Lower Bounds for Randomized Exclusive Write PRAMs

Philip D. MacKenzie*
Sandia National Laboratories
Albuquerque, NM 87185-1110
philmac@cs.sandia.gov

May 2, 1995

Abstract

In this paper we study the question: How useful is randomization in speeding up Exclusive Write PRAM computations? Our results give further evidence that randomization is of limited use in these types of computations. First we examine a compaction problem on both the CREW and EREW PRAM models, and we present randomized lower bounds which match the best deterministic lower bounds known. (For the CREW PRAM model, the lower bound is asymptotically optimal.) These are the first non-trivial randomized lower bounds known for the compaction problem on these models. We show that our lower bounds also apply to the problem of approximate compaction. Next we examine the problem of computing boolean functions on the CREW PRAM model, and we present a randomized lower bound which improves on the previous best randomized lower bound for many boolean functions, including the OR function. (The previous lower bounds for these functions were asymptotically optimal, but we improve the constant multiplicative factor.) We also give an alternate proof for the randomized lower bound on PARITY, which was already optimal to within a constant additive factor. Lastly, we give a randomized lower bound for integer merging on an EREW PRAM which matches the best deterministic lower bound known. In all our proofs, we use the Random Adversary method, which has previously only been used for proving lower bounds on models with Concurrent Write capabilities. Thus this paper also serves to illustrate the power and generality of this method for proving parallel randomized lower bounds.

1 Introduction

Randomization has been a useful tool in developing fast parallel algorithms for a vast spectrum of problems, from computational geometry and graph theory, to routing and load balancing. Often these randomized parallel algorithms are significantly faster than the best possible deterministic parallel algorithms. This prompts the question, "To what extent can randomization improve the speed of parallel algorithms?" We will examine this question in regards to an important class of parallel algorithms, namely Exclusive Write PRAM algorithms. We will show that randomization does not seem to be very effective when Concurrent Writing is not allowed. To do this, we will prove lower bounds for randomized Exclusive Write PRAM algorithms. These lower bounds are (1) within

*Part of this work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under contract DE-AC04-76DP00789. Part of this work was performed while the author was at the University of Texas and was supported by TARP Grant 003658480.

a constant factor from the the best deterministic lower bounds known, and (2) for all problems but that of compaction on the EREW PRAM, within a constant factor from the deterministic upper bound.

The tool that will allow us to prove these randomized lower bounds is the Random Adversary technique. This technique is an extension and generalization of the Random Restriction technique first used in Furst, Saxe, and Sipser [7]. The Random Adversary technique was used to prove lower bounds on randomized CRCW PRAM algorithms in MacKenzie [17] and on randomized OCPC algorithms in Goldberg, Jerrum and MacKenzie [10]. However, this is the first application of the technique to Exclusive Write PRAM algorithms.

The Random Adversary technique requires a specific proof structure. Assuming there exists a deterministic lower bound for a problem with this structure, a randomized lower bound sometimes can be obtained in a relatively straightforward manner using the Random Adversary technique. This was in fact the case for our lower bounds on boolean functions. However, if the deterministic lower bound proof does not have this structure, a completely new lower bound proof has to be developed. This was the case for the compaction lower bound, as will be discussed below.

1.1 Compaction

Compaction algorithms on PRAMs have been studied in [5, 9, 14, 21] and are useful for performing load balancing operations, space allocation, and a variety of other tasks. In this paper, we look at the problem of k-compaction, that is, given an array of n items with at most k of them marked, moving the marked items to the front of the array. On the Common CRCW PRAM model, it is known how to solve this problem deterministically in $O(\log k/\log\log n)$ steps using n processors, and there is a matching lower bound, even for randomized algorithms [21]. On Exclusive Write PRAMs, much less is known. For the case $k = 2$, on the CREW PRAM model, it is known how to solve this problem deterministically in $O(\log\log n)$ steps using n processors, and there is a lower bound of $\Omega(\log\log n)$ steps when using n processors [5]. For the case $k = 2$, on the EREW PRAM model, it is known how to solve this problem deterministically in $O(\log n)$ steps using n processors, and there is a lower bound of $\Omega(\sqrt{\log n})$ when using any number of processors [5]. However, until now, there was no non-trivial lower bound known for randomized Exclusive Write PRAM algorithms for the case $k = 2$. (Note that a lower bound on this case would apply to any $k \geq 2$.)

We prove randomized lower bounds on 2-compaction which match the deterministic lower bounds for the CREW and EREW PRAM. In fact, we prove these lower bounds hold for approximate 2-compaction also, where for some constant γ $(0 < \gamma < 1)$, the 2 marked items simply have to be placed in an array of size $n^{1-\gamma}$. (Approximate compaction has been studied in [2, 8, 11, 12, 17], among others. Note that on the CRCW PRAM, the complexity of compaction problems and approximate compaction problems can be vastly different.)

As mentioned above, the deterministic lower bound proofs in Fich et al. [5] for compaction on Exclusive Write PRAM models do not have the structure required by the Random Adversary technique to admit a randomized lower bound proof. In essence, they are able to perform the lower bound analysis by having a deterministic adversary set many inputs on each step, until only a very few inputs are left. However, for the Random Adversary technique to work, the Random Adversary must not set too many inputs, or else the probability of fixing the result becomes too large. When not as many inputs are set, though, the unset inputs can affect processors and cells in much more complicated ways, and the analysis of Fich et al. breaks down. Thus it was necessary to build a completely new proof, based in part on concepts in Cook, Dwork, and Reischuk [3] (bounding the number of inputs affecting processors and cells for given input maps), in part on concepts found in

many other PRAM lower bound proofs (bounding the number of inputs affecting processors and cells for many or all input maps, and having an adversary set some of the inputs), and in part on the concept of "discarded input maps", which to our knowledge has not been previously used.

1.2 Boolean Functions on Exclusive Write PRAMs

There has been a considerable amount of research in the area of computing boolean functions on the CREW PRAM, mostly with respect to deterministic algorithms. To describe the results, we need the following definitions. Let $a_1, \ldots, a_n$ denote elements of $\{0,1\}$ and let $a = (a_1, \ldots, a_n)$ denote an element of $\{0,1\}^n$. Let B_n denote the set of all Boolean functions on n variables $\{f | f : \{0,1\}^n \rightarrow \{0,1\}\}$. For $f \in B_n$ define the *critical complexity* $c(f)$ of f as $\max\left\{\#\{i | f(a^{\{i\}}) \neq f(a)\} | a \in \{0,1\}^n\right\}$, where $a^{\{i\}} = (a_1, \ldots, \overline{a_i}, \ldots, a_n)$. For $f \in B_n$ define the block sensitivity $bs(f)$ as the maximum of the numbers $\max\{l | \exists S_1, \ldots, S_l$ disjoint such that $f(a^{S_j}) \neq f(a)$, $1 \leq j \leq l\}$ taken over all input vectors a, where a^S is obtained from a by flipping all bits in positions $i \in S$.

First, Cook, Dwork, and Reischuk [3] obtain a lower bound of about $.45 \log c(f)$ for computing a function f with critical complexity $c(f)$. (For instance, the OR function on n inputs has critical complexity n.) Parberry and Yan [20] improve this to $.5 \log c(f)$. Nisan [18] then shows that computing a function f on a CREW PRAM takes $\Theta(\log bs(f))$ time (actually the upper bound applies to an "ideal CREW PRAM"). In particular, it takes at least $.5 \log bs(f)$ time. Thus, the question of CREW PRAM complexity of functions was known up to a constant multiplicative factor. Kutylowski [16] was the first to show a lower bound for OR of $\phi(n) \geq .72 \log n$ which was tight up to a constant additive factor for the ideal CREW PRAM. Dietzfelbinger, Kutylowski and Reischuk [4] generalize this result and show a lower bound of $\phi(\deg(f))$ where $\deg(f)$ is the degree of the polynomial (over the integers) representing f. In particular, this implies a tight lower bound for both OR and PARITY of $\phi(n)$. They also were the first to obtain randomized lower bounds for computing boolean functions on the CREW PRAM model. They show that computing PARITY with a randomized CREW PRAM algorithm requires exactly as long as with a deterministic CREW PRAM algorithm, and computing any other boolean functions using a randomized CREW PRAM algorithm requires at least 1/8th as long as using a deterministic CREW PRAM algorithm (specifically, for a boolean function f, the lower bound they show is $\phi(\sqrt{bs(f)}) - O(1)$). For symmetric functions, they are able to tighten this bound. For symmetric f, and for $0 \leq k \leq n$, let $f_k = 1$ if $f(a) = 1$ for all input vectors a with exactly k 1's, and let $f_k = 0$ otherwise. Let $\gamma(f) = \max\{k : 1 \leq k \leq \frac{1}{2}n$ and $f_k \neq f_{k+1}$ or $f_{n-k+1} \neq f_{n-k}\}$. They obtain a lower bound of $\phi(\sqrt{n\gamma(f)}) - O(1)$ for computing f on a randomized CREW PRAM. Many important functions, such as MAJORITY and MOD_m satisfy $\gamma(f) = \Omega(n)$, so for these functions they obtain a lower bound of $\phi(n) - O(1)$, which is optimal to within a constant additive factor.

We prove a randomized lower bound of $.5 \log bs(f) - O(1)$ for all boolean functions f. Note that this improves the lower bound of $.36 \log bs(f) - O(1)$ for general boolean functions which was proven in [4]. It also improves on the lower bound of $\frac{1}{24} \log(\deg(f)) - O(1)$ given in [4] since $\deg(f) \leq (bs(f))^4$ (see Fact 7.3). Our lower bound is obtained by combining the lower bound of Parberry and Yan [20] with the Random Adversary technique.

To display the generality of the Random Adversary technique, we also show how it can be combined with the deterministic results of [4] to obtain an optimal lower bound for computing PARITY on a randomized CREW PRAM. They were also able to prove this randomized lower bound, but their proof is fundamentally different.

1.3 Integer merging on the EREW PRAM

Our last application of the Random Adversary technique is to obtain a lower bound for integer merging on the EREW PRAM. This problem was studied in the deterministic case by Hagerup and Kutylowski [13], who show the surprising result that merging two sorted arrays of bits can be accomplished in $O(\log\log n)$ time. They also show that merging two sorted arrays of integers in the range $\{0,\ldots,m-1\}$ requires $\Omega(\log\min\{m,n\})$ steps. We prove that this lower bound holds for randomized algorithms also.

