OPTIMAL SEPARATIONS BETWEEN CONCURRENT-WRITE PARALLEL MACHINES

Ravi B. Boppana *
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

ABSTRACT

We obtain tight bounds on the relative powers of the Priority and Common models of parallel random-access machines (PRAMs). Specifically we prove that:

(1) The Element Distinctness function of n integers, though solvable in constant time on a Priority PRAM with n processors, requires $\Omega(A(n,p))$ time to solve on a Common PRAM with $p \geq n$ processors, where

$$A(n,p) = \frac{n \log n}{p \log(\frac{n}{p} \log n + 1)}.$$

(2) One step of a Priority PRAM with n processors can be simulated on a Common PRAM with p processors in $O(A(n,p))$ steps.

As an example, the results show that the time separation between Priority and Common PRAMs each with n processors is $\Theta(\log n / \log \log n)$.

1. INTRODUCTION

A parallel random-access machine (PRAM) is a model of parallel computation consisting of a set of processors that communicate via shared memory. In this paper, multiple processors are allowed to simultaneously read from or write to the same memory cell, with two different rules for handling write conflicts. The Priority rule chooses the value of the lowest-indexed processor writing to a given cell. The Common rule compels all processors writing to the same memory cell to write the same value. A Priority[n] machine (Common[n] machine) is a PRAM with n processors following the Priority rule (Common rule). For more information about the PRAM model, see for example Grolmusz and Ragde (1987).

The problem studied in this paper is the relative power of the Priority and Common rules. Priority is at least as strong as Common, but how much stronger is it? Our main results are a tight upper and lower bound on the cost of simulating a Priority machine by a Common machine. The lower bound applies to the Element Distinctness[n] function: given n integers decide whether or not they are all distinct. Though Element Distinctness[n] is solvable in constant time on a Priority[n] machine, our lower bound shows that it requires $\Omega(A(n,p))$ time to solve on a Common[p] machine for $p \geq n$, where

$$A(n,p) = \frac{n \log n}{p \log(\frac{n}{p} \log n + 1)}.$$

Our upper bound matches the lower bound by showing that each step of a Priority[n] machine can be simulated by a Common[p] machine in $O(A(n,p))$ steps. As an example, the results show that the time separation between Priority[n] machines and Common[n] machines is $\Theta(\log n / \log \log n)$.

We also prove a lower bound for computing Element Distinctness[n] on a Priority[p] machine with bounded memory. "Bounded memory" means that the number of memory cells may not grow with the size of the integers, though it may be an arbitrary function of n. We show that solving Element Distinctness[n] on a Priority[n] machine with bounded memory requires $\Omega(\log n / \log \log n)$ time.

The motivation for studying the above problems is to understand the relationship between ideal models of parallel computation and more feasible models. On one hand, algorithm designers prefer the power of the Priority model. On the other hand, computer architects would find the Common model more feasible to implement in hardware. If there were efficient simulations of Priority machines by Common machines, then

* Supported by an NSF Mathematical Sciences Postdoctoral Fellowship and a Henry Rutgers Research Fellowship.

both the algorithm designer and the computer architect would be happy. Our results show that attempting to preserve the number of processors will result in a time-costly simulation.

The proofs of the lower bounds use some interesting methods from information theory. In particular, the progress of an algorithm will be measured by the entropies of associated random variables.

The remainder of this paper is organized as follows. Section 2 compares the new results to previous work. Section 3 sketches the proof of the lower bound for Element Distinctness on Priority machines with bounded memory. Section 4 sketches the proof of the tight lower bound for Element Distinctness on Common machines. Finally Section 5 gives the simulation of Priority by Common.

2. RELATION TO PREVIOUS WORK

Two lower bounds on the Common PRAM complexity of the Element Distinctness function were previously known. Fich, Meyer auf der Heide, and Wigderson (1986) showed that a Common[n] machine solving Element Distinctness[n] requires $\Omega(\log\log\log n)$ time. Ragde, Steiger, Szemerédi, and Wigderson (1988) improved the lower bound to $\Omega(\sqrt{\log n})$. Our results strengthen the previous results in two ways: the lower bound is improved to the best possible $\Theta(\log n/\log\log n)$, and it is generalized to the case when the number of processors is greater than n.

