $i \in Y(c,t+1,I)$ such that, with input $I(i)$, processor Q_j writes into c at time $t+1$. Let $r_j = |Y_j|$. Obviously, we have $r = \sum_{j=1}^k r_j$.

Let $X_{ij} = |Y_i \cap K(Q_j,t+1)|$. If every element of Y_i belongs to $K(Q_j,t+1)$, then $X_{ij} = r_i$. If this is not the case, let a be an index in Y_i which does not belong to $K(Q_j,t+1)$. With input $I(a)$, processor Q_i writes into cell c (at time $t+1$). Let b be an arbitrary element of Y_j. Processor Q_j writes into c with input $I(b)$ and, since $a \notin K(Q_j,t+1)$, processor Q_j also writes into c with input $I(a)(b)$. (Here $I(a,b)$ stands for the input I, with the entries at positions a and b reversed.) Therefore, processor Q_i does not write into c with input $I(a)(b)$, which shows that $b \in K(Q_i,t+1)$. Therefore, in this case we have $Y_j \subset K(Q_i,t+1)$ and $X_{ji} \geq r_j$. So, in either case we have

$$X_{ij} + X_{ji} \geq \min\{r_i, r_j\}.$$

Let $Z = \max_i\{r_i + \sum_{j=1,\ j\neq i}^k X_{ji}\}$.
Lemma: $Z \geq r/2$.
Proof of the Lemma: Let $U_{ii} = 1$ and $U_{ij} = X_{ji}/r_j$, for $j \neq i$. Thus, $U_{ij} + U_{ji} \geq 1$, for $i \neq j$. Now,

$$rZ = r \max_i\{\sum_{j=1}^k U_{ij} r_j\} \geq \sum_{i,j=1}^k U_{ij} r_i r_j =$$

$$\sum_{i<j} r_i r_j (U_{ij} + U_{ji}) + \sum_{i=1}^{k} r_i^2 \geq \sum_{i<j} r_i r_j + \sum_{i=1}^{k} r_i^2 =$$

$$\frac{1}{2} r^2 + \frac{1}{2} \sum_{i=1}^{k} r_i^2 \geq \frac{1}{2} r^2,$$

which completes the proof of the Lemma. •

We conclude that there exists some processor Q_i such that $r/2 \leq r_i + \sum_{j=1,\ j \neq i}^{k} X_{ji} = |Y(c, t+1, I) \cap K(Q_i, t+1)| \leq |K(Q_i, t+1)| \leq P_{t+1}$. Consequently, $|Y(c, t+1, I)| = r \leq 2P_{t+1}$, which concludes the inductive proof of the inequality $|L(c, t+1, I)| \leq C_{t+1}$.

The proof of the theorem is completed by using the fact that at the time that the computation terminates, there must exist some cell c and an input I such that $|L(c, t, I)| = n$, this being a consequence of the assumption that the function f being computed is n–sensitive somewhere. •

II. ALGORITHMS WITH A LIMITED NUMBER OF SIMULTANEOUS WRITES.

We now consider a slightly different model of computation. In particular, we assume again a PRAM model but we allow simultaneous writes, provided that at any time at most m processors may write simultaneously into the same cell, where m is a given constant. The results that follow are true no matter how write–conflicts are handled and for this reason we do not need to be more specific. We only assume that if a number of processors attempt to write the same value into a cell, this value is written correctly.

With this model it is rather hard to obtain lower bounds for general classes of algorithms, or even if we restrict to proper algorithms. For this reason, we impose a further restriction. We will consider only *non–adaptive* algorithms which are defined by the property that for any given time t and any memory cell c, the union, over all inputs I of the sets of processors who write into c at time t with input I, is bounded by m.

If $m = 1$, we have the PRAM model of the previous section, for which [1–2] have presented a non–adaptive algorithm which computes the Boolean "OR" function of n arguments in time $A \log_b n$, where $b = \frac{1}{2}(1 + 5^{1/2})$ and which is optimal, within the class of non–adaptive algorithms. We extend these results to the case where m simultaneous writes are allowed.