2 Definitions

In the PRAM model, processors communicate by reading and writing to a global shared memory. The PRAM model is further subdivided depending on whether concurrent accesses are allowed to memory on reads and/or writes. An Exclusive Read (ER) model does not allow concurrent reads to a memory cell, whereas a concurrent read (CR) model does allow concurrent reads. An Exclusive Write (EW) model does not allow concurrent writes to a memory cell, whereas a concurrent write (CW) model does allow concurrent writes. We will only deal with EW models in this paper. For further information on CW models see, for example, [15].

We define a *randomized algorithm* as one in which each processor can generate some number of random bits. In our lower bounds, we make no assumption on the number of random bits a processor can generate. Most of our lower bounds will take the form "After t steps, any algorithm allegedly computing f succeeds with probability at most $\frac{1}{2}(1+h(n))$", for some h. (This probability is taken over the random bits.) Note that this implies a lower bound on the number of steps required for an algorithm to do significantly better than guessing.

Given a randomized algorithm, or assuming a random input distribution to an algorithm, in the ER (EW) model of the PRAM, we assume that concurrent reads (writes) can only occur with probability 0.

3 Random Adversary Technique

The Random Adversary Technique allows one to prove a lower bound on the time required for a parallel randomized algorithm to solve a given problem. The first step of the technique is to decide on an input distribution for the problem. By Yao's Theorem (see below), a lower bound on deterministic algorithms over this distribution provides the same lower bound for randomized algorithms.

The next step is to create a Random Adversary that proceeds through the given deterministic algorithm step by step, fixing some of the inputs in order to ensure some desired properties. (As shown below, this entails filling in the details of a procedure called REFINE.) Note that the Random Adversary is similar to a standard deterministic adversary in most parallel lower bound proofs. However, unlike deterministic adversaries that can fix inputs arbitrarily, the Random Adversary must fix inputs according to the chosen input distribution, i.e., using the procedure RANDOMSET, as described below. Also, depending on how RANDOMSET fixes the inputs, the desired properties might not hold. Therefore, it is possible that the Random Adversary might have to make repeated calls to RANDOMSET to ensure the desired properties.

The final step is to show that these desired properties (such as knowledge about the inputs still being widely dispersed among the processors, and that the number of inputs left unset is still large) hold with some given probability.

In the rest of this section we formalize this method.

3.1 Definitions

Let P be a problem and I the set of inputs to P. Let Q be the set of possible values to which each input could be set. Define a *partial input map* to be a function f from I to $\{\{*\} \cup Q\}$. Here '$*$' will denote a "blank" or "unset" input. A partial input map is an *input map* if no inputs are mapped to '$*$'. Let f_* denote the partial input map which maps every input to '$*$'. A partial input map f' is called a *refinement* of a partial input map f if for all $i \in I$, and $q \in Q$, $f(i) = q$ implies $f'(i) = q$. (We denote this by $f' \leq f$.) If we wish to restrict our attention to a subset of the possible input maps, we would call that subset the *relevant input maps*. Likewise, we would say that a partial input map is a *partial relevant input map* if it has a refinement that is a relevant input map. We will often omit the word relevant, when it is clear from the context.

3.2 Yao's Theorem

The following theorem shows that an upper bound on the success probability of a deterministic algorithm over random inputs implies the same upper bound for the success probability of any randomized algorithm over a worst case input. The theorem is similar to one given in Yao [24].

Theorem 3.1 *Let S_1 be the success probability of a T step randomized algorithm solving problem P, where the success probability is taken over the random choices made by the algorithm, and minimized over all possible inputs. Let S_2 be the success probability over a distribution $\mathcal{D}$ of inputs, maximized over all possible T step deterministic algorithms to solve P. Then $S_1 \leq S_2$.*

Proof: (This proof is modeled after the proof of Yao's theorem in [6].) We can consider a randomized algorithm as randomly choosing a deterministic algorithm to run. Let A be a set of deterministic algorithms, and assume each $A_i \in A$ is chosen with some probability p_i. Let $\mathcal{D}$ be the input distribution where each input D_j is chosen with probability q_j. Let $s(A_i, D_j)$ be 1 if A_i successfully solves P on input D_j, and otherwise 0. Then

$$\begin{aligned}
S_1 &= \min_{D_j \in \mathcal{D}} \left(\sum_{A_i \in A} p_i \cdot s(A_i, D_j) \right) \\
&\leq \sum_{D_j \in \mathcal{D}} q_j \sum_{A_i \in A} p_i \cdot s(A_i, D_j) \\
&= \sum_{A_j \in A} p_i \sum_{D_j \in \mathcal{D}} q_j \cdot s(A_i, D_j) \\
&\leq \max_{A_j \in A} \left(\sum_{D_j \in \mathcal{D}} q_j \cdot s(A_i, D_j) \right) \\
&= S_2.
\end{aligned}$$

□

This theorem greatly simplifies the problem of proving randomized lower bounds, as it converts the original problem into one where the only randomness comes from the input distribution. Furthermore, this input distribution can be set as one wishes. It is of course necessary to choose a distribution that will be difficult for any deterministic algorithm. Note that the input distribution cannot place all the probability on one input map (i.e. a "worst case" input map), since then a simple deterministic algorithm which outputs a precomputed answer to this input map will succeed with probability 1.

3.3 RandomSet Procedure

We will assume the distribution chosen is $\mathcal{D}$. Function RANDOMSET can be used to randomly generate an input map one input at a time. It is called with a partial input map f obtained through calls to RANDOMSET, and a set S of elements which are mapped to '$*$'. The elements in S are then randomly set one by one according to the $\mathcal{D}$, conditional on f.

Function RANDOMSET(f, S)
For each $i \in S$
    Set $f(i)$ according to the conditional distribution of i given that
        the input is drawn from $\mathcal{D}$ and is a refinement of f
Return f
End RANDOMSET

Claim 3.1 *Assuming f generated solely by calls to RANDOMSET, then f will be generated according to the distribution $\mathcal{D}$.*

Proof: Straightforward. □

3.4 REFINE and GENERATE

Say f is t-good if it satisfies certain properties, which will be defined with respect to the problem P and the input distribution $\mathcal{D}$. Say $T \leq n$ is the number of steps that we are trying to show is a lower bound for solving the problem P. Let A be an algorithm which allegely solves problem P over the input distribution $\mathcal{D}$ in T steps.

Given this algorithm A, we create a procedure REFINE which tells the Random Adversary how to fix the inputs at each step. Formally, REFINE(t, f) takes a time t and a partial input map f and returns a new partial input map f' that is a refinement of f. We need to prove that the procedure REFINE has two important properties, the first of which is concerned with preservation of "t-goodness". Consider the function GENERATE defined below that starts with the partial input map $f_0 = f_*$, and applies REFINE T times to generate a sequence of partial input maps $f_0 = f_* \geq f_1 \geq \cdots \geq f_T \geq f$ in which each $f_t = \text{REFINE}(t, f_{t-1})$ is a refinement of f_{t-1}, Then for some target probability Z, we need to prove the following.

Lemma 3.1 *With probability Z, for every t ($0 \leq t \leq T$), f_t is t-good.*

The second property is that REFINE is unbiased. Using the same specification as before, and assuming f is an input map generated according to the conditional distribution over $\mathcal{D}$ from the set of refinements of f_T, we need to prove the following lemma.

Lemma 3.2 *The input map f returned by* GENERATE *is generated according to the distribution $\mathcal{D}$.*

In all REFINE procedures we construct in this paper, all inputs are set by calls to RANDOMSET. Consequently, by Claim 3.1, Lemma 3.2 will always hold.

Function GENERATE
    Let $f_0 = f_*$
    Let $f = f_0$
    Let $t = 1$

```
While t ≤ T Do
    If for some p, f(p) = '*' Then
        Let f_t = REFINE(t, f)
    Else
        Let f_t = f
    f = f_t
    t = t + 1
Let P = {p | f(p) = '*'}
Return RANDOMSET(f, P)
End GENERATE
```

In summary, to fill in the Random Adversary framework for a specific problem P, we must specify

1. an input distribution $\mathcal{D}$,
2. a definition for t-good,
3. a function REFINE,
4. a time T,
5. a target probability Z, and
6. a proof for Lemma 3.1.

4 Definitions for the Compaction problem

We would like to prove a lower bound on the time required to solve the problem of 2-Compaction on the CREW PRAM and EREW PRAM models. To this end, Fich et al. [5] show that it is sufficient to prove a lower bound on the 2-OR problem, that is, the problem of computing the OR of n bits, restricted to the case when at most 2 of the n bits are allowed to take the value 1. We will use this approach to prove the lower bound on 2-Compaction. Thus we will have binary inputs (i.e. $Q = \{0, 1\}$), and we will say an input map f is *relevant* if at most 2 inputs are mapped to '1'. Let f_0 be the input map which maps all inputs to 0. Let f_i $(1 \leq i \leq n)$ be the input map which maps all inputs to 0 except i, which is mapped to 1. Let $f_{i,j}$ $(1 \leq i < j \leq n)$ be the input map which maps all inputs to 0 except i and j, which are mapped to 1. Let $F_{\leq 1} = \{f_i : 0 \leq i \leq n\}$.

The input distribution we use can be described as follows. Assume $\tilde{f}$ is the random variable denoting the input map chosen. Then

$$\begin{aligned}
\Pr(\tilde{f} = f_0) &= \frac{1}{2} \\
\Pr(\tilde{f} = f_i) &= \frac{1}{2n} - \frac{\epsilon}{4n}, \text{ for } 1 \leq i \leq n \\
\Pr(\tilde{f} = f_{i,j}) &= \frac{\epsilon}{4}\binom{n}{2}^{-1}, \text{ for } 1 \leq i < j \leq n.
\end{aligned}$$

(Notice that the random input map must have a distribution for which the output of the problem (in this case OR) is zero or one with roughly equal probability. Otherwise, a trivial program could output the correct answer with probability significantly greater than $\frac{1}{2}$.)

For an input map f and an $i \in [1,n]$, define the input map $f_{(i)}$ to be the input map f with the ith bit flipped, i.e.,

$$f_{(i)}(j) = \begin{cases} f(j) & \text{if } j \neq i \\ 1 - f(j) & \text{if } j = i. \end{cases}$$

We say an input map f is *pertinent* to i if both f and $f_{(i)}$ are relevant.

We assume a deterministic algorithm A to compute 2-OR is run on the PRAM. We assume the inputs are stored in the first n memory locations (i.e., for an input map f, and $1 \leq i \leq n$, cell i contains $f(i)$) and the output is contained in the first cell.