Three simulations of Priority machines by Common machines were previously known. Kučera (1982) showed that one step of a Priority[n] machine can be simulated on a Common[n^2] machine in constant time. Chlebus, Diks, Hagerup, and Radzik (1988) improved the simulation to only require a Common[$n\log n$] machine. Fich, Ragde, and Wigderson (1988) showed that one step of a Priority[n] machine can be simulated on a Common[n] machine in $O(\log n/\log\log n)$ time. Our simulation generalizes to the case when p, the number of Common processors, is an arbitrary function of n. Notice that our simulation bound matches the previous bounds in the special cases $p = n$ and $p = n\log n$.

Grolmusz and Ragde (1987) obtained separations of $\Omega(\log\log\log n)$ time between other PRAM models, such as between the Priority[n] model and the so-called Collision[n] model. Our methods should be able to improve their bounds, but we have not yet verified this belief.

Beame and Hastad (1987) showed that computing the Parity function of n bits on a Priority[$n^{O(1)}$] machine requires $\Theta(\log n/\log\log n)$ time. Cook, Dwork, and Reischuk (1986) showed that computing the OR of n bits on a PRAM without concurrent writes requires $\Theta(\log n)$ time. Neither of their methods appears to be useful for the separation problems considered here.

3. PRIORITY WITH BOUNDED MEMORY

This section will show that a Priority[n] machine with bounded memory requires $\Omega(\log n/\log\log n)$ time to solve Element · Distinctness[n]. Besides being of independent interest, this result will be useful for proving the main lower bound of the next section. The best upper bound known seems to be the trivial $O(\log n)$ time bound. Ajtai, Karabeg, Komlós, and Szemerédi (1988) presented a *randomized* Priority[n] machine with bounded memory for sorting n numbers (and hence solving Element Distinctness [n]) which runs in expected time $O(\log n/\log\log n)$.

We will show the following general time-processor tradeoff.

Theorem 3.1: *Every Priority[p] machine with bounded memory that solves Element Distinctness[n] requires $\Omega(B(n,p))$ time, where*

$$B(n,p) = \left(\frac{\log n}{\log(\frac{p}{n}\log n + 1)}\right).$$

This theorem is proved by first simulating a PRAM on a structured machine called a merging machine (lemma 3.3), and then showing a lower bound for merging machines (lemma 3.5). A *merging machine* consists of a set of processors $\{P_i\}$ operating on a set of variables V. At a given time, each processor P_i will know a particular variable set $V_i \subseteq V$. (Initially these sets are empty.) Each processor will also know the partial order Γ on V that is the union of the total orders of each V_i. A given processor at a given time decides, based on the partial order Γ, to union its variable set either with another processor's variable set or with a single variable. The merging machine is said to *sort* V if at the end of each computation path the partial order Γ has become a total order of V.

The proof of the simulation result uses a Ramsey-theoretic argument similar to one given by Meyer auf der Heide and Wigderson (1985). First we require some definitions. Let D be a totally-ordered set. A *D-function* is a function f whose domain is of the form D^V for some finite set $V = var(f)$ called the *variable set* of f. The dimension of f, written $\dim(f)$, is the cardinality of its variable set. Let $\mathcal{F}$ be a finite collection of D-functions. The *dimension* of $\mathcal{F}$, written $\dim(\mathcal{F})$, is the maximum dimension of all functions in $\mathcal{F}$. The *range* of $\mathcal{F}$, written $range(\mathcal{F})$, is the union of the ranges of the functions in $\mathcal{F}$. For $C \subseteq D$, let $\mathcal{F}|_C$ be the collection of C-functions obtained by restricting every function f in $\mathcal{F}$ to the domain $C^{var(f)}$.

Say that two elements x and y of D^V are *equivalent* if for every i and j in V, we have $x_i < x_j$ if and only if $y_i < y_j$. The equivalence classes induced are called

the *order classes* of D^V. Say that $\mathcal{F}$ is *invariant* if for every f in $\mathcal{F}$ and every order class O of $D^{var(f)}$, the function f is constant on O. The following result is an easy consequence of Ramsey's theorem (see Graham, Rothschild, and Spencer (1980), theorem 1.9).

Lemma 3.2: *There is a function f_1 such that the following holds: If $\mathcal{F}$ is a collection of D-functions such that $|D| \geq f_1(|\mathcal{F}|, dim(\mathcal{F}), |range(\mathcal{F})|, c)$, then there is a subset $C \subseteq D$ of cardinality at least c such that $\mathcal{F}|_C$ is invariant.*

With this Ramsey-theoretic background, the simulation will now be given.