Algorithm (for computing the "OR" of n variables): Let $M(i)$ denote the contents of the i-th memory cell and let $Y(i)$ be a Boolean variable residing in the memory of the i-th processor, initialized at zero. Assume that the inputs $X(1), ..., X(n)$ are initially placed in the first n memory cells. At each time t, each processor i executes the following instructions:

```
a) $Y(i) \leftarrow Y(i) \vee M(i + P_t)$
b) If $i > (m-1)P_{t+1} + C_t$ and $Y(i) = 1$ then
for k=0 to $m-1$ do $M(i - kP_{t+1} - C_t) \leftarrow 1$.
```

Here, the quantities P_t and C_t are defined by the recursions:

$P_{t+1} = P_t + C_t; \quad P_0 = 0;$ (3)

$C_{t+1} = C_t + mP_{t+1}; \quad C_0 = 1.$ (4)

It is easy to see that this is a non–adaptive algorithm and that every cell is written by at most m processors so that our requirement is met.

Claim: At any time t, we have

$Y(i) = X(i) \vee ... \vee X(i + P_t - 1),$

$M(i) = X(i) \vee ... \vee X(i + C_t - 1).$

Proof of the Claim: This is obviously true for time $t = 0$. Suppose it is true at time t. Then,

$$Y(i) \leftarrow Y(i) \vee M(i + P_t) = (\vee_{k=i}^{i+P_t-1} X(k)) \vee (\vee_{k=i+P_t}^{i+P_t+C_t-1} X(k)) =$$

$$\vee_{k=i}^{i+P_{t+1}-1} X(k),$$

where we have used equation (3). Also,

$$M(i) \leftarrow M(i) \vee Y(i + C_t) \vee Y(i + P_{t+1} + C_t) \vee ... \vee Y(i + (m-1)P_{t+1} + C_t) =$$

$$X(i) \vee ... \vee X(i + mP_{t+1} + C_t) = X(i) \vee ... X(i + C_{t+1}),$$

where we have used equation (4). •

By solving the recursions for C_t and P_t we see that $C_t \approx b(m)^t$, where

$$b(m) = [(m + 2 + (m^2 + 4m)^{1/2})/2]. \quad (5)$$

Therefore, the time required for this algorithm to compute the "OR" of n arguments is approximately $\log_{b(m)} n$. Notice that $b(m)/m$ converges to 1, as $m \to \infty$.

Next we show that the above algorithm is optimal within the class of non–adaptive algorithms.

Theorem 2: Any non–adaptive algorithm which computes a n–sensitive somewhere function, with at most m simultaneous writes allowed, requires at least $A \log_{b(m)} n$ steps, with $b(m)$ defined in (5).

Proof: The proof differs from the proof of Theorem 1 only in case (b) of step 2. We thus consider only this case and we keep the notation of that proof. Let $A(c, t+1)$ denote the set of processors who can write into cell c at time $t + 1$. By our assumptions, $|A(c, t+1)| \leq m$. Furthermore,

$$Y(c, t+1, I) \subset \cup_{p \in A(c,t+1)} K(p, t+1, I).$$

Therefore,

$$|Y(c, t+1) \leq |A(c, t+1)| \max_p |K(p, t+1, I)| \leq mP_{t+1}.$$

We then use the inequality $|L(c, t+1, I)| \leq |L(c, t, I)| + |Y(c, t+1, I)|$, to obtain $C_{t+1} \leq C_t + mP_{t+1}$.

Therefore, for any time t and for any input I, at most $C_t \approx b(m)^t$ input indices may influence the contents of a memory cell. The result follows because there exists some input I for which the output cell at termination time is influenced by n input indices (this being a consequence of the assumption that we are computing a somewhere n–sensitive function). •

REFERENCES

1. Cook, S.A., and Dwork C., "Bounds on the time for parallel RAM's to compute simple functions" 14th STOC, 1982, pp. 231-233.

2. Cook, S.A., Dwork, C., and Reischuk, R., "Upper and lower bounds for parallel random access machines without simultaneous writes", *SIAM J. on Computing,* 15, 1,1986, pp. 87–97.