Say the state of a processor p at time t with f is the $t+1$-tuple $(p, v_1, v_2, \ldots, v_t)$, where v_i is the value read by p in step i, or "nil" if p does not read any cell in step i.

Say input i *affects* a processor p (respectively, a cell c) at time t with f iff f and $f_{(i)}$ are relevant and the state of p (contents of c) at t with $f_{(i)}$ differs from the state of p (contents of c) at t with f. Note that by the definition of state, if an input i affects p at t with f, then it affects p at t' with f for all $t' \geq t$.

Say input i *causes* processor p to write into c at time t with f iff f and $f_{(i)}$ are relevant, p does not write into c at t with f, but p does write into c at t with $f_{(i)}$.

In the following, the letter f will always denote an input map, and the letter g will denote a partial input map. Let $G_0 = \{g : \text{for all inputs } i,\ g(i) \neq 1\}$. Let Unset$(g)$ be the set of inputs mapped to '$*$' by g.

5 Compaction on the CREW PRAM

For the CREW PRAM lower bound proof, we must assume there are n processors. (Note that when the number of processors is n^2, the problem can be solved trivially in constant time.) We have already defined the input distribution. In this section we will define the notion of t-good, the procedure REFINE, and the values of T and Z. Then we will prove that REFINE preserves the t-goodness property, and that the desired lower bound holds. To define t-good, we first need to define the following sets.

- Let $P(p,t,f,g)$ where $f \leq g$, be the set of inputs in Unset(g) which affect processor p at t with f.
- Let $C(p,t,f,g)$ where $f \leq g$, be the set of inputs in Unset(g) which affect cell c at t with f.
- Let $MV(i,0,g)$ be the empty set, and let $MV(i,t+1,g)$ be defined inductively as the set of processors in $MV(i,t,g)$ plus the set of processors that read a cell in $MW(i,t,g)$ at $t+1$ with some $f \leq g$ in $F_{\leq 1}$.
- Let $MW(i,0,g) = \{i\}$ if $1 \leq i \leq n$ and $g(i) =$ '$*$', else let $MW(i,0,g)$ be the empty set. Let $MW(i,t+1,g)$ be defined inductively as the set of cells in $MW(i,t,g)$ plus the set of cells that are written to by a processor $p \in MV(i,t+1,g)$ with some $f \leq g$ pertinent to i but not in $D(i,t+1,g)$.
- Let $D(i,t,g)$ be a set of "discarded" input maps for an input i. Let

  $$S_{i,p,t,g} = (P(p,t+1,f_0,g) \cup P(p,t+1,f_i,g)) \setminus \{i\}.$$

  Let $D(i,0,g)$ be the empty set, and let $D(i,t+1,g)$ be defined to be, for all $p \in MV(i,t+1,g)$, the inputs maps f_j $(j \in S_{i,p,t,g})$ and $f_{i,j}$ $(j \in S_{i,p,t,g})$.
- $MP(p,t,g) = \{i : p \in MV(i,t,g)\}$

- $MC(c,t,g) = \{i : c \in MW(i,t,g)\}$

The sets $P(p,t,f,g)$ and $C(c,t,f,g)$ correspond to sets with the same names in Cook, Dwork, and Reischuk [3]. If all input maps were relevant, these would be the only sets necessary to prove the desired lower bound. However, the method used in [3] to bound $|C(c,t,f,g)|$ breaks down when not all input maps are relevant. By defining the remaining sets (and bounding their sizes), we provide a new way to bound $|C(c,t,f,g)|$, and thus achieve our lower bound.

Here we give some intuition behind the definitions of the other sets. $MV(i,t,g)$ should be thought of as the processors that are affected by input i for most input maps, and $MW(i,t,g)$ should be thought of as the cells that are affected by input i for most input maps. $MP(p,t,g)$ should be thought of as the inputs that affect processor p for most input maps, and $MC(c,t,g)$ should be thought of as the inputs that affect cell c for most input maps.

The difficulty in this proof comes in dealing with the fact that we can only bound the sizes of the sets $MV(i,t,g)$, $MW(i,t,g)$, $MP(p,t,g)$ and $MC(c,t,g)$ for most input maps, but not all inputs maps. For instance, if processor p reads inputs 1 and 2, and finds they are both equal to 1, then p would know the full input map (i.e. $MP(p,t,g) = n$). Thus we must not consider this input map when bounding $MP(p,t,g)$. These "discarded" input maps are put into $D(i,t,g)$, for appropriate inputs i. In this example, we would add f_2 and $f_{1,2}$ to $D(1,t,g)$, and we would add f_1 and $f_{1,2}$ to $D(2,t,g)$.

Now define the following constants (with $K \geq 0$ to be chosen later) used to bound the sizes of the corresponding sets above:

- $P_0 = K$, $P_{t+1} = P_t + C_t$
- $C_0 = K + 1$, $C_{t+1} = C_t + 2D_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$.
- $D_0 = 0$, $D_{t+1} = 4MV_{t+1}P_{t+1}$
- $MV_0 = 0$, $MV_{t+1} = MV_t + 2MW_t(P_t + 1)^2 MC_t$
- $MW_0 = 0$, $MW_{t+1} = MW_t + 2MV_{t+1}(P_{t+1} + 1)$
- $MP_0 = 0$, $MP_{t+1} = MP_t + MC_t(P_t + 1)$
- $MC_0 = 1$, $MC_{t+1} = MC_t + 3P_{t+1} + MP_{t+1}$

The following claim gives a bound on the rate of growth of these constants.

Claim 5.1 *For $t \geq 1$ and some constant $\alpha \geq 0$, $C_t \leq K^{\alpha^t}$.*

Proof: First note that for all $t \geq 0$, $P_t \leq C_t$, $MV_t \leq MW_t$, $MP_t \leq MC_t$, and $P_{t+1} \leq 2C_t$. From these we can see that $C_{t+1} \leq O(C_t MW_t)$, $MW_{t+1} \leq O(MW_t C_t^3 MC_t)$, $MC_{t+1} \leq O(MC_t C_t)$, and $C_{t+2} \geq C_t^2$. Using these inequalities, we can show that for some given constants k, k', k'', $MC_{t+1} \leq k^t C_t^4$, $MW_{t+1} \leq (k')^{t^2} C_t^{28}$, and $C_{t+1} \leq (k'')^{t^3} C_t^{116}$. The last inequality implies that $C_t \leq K^{\alpha^t}$, for some $\alpha \geq 0$. (Note: no attempt has been made to optimize the value of α.) □

Say g is t-good if

1. $g \in G_0$,
2. for all processors p, cells c, inputs i, and input maps $f \leq g$, $|P(p,t,f,g)| \leq P_t$, $|C(c,t,f,g)| \leq C_t$, $|MV(i,t,g)| \leq MV_t$, $|MW(i,t,g)| \leq MW_t$, $|D(i,t,g)| \leq D_t$, $|MP(p,t,g)| \leq MP_t$, $|MC(c,t,g)| \leq MC_t$, and
3. $\text{Unset}(g) \geq n - \frac{n}{K} \sum_{i=1}^{t} 2^{-t}$,

Now assume g is the partial input map produced after the first t steps. Then in step $t+1$ let C_p be the set of cells read by a processor p for input maps $f \in F_{\leq 1}$ where $f \leq g$. Then $|C_p| \leq |P(p,t,f_0,g)| + 1 \leq P_t + 1$. Let $K_t = P_t + 1$.

Consider the nK_t possible reads by all processors on inputs in $F_{\leq 1}$. Let L contain each cell that has over $2K_t^2 MC_t$ possible reads to it. We then define the function REFINE as follows. (Basically, REFINE will set inputs such that the value of each cell in L becomes fixed. Thus large concurrent reads do not transfer too much information about unset inputs.)

Function REFINE(t,g)
(1) Let $g'' = g$
(2) For each $c \in L$
(3) Let $g'' =$ RANDOMSET$(g'', MC(c,t,g''))$
(4) Next c
(5) Let $g' = g''$
(6) Return g'
End REFINE

The following is a useful technical fact that we will use throughout this section.

Claim 5.2 *If $g' \leq g$ and $f \leq g'$, (1) $P(p,t,f,g') \subseteq P(p,t,f,g)$, (2) $C(c,t,f,g') \subseteq C(c,t,f,g)$. (3) $MV(i,t,g') \subseteq MV(i,t,g)$. (4) $MW(i,t,g') \subseteq MW(i,t,g)$. (5) $D(i,t,g') \subseteq D(i,t,g)$. (6) $MP(p,t,g') \subseteq MP(p,t,g)$. (7) $MC(c,t,g') \subseteq MC(c,t,g)$.*

Proof: The proof for (1) and (2) is immediate.

The proofs for (3), (4), and (5) can be handled together by induction. Note that the case $t = 0$ is true.. Now assume (3), (4), and (5) are true for some t. We can prove that they hold for $t+1$ as follows. (3) follows directly by induction. (5) follows directly from (1) and (3). To prove (4) holds, consider a cell $c \in MW(i,t+1,g')$. If $c \in MW(i,t,g')$, then $c \in MW(i,t,g) \subseteq MW(i,t+1,g)$. Otherwise, c is written to by some $p \in MV(i,t+1,g')$ for some $f \leq g'$ pertinent to i but not in $D(i,t+1,g')$. Consider the case when $f \neq f_0$ and $f \neq f_i$. Then for some $j \neq i$, $f = f_j$ or $f = f_{i,j}$. Since f is not in $D(i,t+1,g')$, j is not in $P(p,t,f_0,g')$ or $P(p,t,f_i,g')$, and thus p also writes to c with f_0 or f_i. From (3), $p \in MV(i,t+1,g)$. Also, neither f_0 nor f_i is in $D(i,t+1,g)$. Thus $c \in MW(i,t+1,g)$, and (4) holds.

(6) and (7) follow from (4) and (5), respectively. □

The following claim shows the important relationship between the sets defined above, namely if a processor p (cell c) is affected by input i with f, then either f is "discarded" or i affects p (c) for most input maps. (This is the link we need in our new method of bounding $|C(c,t,f,g)|$ in Claim 5.7.)

Claim 5.3 *For all p,c,f,g,t,i with $f \leq g$, if $i \in P(p,t,f,g)$, then either $f \in D(i,t,g)$ or $p \in MV(i,t,g)$; also, if $i \in C(c,t,f,g)$, then either $f \in D(i,t,g)$ or $c \in MW(i,t,g)$.*

Proof: By induction. The case for $t = 0$ is straightforward.

For the first part of the claim, assume $i \in P(p,t+1,f,g)$. Then either $i \in P(p,t,f,g)$, or i is not in $P(p,t,f,g)$ but $i \in C(c,t,f,g)$ for c read by p at $t+1$ with f. In the first case, either $f \in D(i,t,g) \subseteq D(i,t+1,g)$ or $p \in MV(i,t,g) \subseteq MV(i,t+1,g)$. In the second case, either $f \in D(i,t,g) \subseteq D(i,t+1,g)$ or $c \in MW(i,t,g)$. Since i is not in $P(p,t,f,g)$, then p reads c at $t+1$ for both f and $f_{(i)}$, one of which is in $F_{\leq 1}$, since f must be pertinent to i by the assumption. So by the definition of $MV(i,t+1,g)$, $p \in MV(i,t+1,g)$.

For the second part of the claim, assume $i \in C(c,t+1,f,g)$. Then either $i \in C(c,t,f,g)$, or i is not in $C(c,t,f,g)$ but either (1) $i \in P(p,t+1,f,g)$ for some processor p which writes to c at $t+1$ with f, or (2) i causes some p to write at $t+1$ with f. In the first case, either $f \in D(i,t,g) \subseteq D(i,t+1,g)$ or $c \in MW(i,t,g) \subseteq MW(i,t+1,g)$. In the second case (1), either $f \in D(i,t+1,g)$ or $p \in MV(i,t+1,g)$ and since f is pertinent to i, $c \in MW(i,t+1,g)$ by definition. In the second case (2), note that $i \in P(p,t+1,f,g)$, so either $f \in D(i,t+1,g)$ or $p \in MV(i,t+1,g)$. If $p \in MV(i,t+1,g)$, then since f is pertinent to i and not in $D(i,t+1,g)$ and hence $f_{(i)}$ is pertinent to i and not in $D(i,t+1,g)$ (by definition of $D(i,t+1,g)$), p writes to c on an input which is pertinent to i and not in $D(i,t+1,g)$. Thus $c \in MW(i,t+1,g)$ by definition. □

The next seven claims prove the desired bounds on the sizes of the sets defined above. These will be used to prove that REFINE preserves the t-goodness property in Lemma 5.1. For these claims, assume g is t-good, and REFINE(g,t) returns g'.

The following claim was essentially proven as part of Lemma 2 in [3].

Claim 5.4 *For all p and all $f \leq g'$, $|P(p,t+1,f,g')| \leq P_t + C_t$*

Proof: At $t+1$, a processor could be affected by all inputs which affect it at t plus those inputs which affect the contents of the cell it reads with f. Thus

$$P(p,t+1,f,g') \subseteq P(p,t,f,g') \cup C(c,t,f,g').$$

The claim follows from Claim 5.2, and the fact that g is t-good □

Claim 5.5 *For all inputs i, $|MV(i,t+1,g')| \leq MV_t + 2MW_tK_t^2MC_t$.*

Proof: For i not in Unset(g'), $MV(i,t+1,g') = \emptyset$, so the claim is trivial. For $i \in$ Unset(g'), by the definition of REFINE(g,t), $MW(i,t,g)$ contains no cell c that is read by more than $2K_t^2MC_t$ processors with $f \in F_{\leq 1}$. Thus for each cell in $MW(i,t,g') \subseteq MW(i,t,g)$, at most $2K_t^2MC_t$ processors are added to $MV(i,t+1,g')$. Also the processors in $MV(i,t,g') \subseteq MV(i,t,g)$ are added. The claim follows from Claim 5.2 and the fact that g is t-good □

Claim 5.6 *For all p, $|MP(p,t+1,g')| \leq MP_t + MC_tK_t$.*

Proof: An index $i \in MP(p,t+1,g')$ if $p \in MV(i,t+1,g')$. If $p \in MV(i,t+1,g')$ then either $p \in MV(i,t,g')$ and thus $i \in MP(p,t,g') \subseteq MP(p,t,g)$, or p is not in $MV(i,t,g')$ but p reads a cell $c \in MW(i,t,g')$ at $t+1$ with some $f \in F_{\leq 1}$ and thus $i \in MC(c,t,g') \subseteq MC(c,t,g)$. The bound follows by noting that p reads at most K_t different cells at $t+1$ with some $f \in F_{\leq 1}$. □

The following claim is similar to Lemma 2 in [3], but it is much more complex due to the fact that not all input maps are relevant.

Claim 5.7 *For all c and all $f \leq g'$, $|C(c,t+1,f,g')| \leq C_t + 2D_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$.*

Proof: For a given cell c there are two cases.

Case 1 Some processor p writes into c with f at step $t+1$.

Case 2 No processors write into c with f at step $t+1$.

In Case 1, an input i can only affect c at $t+1$ with f if i affects p at $t+1$ with f, so $C(c,t+1,f,g') = P(p,t+1,f,g')$, and $|C(c,t+1,f,g')| \leq P_{t+1}$.

In Case 2, an input i can only affect c at $t+1$ with f if i affects c at t with f or if i causes some p to write into c at $t+1$ with f. Then $|C(c,t+1,f,g')| \leq |C(c,t,f,g')| + |Y(c,t+1,f,g')|$, where $Y = Y(c,t+1,f,g')$ is the set of inputs $i \in$ Unset(g') which cause some p to write into c at $t+1$ with f. Let $Y = \{u_1,\ldots,u_r\}$, and suppose input u_i causes processor z_i to write into c with f, for $i \in \{1,2,\ldots,r\}$.

Subclaim 5.1 *For all $u_i, u_j \in Y$, with $z_i \neq z_j$, either u_i affects z_j at $t+1$ with $f_{(u_j)}$, or u_j affects z_i at $t+1$ with $f_{(u_i)}$, or $f_{(u_i)(u_j)}$ is not relevant. In the last case, let u_0 be the index that f maps to 1. Then u_0 affects z_i at $t+1$ with $f_{(u_i)}$, or u_0 affects z_j at $t+1$ with $f_{(u_j)}$, or u_i affects z_j at $t+1$ with $f_{(u_j)(u_0)}$, or u_j affects z_i at $t+1$ with $f_{(u_i)(u_0)}$.*

Proof: First assume $f_{(u_i)(u_j)}$ is relevant. Then if neither of the first conditions held, processors z_i and z_j would both write to c at $t+1$ with $f_{(u_i)(u_j)}$. Now assume $f_{(u_i)(u_j)}$ is not relevant. Then f must map at least one other index, say u_0, to 1. If none of the latter conditions held, processors z_i and z_j would both write to c at $t+1$ with $f_{(u_i)(u_j)(u_0)}$. □

Now consider a bipartite graph G with two sets of vertices labeled $\{u_1, \ldots, u_r\}$ and $\{z_1 \ldots z_r\}$, For $i \geq 1$, say there is an edge between u_i and z_j iff u_i affects z_j at $t+1$ with $f_{(u_j)}$ or with $f_{(u_j)(u_0)}$. Let E be the number of edges in G. The degree of each vertex z_j is at most $|P(z_j, t+1, f_{(u_j)}, g')| + |P(z_j, t+1, f_{(u_j)(u_0)}, g')|$, and thus $E \leq 2rP_{t+1}$.

By Claim 5.3 and the fact that at most P_{t+1} of the z_i's can be equal, u_0 affects processor z_i at $t+1$ with $f_{(u_i)}$ for at most $|D(u_0, t+1, g')| + P_{t+1}|MV(u_0, t+1, g')| \leq D_{t+1} + P_{t+1}MV_{t+1}$ values of i. Then there are at least $r - D_{t+1} - P_{t+1}MV_{t+1}$ choices for u_i such that u_0 does not affect z_i at $t+1$ with $f_{(u_i)}$. Given u_i, there are at least $r - D_{t+1} - P_{t+1}MV_{t+1} - P_{t+1}$ choices for u_j such that u_0 does not affect z_j at $t+1$ with $f_{(u_j)}$ and u_j does not cause z_i to write into c at $t+1$ with f (i.e., $z_j \neq z_i$). For a u_i and u_j such as this, one of the edges (u_i, z_j) or (u_j, z_i) must exist. Hence $\frac{1}{2}(r - D_{t+1} - P_{t+1}MV_{t+1})(r - D_{t+1} - P_{t+1}MV_{t+1} - P_{t+1}) \leq E \leq 2rP_{t+1}$. This implies $r \leq 2D_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$. Then $|C(c, t+1, f, g')| \leq |C(c, t, f, g')| + |Y(c, t+1, f, g')| \leq C_t + 2D_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$. □

Claim 5.8 *For all i, $|MW(i, t+1, g')| \leq MW_t + 2MV_{t+1}$.*

Proof: Consider a processor p in $MV(i, t+1, g')$. This processor p writes to a cell c with f_0 and a cell c' with f_i. Any possible writes to other cells with inputs pertinent to i only occur with $f \in D(i, t+1, g')$. □

Claim 5.9 *For all i, $|D(i, t+1, g')| \leq 4MV_{t+1}P_{t+1}$.*

Proof: Straightforward, from the definition. □

Claim 5.10 *For all c, $|MC(c, t+1, g')| \leq MC_t + 3P_{t+1} + MP_{t+1}$*

Proof: An input $i \in MC(c, t+1, g')$ if $c \in MW(i, t+1, g')$. If $c \in MW(i, t+1, g')$, then either $c \in MW(i, t, g')$, and thus $i \in MC(c, t, g')$, or c is not in $MW(i, t, g')$ but is written to by a processor $p \in MV(i, t+1, g')$ with some f pertinent to i but not in $D(i, t+1, g')$. In the second case, it is enough to consider only the case of p writing to c with f_0 or f_i, since for any other f pertinent to i, either p writes to the same cell as with f_0 or f_i, or $f \in D(i, t+1, g')$. Note that only one processor can write to c at $t+1$ with f_0, and this adds at most MP_{t+1} inputs to $MC(c, t+1, g')$.

Let $W = W(c, t+1, g')$ be the set of inputs i such that $p \in MV(i, t+1, g')$ and p writes to c at $t+1$ with f_i but p does not write to c at $t+1$ with f_0. (Note that for each i, there is at most one p, else there would be a write conflict.) Then $W(c, t+1, g') = Y(c, t+1, f_0, g')$ from above, so $|W(c, t+1, g')| \leq 2D_t + 2P_{t+1}MV_{t+1} + 6P_{t+1}$. (Actually, since $f_{0(j,j')}$ is always relevant, we can use the simpler analysis in [3] to show that $|Y(c, t+1, f_0, g')| \leq 3P_{t+1}$. We will use this better bound, though asymptotically, it does not affect our lower bound.) This adds at most $3P_{t+1}$ inputs to $MC(c, t+1, g')$ □

Let δ be a constant, $0 \leq \delta < 1$. Let $\epsilon = 2^{-\log^\delta n}$. Let $K = 5/\epsilon$. Let $T = (1-\delta)\log_\alpha(2^{-1/(1-\delta)}\log n)$, for α given in Claim 5.1. (Note that $T = \Theta(\log\log n)$.) Let $Z = 1 - K^{-1}$. For the following we will assume n is large enough so that the analysis holds.

Lemma 5.1 *With probability Z, for every t $(0 \leq t \leq T)$, f_t is t-good.*

Proof: The bounds on the sizes of sets follows from Claim 5.4, Claim 5.5, Claim 5.6, Claim 5.7, Claim 5.9, Claim 5.8, and Claim 5.10. Using the fact that $P_{t+1} \geq 2P_t$, we can see that the number of inputs set by RandomSet at step t is at most $n/2K_t \leq n2^{1-t}/2K$. Thus, the total number of inputs set in all steps is at most n/K. Then over the chosen input distribution, the probability that any of these inputs is set to 1 is at most

$$\frac{1}{2K} + \frac{\epsilon}{K} \leq \frac{1}{K}.$$

Thus the probability of success in every round is at least $1 - K^{-1} = Z$. □

Theorem 5.1 *For any constant δ, $0 \leq \delta < 1$, solving 2-OR with probability greater than $\frac{1}{2}(1+\epsilon)$ on an n processor Randomized CREW PRAM requires T steps*

Proof: By Theorem 3.1, we simply need to show that any deterministic algorithm solving 2-OR with the desired probability over the given distribution requires T steps. From Lemma 5.1, the probability that any of the inputs will be set to 1 is at most $K^{-1} = \epsilon/5$. Given that no input is set to 1,

$$C(c, T, f_0, g_T) \leq K^{\alpha^T} \leq (5 \cdot 2^{\log^\delta n})^{\frac{1}{2}\log^{1-\delta} n} \leq O(\sqrt{n}).$$

Thus the probability that some input that affects the output cell is 1 is at most

$$O(n^{-1/2}) + \frac{\epsilon}{4}.$$

Assuming that no inputs were set to 1 by RANDOMSET, and no inputs affecting the output cell are 1, the output cell is constant over all the remaining possible input maps. Then whatever value is output by the algorithm, it is correct with probability at most

$$\frac{1}{2} + \epsilon/5 + O(n^{-1/2}) + \epsilon/4 < \frac{1}{2}(1+\epsilon).$$

□

Corollary 5.1 *For any constant δ, $0 \leq \delta < 1$, solving 2-compaction with probability greater than $\frac{1}{2}(1+\epsilon)$ on an n processor Randomized CREW PRAM requires $\Omega(\log\log n)$ steps*

Proof: There is a constant time reduction from 2-OR to 2-Compaction [5]. □

Corollary 5.2 *For any constant δ, $0 \leq \delta < 1$, and any constant γ, $0 < \gamma < 1$, solving approximate 2-compaction with a destination array of size $n^{1-\gamma}$ with probability greater than $\frac{1}{2}(1+\epsilon)$ on an n processor Randomized CREW PRAM requires $\Omega(\log\log n)$ steps*

Proof: Let $C = \lceil -\log^{-1}(1-\gamma) \rceil$. After performing approximate 2-compaction C times, the marked items would reside in an array of size at most $\sqrt{n}$. Then 2-compaction could be performed in constant time using n processors. Thus approximate 2-compaction can be solved at most C times faster than 2-compaction. □

6 Compaction on the EREW PRAM

The proof of the lower bound for compaction on the EREW PRAM is similar to that on the CREW PRAM. One major difference is that the lower bound for the EREW PRAM applies regardless of the number of processors used.

We will use many of the same set names, since they intuitively denote the same kinds of sets. However, we must change their definitions slightly to obtain the better EREW PRAM lower bound.

For the EREW PRAM, REFINE(t, g) will simply return g. Therefore, we will leave off the parameter g from the set definitions.

- Let $P(p, t, f)$ be the set of inputs which affect processor p at t with f.
- Let $C(c, t, f)$ be the set of inputs which affect cell c at t with f.
- Let $MV(i, 0)$ be the empty set, and let $MV(i, t+1)$ be defined inductively as the set of processors in $MV(i, t)$ plus the set of processors that read a cell in $MW(i, t)$ at $t+1$ with some f pertinent to i not in $DV(i, t+1)$.
- Let $MW(i, 0, g) = \{i\}$ if $1 \leq i \leq n$, else let $MW(i, 0, g)$ be the empty set. Let $MW(i, t+1)$ be defined inductively as the set of cells in $MW(i, t)$ plus the set of cells that are written to by a processor $p \in MV(i, t+1)$ with some f pertinent to i not in $DW(i, t+1)$.
- Let $DV(i, t)$ be a set of "discarded" input maps for an input i. Let $DV(i, 0)$ be the empty set, and let $DV(i, t+1)$ be all inputs in $DW(i, t)$, plus for all $c \in MW(i, t)$, if some p not in $MV(i, t)$ reads c with f_0, the inputs f_j and $f_{i,j}$, for all $j \in P(p, t, f_0)$. else for one p not in $MV(i, t)$ which reads c with an f_h pertinent to i not in $DW(i, t)$, the inputs f_j and $f_{i,j}$ for all

$$\begin{aligned} j \in\ & P(p, t, f_h) \cup \{j' : f_{j'} \in DW(h, t) \text{ and } j' \neq i\} \cup \\ & \{j' : j' \in P(p', t, f_0) \text{ for some } p' \in MV(h, t), \text{ and } j' \neq i\}. \end{aligned}$$

- Let $DW(i, t)$ be another set of "discarded" input maps for an input i. Let

$$S_{i,p,t} = (P(p, t+1, f_0) \cup P(p, t+1, f_i)) \setminus \{i\}.$$

  Let $DW(i, 0)$ be the empty set, and let $DW(i, t+1)$ be all inputs in $DV(i, t+1)$, plus for all $p \in MV(i, t+1)$, the inputs f_j and $f_{i,j}$ for all $j \in S_{i,p,t}$.

The sets $P(p, t, f)$, $C(c, t, f)$, $MV(i, t)$, and $MW(i, t)$ are intuitively the same sets as in the previous section. However, in this proof we have two sets of discarded input maps. Intuitively, $DV(i, t)$ denotes the set of input maps discarded before an exclusive read step, and $DW(i, t)$ denotes the set of input maps discarded before an exclusive write step. ($DW(i, t)$ is basically the same as $D(i, t, g)$ of the previous section.) The complex definition of $DV(i, t)$ is needed in order to obtain a very tight bound on $|MV(i, t)|$.

Define the following constants (with K to be chosen later):

- $P_0 = K$, $P_{t+1} = P_t + C_t$
- $C_0 = K + 1$, $C_{t+1} = C_t + 2DV_{t+1} + 4P_{t+1}MV_{t+1} + 6P_{t+1}$.
- $DV_0 = 0$, $DV_{t+1} = 2DW_t + 2(MV_t + 1)MW_tP_t$
- $DW_0 = 0$, $DW_{t+1} = DV_{t+1} + 4MV_{t+1}P_{t+1}$
- $MV_0 = 0$, $MV_{t+1} = MV_t + MW_t$

- $MW_0 = 1$, $MW_{t+1} = MW_t + 2MV_{t+1}$

The following claim gives a bound on the rate of growth of these constants.

Claim 6.1 *For $t \geq 1$ and some constant $\beta \geq 0$, $C_t \leq \beta^{t^2}$.*

Proof: First note that for all $t \geq 0$, $P_t \leq C_t$, $MV_t \leq MW_t$, $DV_t \leq DW_t \leq C_t$, and $P_{t+1} \leq 2C_t$. From these we can see that $MW_{t+1} \leq 5MW_t$, and thus $MW_{t+1} \leq 5^{t+1}$. Also, $DW_{t+1} \leq O(25^{t+1}C_t)$, and $C_{t+1} \leq O(25^{t+1}C_t)$. The last inequality implies that $C_t \leq \beta^{t^2}$, for some $\beta \geq 0$. (Note: no attempt has been made to optimize the value of β.) □

Say g is t-good if

1. for all processors p, cells c, inputs i, and input maps $f \leq g$, $|P(p,t,f)| \leq P_t$, $|C(c,t,f)| \leq C_t$, $|MV(i,t)| \leq MV_t$, $|MW(i,t)| \leq MW_t$, $|DV(i,t)| \leq DV_t$, $|DW(i,t)| \leq DW_t$,
2. $\text{Unset}(g) = n$,

As in the previous section, we now prove the following claim showing the important relationship between the sets defined above.

Claim 6.2 *For all p, c, f, t, i, if $i \in P(p,t,f)$, then either $f \in DV(i,t)$ or $p \in MV(i,t)$; also, if $i \in C(c,t,f)$, then either $f \in DW(i,t)$ or $c \in MW(i,t)$.*

Proof: By induction. The case for $t = 0$ is straightforward.

For the first part of the claim, assume $i \in P(p,t+1,f)$. Then either $i \in P(p,t,f)$, or i is not in $P(p,t,f)$ but $i \in C(c,t,f)$ for c read by p at $t+1$ with f. In the first case, either $f \in DV(i,t) \subseteq DV(i,t+1)$ or $p \in MV(i,t) \subseteq MV(i,t+1)$. In the second case, either $f \in DW(i,t) \subseteq DV(i,t+1)$ or $c \in MW(i,t)$. Assuming f is not in $DV(i,t+1)$, by the definition of $MV(i,t+1)$, $p \in MV(i,t+1)$.

For the second part of the claim, assume $i \in C(c,t+1,f)$. Then either $i \in C(c,t,f)$, or i is not in $C(c,t,f)$ but either (1) $i \in P(p,t+1,f)$ for some processor p which writes to c at $t+1$ with f, or (2) i causes some p to write to c at $t+1$ with f. In the first case, either $f \in DW(i,t) \subseteq DW(i,t+1)$ or $c \in MW(i,t) \subseteq MW(i,t+1)$. In the second case (1), either $f \in DV(i,t+1) \subseteq DW(i,t+1)$ or $p \in MV(i,t+1)$ and since f is pertinent to i, $c \in MW(i,t+1)$ by definition. In the second case (2), note that $i \in P(p,t+1,f)$, so either $f \in DV(i,t+1) \subseteq DW(i,t+1)$ or $p \in MV(i,t+1)$. If $p \in MV(i,t+1)$, then since f is pertinent to i and not in $DW(i,t+1)$ and hence $f_{(i)}$ is pertinent to i and not in $DW(i,t+1)$ (by definition of $DW(i,t+1)$), p writes to c on an input which is pertinent to i and not in $DW(i,t+1)$. Thus $c \in MW(i,t+1)$ by definition. □

For the remaining claims, we assume $g(= g_*)$ is t-good.

Claim 6.3 *For all p and all relevant f, $|P(p,t+1,f)| \leq P_t + C_t$*

Proof: At $t+1$, a processor could be affected by all inputs which affect it at t plus those inputs which affect the contents of the cell it reads with f. Thus

$$P(p,t+1,f) \subseteq P(p,t,f) \cup C(c,t,f).$$

The claim follows from the fact that g is t-good □

Claim 6.4 *For all i, $|MV(i,t+1)| \leq MV_t + MW_t$.*

Proof: If $p \in MV(i,t+1)$, then either $p \in MV(i,t)$ or p is not in $MV(i,t)$ but p reads a cell c in $MW(i,t)$ at $t+1$ with some f pertinent to i not in $DV(i,t+1)$. In the second case, we argue that p is the only processor not in $MV(i,t)$ that reads c with some f pertinent to i not in $DV(i,t+1)$. (In fact, p reads c with *all* f pertinent to i not in $DV(i,t+1)$.)

First, if p reads c with f_0, then to avoid a read conflict, p is the only processor to read c with f_0. Then by the construction of $DV(i,t+1)$, p reads c for all $f \in F_{\leq 1}$ not in $DV(i,t+1)$. Note that since $p \notin MV(i,t)$, Claim 6.2 implies that p reads c for all $f_{(i)}$ for all $f \in F_{\leq 1}$ not in $DV(i,t+1)$, i.e., for all f pertinent to i not in $DV(i,t+1)$. Thus to avoid a read conflict, p must be the sole processor that reads c for some f pertinent to i not in $DV(i,t+1)$.

Now assume p does not read c with f_0. We obtain a contradiction as follows. First, if another processor $p' \notin MV(i,t)$ reads c with f_0, then from the above argument, there would be a read conflict. Thus we can assume no processor not in $MV(i,t)$ reads c with f_0. Now notice that p reads the same cell for f_0 and f_i (since by Claim 6.2, $p \notin MV(i,t)$ and $f_0 \notin DV(i,t+1)$ implies $i \notin P(p,t,f_0)$). So without loss of generality, assume p reads c with either $f = f_j$ or $f = f_{i,j}$ $(j \neq i)$, and that $f \notin DV(i,t+1)$. Note that $f \notin DV(i,t+1)$ implies $f_{(i)} \notin DV(i,t+1)$. Then note that since $p \notin MV(i,t+1)$, by Claim 6.2, $i \notin P(p,t,f)$, so p reads cell c for both f and $f_{(i)}$ (i.e. for both f_j and $f_{i,j}$). Consequently, there is at least one processor p' that reads c with an f_h pertinent to i not in $DW(i,t)$. Let p' and f_h be the actual processor and input map chosen in the construction of $DV(i,t+1)$.

First consider the case $h = j$. Note that $j \in P(p,t,f_0)$, and thus by Claim 6.2 $p \in MV(j,t) = MV(h,t)$. Consequently, f_j would be placed in $DV(i,t+1)$, a contradiction. Second consider the case $p = p'$ and $h \neq j$. Then $P(p,t,f_0)$ would contain both j and h, and thus by Claim 6.2 $p \in MV(h,t)$. Consequently, f_j would be placed in $DV(i,t+1)$, a contradiction. Lastly, consider the case $p \neq p'$ and $h \neq j$. Again note that $j \in P(p,t,f_0)$. If $p \in MV(h,t)$ then f_j would be placed in $DV(i,t+1)$, so $p \notin MV(h,t)$. If $f_j \in DW(h,t)$, then f_j would be placed in $DV(i,t+1)$, so $f_j \notin DW(h,t)$. Then by Claim 6.2, $h \notin P(p,t,f_j)$, so p reads c on $f_{h,j}$. Now if $j \in P(p',t,f_h)$, then f_j would be placed in $DV(i,t+1)$, so $j \notin P(p',t,f_h)$. Therefore, p' also reads c on $f_{h,j}$. This would cause a read conflict and thus implies a contradiction. □

Claim 6.5 *For all i, $|DV(i,t+1)| \leq 2DW_t + 2(MV_t+1)MW_tP_t$.*

Proof: Straightforward, from the definition. □

Claim 6.6 *For all c and all relevant f, $|C(c,t+1,f)| \leq C_t + 2DV_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$.*

Proof: For a given cell c there are two cases.

Case 1 Some processor p writes into c with f at step $t+1$.

Case 2 No processors write into c with f at step $t+1$.

In Case 1, an input i can only affect c at $t+1$ with f if i affects p at $t+1$ with f, so $C(c,t+1,f) = P(p,t+1,f)$, and $|C(c,t+1,f)| \leq P_{t+1}$.

In Case 2, an input i can only affect c at $t+1$ with f if i affects c at t with f or if i causes some p to write into c at $t+1$ with f. Then $|C(c,t+1,f)| \leq |C(c,t,f)| + |Y(c,t+1,f)|$, where $Y = Y(c,t+1,f)$ is the set of inputs i which cause some p to write into c at $t+1$ with f. Let $Y = \{u_1, \ldots, u_r\}$, and suppose input u_i causes processor z_i to write into c with f, for $i \in \{1,2,\ldots,r\}$.

Subclaim 6.1 *For all $u_i, u_j \in Y$, with $z_i \neq z_j$, either u_i affects z_j at $t+1$ with $f_{(u_j)}$, or u_j affects z_i at $t+1$ with $f_{(u_i)}$, or $f_{(u_i)(u_j)}$ is not relevant. In the last case, let u_0 be the index that f maps to 1. Then u_0 affects z_i at $t+1$ with $f_{(u_i)}$, or u_0 affects z_j at $t+1$ with $f_{(u_j)}$, or u_i affects z_j at $t+1$ with $f_{(u_j)(u_0)}$, or u_j affects z_i at $t+1$ with $f_{(u_i)(u_0)}$.*

Proof: As in Subclaim 5.1. □

Now consider a bipartite graph G with two sets of vertices labeled $\{u_1, \ldots, u_r\}$ and $\{z_1 \ldots z_r\}$, For $i \geq 1$, say there is an edge between u_i and z_j iff u_i affects z_j at $t+1$ with $f_{(u_j)}$ or with $f_{(u_j)(u_0)}$. Let E be the number of edges in G. The degree of each vertex z_j is at most $|P(z_j, t+1, f_{(u_j)}, g')| + |P(z_j, t+1, f_{(u_j)(u_0)}, g')|$, and thus $E \leq 2rP_{t+1}$.

By Claim 6.2 and the fact that at most P_{t+1} of the z_i's can be equal, u_0 affects processor z_i at $t+1$ with $f_{(u_i)}$ for at most $|DV(u_0, t+1)| + P_{t+1}|MV(u_0, t+1)| \leq DV_{t+1} + P_{t+1}MV_{t+1}$ values of i. Then there are at least $r - DV_{t+1} - P_{t+1}MV_{t+1}$ choices for u_i such that u_0 does not affect z_i at $t+1$ with $f_{(u_i)}$. Given u_i, there are at least $r - DV_{t+1} - P_{t+1}MV_{t+1} - P_{t+1}$ choices for u_j such that u_0 does not affect z_j at $t+1$ with $f_{(u_j)}$ and u_j does not cause z_i to write into c at $t+1$ with f (i.e., $z_j \neq z_i$). For a u_i and u_j such as this, one of the edges (u_i, z_j) or (u_j, z_i) must exist. Hence $\frac{1}{2}(r - DV_{t+1} - P_{t+1}MV_{t+1})(r - DV_{t+1} - P_{t+1}MV_{t+1} - P_{t+1}) \leq E \leq 2rP_{t+1}$. This implies $r \leq 2DV_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$. Then $|C(c, t+1, f, g')| \leq |C(c, t, f, g')| + |Y(c, t+1, f, g')| \leq C_t + 2DV_{t+1} + 2P_{t+1}MV_{t+1} + 5P_{t+1}$. □

Claim 6.7 *For all i, $|MW(i, t+1)| \leq MW_t + 2MV_{t+1}$.*

Proof: Consider a processor p in $MV(i, t+1)$. This processor p writes to a cell c with f_0 and a cell c' with f_i. Any possible writes to other cells with inputs pertinent to i only occur with $f \in DW(i, t+1)$. □

Claim 6.8 *For all i, $|DW(i, t+1)| \leq DV_t + 4MV_{t+1}P_{t+1}$.*

Proof: Straightforward, from the definition. □

Let $Z = 1$, $K = 1$, and assume $T > 0$. For the following we will assume n is large enough so that the analysis holds.

Lemma 6.1 *With probability Z, for every t ($0 \leq t \leq T$), f_t is t-good.*

Proof: The bounds on the sizes of sets follows from Claim 6.3, Claim 6.4, Claim 6.6, Claim 6.5, Claim 6.7, and Claim 6.8. Also, REFINE never sets any inputs. □

Let δ be a constant, $0 \leq \delta < 1$. Let $\epsilon = n^{-\delta}$. Let $T = \sqrt{(1-\delta)\log_\beta n - \log_\beta 4}$, for β given in Claim 6.1. (Note that $T = \Theta(\sqrt{\log n})$.)

Theorem 6.1 *For any constant δ, $0 \leq \delta < 1$, solving 2-OR with probability greater than $\frac{1}{2}(1+\epsilon)$ on a Randomized EREW PRAM requires $\Omega(\sqrt{\log n})$ steps*

Proof: By Theorem 3.1, we simply need to show that any deterministic algorithm solving 2-OR with the desired probability over the given distribution requires $\Omega(\sqrt{\log n})$ steps. From Lemma 6.1 and Claim 6.1,

$$C(c, T, f_0, g_T) \leq \beta^{T^2} \leq \frac{n^{1-\delta}}{4}.$$

The probability that some input that affects the output cell is 1 is at most

$$\frac{n^{-\delta}}{4} + \frac{\epsilon}{4}.$$

Assuming no inputs affecting the output cell are 1, the output cell is constant over all the remaining possible input maps. Then whatever value is output by the algorithm, it is correct with probability at most

$$\frac{1}{2} + \frac{n^{-\delta}}{4} + \frac{\epsilon}{4} \leq \frac{1}{2}(1+\epsilon).$$

□

Corollary 6.1 *For any constant δ, $0 \leq \delta < 1$, solving 2-compaction with probability greater than $\frac{1}{2}(1+\epsilon)$ on a Randomized EREW PRAM requires $\Omega(\sqrt{\log n})$ steps*

Proof: There is a constant time reduction from 2-OR to 2-Compaction [5]. □

Let $\epsilon = n^{-\delta-\gamma}$. Let $T = \sqrt{\delta \log_\beta n - \log_\beta 4}$, for β given in Claim 6.1. (Note that $T = \Theta(\sqrt{\log n})$.)

Corollary 6.2 *For any constant γ, $0 < \gamma < 1$, and any constant δ, $0 \leq \delta < \gamma$, solving approximate 2-compaction with a destination array of size $n^{1-\gamma}$ with probability greater than $\frac{1}{2}(1+\epsilon)$ on a Randomized EREW PRAM requires $\Omega(\sqrt{\log n})$ steps.*

Proof: Similar to reducing 2-OR to 2-Compaction, we can reduce the problem of approximate 2-OR (that is, if all inputs are 0, then the array of size $n^{1-\gamma}$ contains all zeros, else it contains at least one 1) to approximate 2-Compaction. By Theorem 3.1, we simply need to show that any deterministic algorithm solving approximate 2-OR with the desired probability over the given distribution requires $\Omega(\sqrt{\log n})$ steps. From Lemma 6.1 and Claim 6.1,

$$C(c, T, f_0, g_T) \leq \beta^{T^2} \leq \frac{n^\delta}{4}.$$

The probability that some input is 1 that affects any of the $n^{1-\gamma}$ output cells is at most

$$\frac{n^{\delta-\gamma}}{4} + \frac{\epsilon}{4}.$$

Assuming no inputs affecting these output cells are 1, the output cells are constant over all the remaining possible input maps. Then whatever values are output by the algorithm, they could be correct with probability at most

$$\frac{1}{2} + \frac{n^{\delta-\gamma}}{4} + \frac{\epsilon}{4} \leq \frac{1}{2}(1+\epsilon).$$

□

7 Boolean Functions

In this section we will prove randomized lower bounds for computing boolean functions on the CREW PRAM model. Our main result is a lower bound for general boolean functions, and it improves a result of Dietzfelbinger, Kutylowski, and Reischuk [4]. We also present a randomized lower bound for computing PARITY which matches the optimal randomized lower bound of Dietzfelbinger, Kutylowski, and Reischuk [4]. Although we will provide all the relevant definitions here, the reader is encouraged to examine [3, 4, 20] for further details.

7.1 Definitions

Let Q be the set $\{0, 1\}$. The input distribution for each problem will give some non-zero probability to every possible input map. The function REFINE(t, g) always returns g (i.e., the Random Adversary does not set any inputs)

- Let $P(p, t, f)$ be the set of inputs which affect processor p at t with f.
- Let $C(c, t, f)$ be the set of inputs which affect cell c at t with f.

Now define the following recursive functions.

- Let $P_0 = 0$, and $P_{t+1} = P_t + C_t$ for $t \geq 0$.
- Let $C_0 = 1$, and $C_{t+1} = 3(P_t + C_t)$ for $t \geq 0$.

Parberry and Yan [20] show that $P_t \leq 4^t$, and $C_t \leq 3(4^t)$.

Let $\text{fib}(k)$ be the kth Fibonacci number, that is, $\text{fib}(0) = 0$, $\text{fib}(1) = 1$, and $\text{fib}(k) = \text{fib}(k-2) + \text{fib}(k-1)$ for $k \geq 2$. Let $\phi(x) = \min\{t | \text{fib}(2t+1) \geq x\}$.

Let $x_1, \ldots, x_n$ denote boolean variables, $a_1, \ldots, a_n$ denote elements of $\{0,1\}$ and a denote an element of $\{0,1\}^n$. Let B_n denote the set $\{f | f : \{0,1\}^n \rightarrow \{0,1\}\}$. To avoid ambiguity, we use the term *input vector* to denote an element from the domain of f (input vectors have a 1-to-1 correspondence with input maps), and *input* to denote the boolean variable corresponding to one of the n locations in the input vector. For $f \in B_n$ define $bs(f)$ as the maximum of the numbers $\max\{l | \exists S_1, \ldots, S_l$ disjoint such that $f(a^{S_j}) \neq f(a), 1 \leq j \leq l\}$ taken over all input vectors a, where a^S is obtained from a by flipping all bits in positions $i \in S$. Say a is a *base input vector* if there are $bs(f)$ disjoint sets $S_1, \ldots, S_{bs(f)}$ such that $f(a^{S_j}) \neq f(a)$, $1 \leq j \leq bs(f)$.

For $S \subseteq \{1, \ldots, n\}$ let m_S be the positive monomial $\prod_{i \in S} x_i$ and $m_S(a) = \prod_{i \in S} a_i$

Fact 7.1 ([22]) *Every $f \in B_n$ can be written $f = \sum_S \alpha_S(f) m_S$ for unique integer coefficients $\alpha_S(f)$.*

Dietzfelbinger, Kutylowski, and Reischuk obtain the following explicit formula for the coefficient $\alpha_S(f)$:

$$\alpha_S(f) = (-1)^{|S|} \sum_{\substack{a \in f^{-1}(1) \\ \{i | a_i = 1\} \subseteq S}} (-1)^{\text{PARITY}_n(a)}.$$

For $f \in B_n$, let $\deg(f) = \max\{|S| \ | \alpha_S(f) \neq 0\}$.

Fact 7.2 ([23]) *$deg(f) \geq \sqrt{bs(f)}$.*

From [4], Facts 2, 4, and 5, we obtain

Fact 7.3 *$deg(f) \leq (bs(f))^4$.*

Let P_i denote processor i and C_j denote cell j. Let S be the possible states of any processor, and let C be the possible contents of a cell. Now we define the partitions of $\{0,1\}^n$ induced by the states of the processors and cells at step t. For $i, j \in N$, $s \in S$, and $c \in C$, let

$$\begin{aligned} G(s,i,t) &= \{a \in \{0,1\}^n | P_i \text{ is in state } s \text{ after step } t \text{ on input vector } a\}, \\ H(c,j,t) &= \{a \in \{0,1\}^n | C_j \text{ contains } c \text{ after step } t \text{ on input vector } a\}. \end{aligned}$$

Let

$$\begin{aligned} \mathcal{G}(t) &= \{G(s,i,t) | i \in N, s \in S\}, \text{and} \\ \mathcal{H}(t) &= \{H(c,j,t) | j \in N, c \in C\}. \end{aligned}$$

For $W \in \{0,1\}^n$, let c_W be the characteristic function of W. For $\mathcal{W}$ a class of subsets of $\{0,1\}^n$, let $\deg(\mathcal{W}) = \max\{\deg(c_W) | W \in \mathcal{W}\}$.

Lemma 7.1 ([4])
(a) $deg(\mathcal{G}(0)) = 0$ and $deg(\mathcal{H}(0)) = 1$,
(b) $deg(\mathcal{G}(t)) \leq deg(\mathcal{H}(t-1)) + deg(\mathcal{G}(t-1))$,
(c) $deg(\mathcal{H}(t)) \leq deg(\mathcal{H}(t-1)) + deg(\mathcal{G}(t))$, and
(d) $deg(\mathcal{G}(t)) \leq fib(2t)$ and $deg(\mathcal{H}(0)) \leq fib(2t+1)$.

Fact 7.4 ([4]) *Let $f \in B_n$, and let A be a deterministic CREW PRAM algorithm that computes f. Then A requires at least $\phi(deg(f))$ steps.*

8 Improved Lower Bounds for General Boolean Functions

Here we would like to prove a lower bound on the time for a randomized CREW PRAM algorithm to compute any boolean function f on n inputs with probability greater than $\frac{1}{2}(1+\epsilon)$, for some ϵ. Consider a base input vector a for f, and the $bs(f)$ sets $S_1, \ldots, S_{bs(f)}$, such that $f(a) \neq f(a^{S_j})$ $(1 \leq j \leq bs(f))$.

The input distribution we use is the following:

- With probability $\frac{1}{2}$, use input vector a.
- For each j, $(1 \leq j \leq bs(f))$, with probability $\frac{1}{2bs(f)}(1-\epsilon/2)$ use input vector a^{S_j}.
- Each of the $k = 2^n - (bs(f)+1)$ other possible input vectors occurs with probability $\frac{\epsilon}{4k}$.

Say g is t-good if

1. for all processors p, cells c, inputs i, and input maps $f \leq g$, $|P(p,t,f)| \leq P_t$, $|C(c,t,f)| \leq C_t$, and
2. $\text{Unset}(g) = n$.

Let f be a boolean function. Let δ be a constant, $0 \leq \delta < 1$. Let ϵ be a constant, $0 < \epsilon < 1$, and let $T = \log_4 bs(f) + \log_4(\epsilon/12)$.

Lemma 8.1 *With probability* 1, *for every* t $(0 \leq t \leq T)$, f_t *is* t*-good.*

Proof: The algorithm A is simply a deterministic CREW PRAM algorithm which must not perform a concurrent write for any input vector, since each input vector occurs with non-zero probability. For any algorithm A of this type, Parberry and Yan [20] shows that the properties required hold for all t. □

Theorem 8.1 *Let* A *be a randomized CREW PRAM algorithm that allegedly computes* f *with probability greater than* $\frac{1}{2}(1+\epsilon)$. *Then* A *runs in at least* T *steps.*

Proof: By Theorem 3.1, it suffices to show that for any deterministic algorithm A' running in less than T steps over the input distribution, A' computes f correctly with probability less than $\frac{1}{2}(1+\epsilon)$. From Lemma 8.1, $C_T \leq 3(4^T) \leq \epsilon(bs(f))/4$. Then the probability that some input that affects the output cell indicates that $f(a)$ is the wrong answer is $\epsilon/2$. Thus, as argued in previous Theorems, the algorithm will then be correct with probability at most

$$\frac{1}{2} + \frac{\epsilon}{2} \leq \frac{1}{2}(1+\epsilon).$$

□

Notice that for constant ϵ, we achieve the bound $.5 \log bs(f) - O(1)$, whereas the previous best bound was $\phi(\sqrt{\epsilon \cdot bs(f)}) \approx .36 \log bs(f) - O(1)$ [4].

Also note that using the Random Adversary combined with the deterministic lower bound of [4] would also yield a $\phi(\sqrt{\Theta(\epsilon \cdot bs(f))})$ lower bound for the randomized computation of f on a CREW PRAM, and thus the multiplicative constant would remain approximately .36.

9 Matching lower bound for PARITY

Here we prove a lower bound for the time of a randomized CREW PRAM to compute the PARITY of n bits with probability greater than $\frac{1}{2}$.

We say g is t-good if

1. Lemma 7.1 holds for all $t' \leq t$, and
2. Unset$(g) = n$.

For our input distribution we assume each input is assigned a value in Q with equal probability. Let $T = \lceil \phi(n) \rceil$.

Lemma 9.1 *With probability* 1, *for every* t $(0 \leq t \leq T)$, f_t *is* t*-good.*

Proof: The algorithm A is simply a deterministic CREW PRAM algorithm which must not perform a concurrent write for any input vector, since each input vector occurs with non-zero probability. For any algorithm A of this type, [4] shows that Lemma 7.1 holds for all t. □

Theorem 9.1 *Let* A *be a randomized CREW PRAM algorithm that allegedly finds the parity of* n *inputs with probability greater than* $\frac{1}{2}$. *Then* A *runs in at least* T *steps.*

Proof: By Theorem 3.1, it suffices to show that for any deterministic algorithm A' running in less than T steps over the input distribution, A' computes the correct parity with probability at most $\frac{1}{2}$. From Lemma 9.1, after $t \leq T$ steps, if h_1 (h_0) be the characteristic function $H(1,1,T)$ $(H(0,1,T))$, i.e., the input vectors for which A' outputs 1 (0), then $deg(h_1) < n$ $(deg(h_0) < n)$. Then from the formula for the coefficients $\alpha_S(f)$, we see that

$$0 = |\alpha_{1,\ldots,n}(h_1)| = \left| \sum_{a \in h_1^{-1}(1)} (-1)^{\text{PARITY}_n(a)} \right|.$$

and

$$0 = |\alpha_{1,\ldots,n}(h_0)| = \left| \sum_{a \in h_0^{-1}(1)} (-1)^{\text{PARITY}_n(a)} \right|.$$

Thus $h_1^{-1}(1)$ and $h_0^{-1}(1)$ both contain the same number of input vectors with even and odd parity, and thus both must err on exactly half of the input vectors that set them to 1. Thus A' errs on exactly half of the input vectors. Since all input vectors are equally likely, A' errs with probability $\frac{1}{2}$. □

10 Merging on the EREW PRAM

In this section we prove randomized lower bounds for Integer Merging on the EREW PRAM model. For an integer $m \geq 1$, We show that merging two sorted lists of size n containing elements from $0, 2, \ldots, m-1$ requires $\Omega(\log \min\{n, m\})$ time. This matches the deterministic lower bound of Hagerup and Kutylowski [13], and it is optimal for $m = \Omega(\log n)$.

Let $Q = \{0, 1, \ldots, m-1\}$. Let $r = \min\{n, m\}$. As in [13] let $X = (1, 2, \ldots, r-1, r-1, \ldots, r-1)$, $Y_1 = (0, r-1, r-1, \ldots, r-1)$, and $Y_2 = (r-1, r-1, \ldots, r-1)$ be sorted lists of length n. The input distribution we will use will place equal probability on inputs of (X, Y_1), and (X, Y_2). The function REFINE(t, g) simply returns g. Say g is t-good if

1. the first input in the second list affects the values of at most 4^t cells, and
2. Unset$(g) = n$.

Let $T = \log_4(r-2)$, and let $Z = 1$.

Lemma 10.1 *With probability* 1, *for every* t $(0 \leq t \leq T)$, f_t *is* t*-good.*

Proof: The algorithm A is simply a deterministic EREW PRAM algorithm, and Beame, Kik, and Kutylowski [1] show that one input can affect at most 4^t cells after t steps. □

Theorem 10.1 *Let* A *be a randomized EREW PRAM algorithm that allegedly merges two lists of size* n *with probability greater than* $\frac{1}{2}$. *Then* A *runs in at least* T *steps.*

Proof: By Theorem 3.1, it suffices to show that for any deterministic algorithm A' running in less than T steps over the input distribution, A' computes f correctly with probability at most $\frac{1}{2}$. From Lemma 10.1, only $4^T \leq r-2 =$ cells are affected by the first input (which is the only input that changes in the distribution). However, this input affects $r-1$ output cells. Thus at least one of the output cells that should be affected is not. Since each input map occurs with probability $\frac{1}{2}$, the output is correct with probability at most $\frac{1}{2}$. □

11 Conclusion

We have shown lower bounds on the time required to solve many problems on randomized Exclusive Write PRAMs. In all cases, these were asymptotically equivalent to the corresponding deterministic lower bounds. In fact we do not know of any function which can be computed asymptotically faster using a randomized Exclusive Write PRAM rather than a deterministic Exclusive Write PRAM, and we leave this as an open problem.

Another open problem is to close the constant factor gap between the randomized upper and lower bounds for computing the OR function. We conjecture that the lower bound must be improved. Unfortunately, the method of Dietzfelbinger et al. [4] for obtaining a tight lower bound for computing OR deterministically does not seem to help for proving a tight randomized lower bound for the distribution we chose. Specifically, there is a function of degree about $n^{.63}$ which outputs 0 for the vector of 0 inputs, and 1 for each vector of inputs with at most one 1. Then using the lower bound of [4], we could obtain a deterministic lower bound of about $.46 \log n$ for this function, which would translate into the same lower bound for the OR function. (This function is given in Example 1 in Nisan and Szegedy [19].)

Noting Parberry and Yan [20] give a $\log_{2+\sqrt{2}} n \approx .57 \log n$ step algorithm which computes zero on inputs of weight 0 and 1 on inputs of weight 1, we can see that using our input distribution we could at best achieve a lower bound of about $.57 \log n$.

However, one could choose a distribution giving weight to more than just vectors of weight 0 and 1, in order to come closer to the deterministic lower bound. However, any improvements in our the randomized lower bound would depend on either (1) proving lower bounds on the degree of certain types of functions which output 0 on vectors of weight 0, and 1 for all (or at least many) vectors with weights up to a given constant, or (2) improving the techniques of [3, 20] to take into account "critical *groups* of bits".

Acknowledgements

I would like to thank Vijaya Ramachandran for many helpful comments and discussions.

References

[1] P. Beame, M. Kik, and M. Kutylowski. Information broadcasting by exclusive-read prams. *Para. Process. Lett.*, 4:159–169, 1994.

[2] S. Chaudhuri. Sensitive functions and approximate problems. In *Proc. 34th Symp. on Found. of Comp. Sci.*, pages 186–193, 1993.

[3] S. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, February 1986.

[4] M. Dietzfelbinger, M. Kutylowski, and R. Reischuk. Exact lower time bounds for computing boolean functions on CREW PRAMs. *J. Comput. System Sci.*, 48:231–254, 1994.

[5] F. Fich, M. Kowaluk, M. Kutylowski, K. Loryś, and P. Ragde. Retrieval of scattered information by EREW, CREW, and CRCW PRAMs. In *Proc. 3rd Scand. Workshop on Alg. Theory*, pages 30–41. Lec. Notes in Comp. Sci., Vol. 621, 1992.

[6] F. E. Fich, F. Meyer auf der Heide, P. Ragde, and A. Wigderson. One, two, three ...infinity: Lower bounds for parallel computation. In *Proc. 17th Symp. on Theory of Computing*, pages 48–58, 1985.

[7] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial time hierarchy. *Math. Syst. Theory*, 17(1):13–28, 1984.

[8] J. Gil, Y. Matias, and U. Vishkin. Towards a theory of nearly constant time parallel algorithms. In *Proc. 32nd Symp. on Found. of Comp. Sci.*, pages 698–710, 1991.

[9] J. Gil and L. Rudolph. Counting and packing in parallel. In *Proc. 15th Intl. Conf. on Parallel Processing*, pages 1000–1002, 1986.

[10] L. A. Goldberg, M. Jerrum, and P. D. MacKenzie. An $\Omega(\sqrt{\log \log n})$ lower bound for routing in optical networks. In *Proc. ACM Symp. on Para. Alg. and Arch.*, pages 147–156, 1994.

[11] T. Hagerup. Fast parallel space allocation, estimation and integer sorting. Technical Report MPI-I-91-106, Max-Planck-Institut für Informatik, Saarbrücken, 1991.

[12] T. Hagerup. Fast deterministic processor allocation. In *4th ACM-SIAM Symp. on Disc. Alg.*, pages 1–10, 1993.

[13] T. Hagerup and M. Kutylowski. Fast integer merging on the EREW PRAM. In *Proc. 19th Intl. Coll. on Automata, Languages, and Programming*, pages 318–329, 1992.

[14] T. Hagerup and M. Nowak. Parallel retrieval of scattered information. In *Proc. 16th Intl. Coll. on Automata, Languages, and Programming*, pages 439–450, 1989.

[15] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, chapter 17, pages 869–941. MIT Press/Elsevier, 1990.

[16] M. Kutylowski. Time complexity of boolean functions on CREW PRAMs. *SIAM J. Comput.*, 20:824–833, 1991.

[17] P. MacKenzie. The random adversary: A lower bound technique for randomized parallel algorithms and its application to load balancing. Manuscript.

[18] N. Nisan. CREW PRAMs and decision trees. *SIAM J. Comput.*, 20:999–1007, 1991.

[19] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 462–474, 1992.

[20] I. Parberry and P. Y. Yan. Improved upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 20(1):88–99, 1991.

[21] P. Ragde. The parallel simplicity of compaction and chaining. In *Proc. 17th Intl. Coll. on Automata, Languages, and Programming*, pages 744–751, 1990.

[22] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 77–82, 1987.

[23] M. Szegedy. *Algebraic methods in lower bounds for computational models with limited communication.* PhD thesis, University of Chicago, 1989.

[24] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proc. 18th Symp. on Found. of Comp. Sci.*, pages 222–227, 1977.