Lemma 3.3: *If there is a Priority[p] machine with bounded memory that solves Element Distinctness[n] in time t, then there is a merging machine with p processors that sorts n variables in time t.*

Proof: Consider a particular PRAM satisfying the hypothesis with variable set V and an infinite domain D of integers. By adding an extra memory cell $m+1$, we may assume that every processor reads and writes at every time step. Define the write-index function $W_{i,j}$ with domain D^V and range $\{1, 2, \ldots, m, m+1\}$ such that $W_{i,j}(x)$ gives the index of the memory cell that processor P_i writes to at time j on input x. Let $R_{i,j}$ be the analogous read-index function. Consider the collection of D-functions $\mathcal{F} = \{W_{i,j}\} \cup \{R_{i,j}\}$ for $1 \leq i \leq p$ and $1 \leq j \leq t$. The cardinality of $\mathcal{F}$ is $2pt$, its range has cardinality at most $m+1$, and its dimension is n. We can apply lemma 3.2 to obtain a subset $C \subseteq D$ of cardinality n such that $\mathcal{F}|_C$ is invariant on C^V.

We will now construct the merging machine. Suppose by induction we have described its behaviour for all times preceding j; we will describe what a processor P_i should do at time j. There is a partial order Γ on V representing the global order information learned so far. Let x be some vector in C^V whose coordinates are distinct and consistent with Γ. Let $c = R_{i,j}(x)$ be the memory cell it reads. Let P_k be the last processor before time j to have its value written into cell c on input x (provided there is such a processor.) In the merging machine we will make the new variable set of P_i be the union of the old variable sets of P_i and P_k. (If there was no such processor P_k, then we adjoin the variable originally in cell c provided there was one.) This completes the description of the merging machine.

Why is this merging machine guaranteed to sort? Suppose that some path of its computation ends with the variables i and j incomparable. Consider any vector x in C^V whose coordinates are distinct, consistent with Γ, and such that no coordinate lies between x_i and x_j in value. Let x' be the same vector except that $x'_i = x'_j = x_i$. From the simulation above, it follows that our original PRAM does not distinguish between these two inputs (*i.e.*, it either accepts or rejects both). But this contradicts the hypothesis that the PRAM solves the Element Distinctness problem. □

The proof of the lower bound for merging machines will use some concepts from Shannon's information theory. Given a random variable $\mathbf{X}$ with countable range, its *entropy* is

$$H(\mathbf{X}) = \sum_x -\Pr[\mathbf{X} = x] \log_2 \Pr[\mathbf{X} = x].$$

By convention, the *entropy* of a function f with finite domain is $H(f) = H(f(\mathbf{v}))$, where $\mathbf{v}$ is a random variable with uniform distribution on $dom(f)$ (the domain of f). The most useful property of entropy is subadditivity: the entropy of $(\mathbf{X}, \mathbf{Y})$ is at most the sum of the entropies of $\mathbf{X}$ and $\mathbf{Y}$.

Entropy will be used to measure the complexity of graphs and partial orders. Recall that a *coloring* of a graph is a function on its vertex set assigning distinct values to adjacent vertices. A *layering* of a partial order is an order-preserving mapping of the partially-ordered set into some totally-ordered set. Since colorings and layerings are functions, it makes sense to speak of their entropies. The following lemma shows that every graph with few edges has a coloring with low entropy.

Lemma 3.4: *Every graph with n vertices and m edges has a coloring with entropy at most $L(\frac{m}{n})$, where*

$$L(x) = (x+1)\log_2(x+1) - x\log_2(x) \leq \log_2((x+1)e).$$

Proof: Given such a graph, consider the following greedy coloring of it. Taking the vertices one at a time, set the color $\chi(v)$ of a vertex v to be the minimum nonnegative integer not used by any of the previous neighbors of v. Set $d(v)$ to be the number of previous neighbors of v. By definition $\chi(v) \leq d(v)$. Consequently if $\mathbf{v}$ is a random vertex with uniform distribution, then

$$E[\chi(\mathbf{v})] \leq E[d(\mathbf{v})] = \frac{m}{n}.$$

Since $\chi(\mathbf{v})$ is concentrated on the nonnegative integers, it follows from a result in Csiszár and Körner (1982, corollary 3.12) that

$$H(\chi(\mathbf{v})) \leq L(E[\chi(\mathbf{v})]) \leq L(\frac{m}{n}),$$

which completes the proof. □

With this information-theoretic background, the lower bound will finally be proved.

Lemma 3.5: *Every merging machine with p processors that sorts n variables requires $\Omega(B(n,p))$ time.*

Proof: For $p \geq n^2$, the lower bound is a trivial $\Omega(1)$. For $p \leq \frac{n}{\log n}$, the lower bound is $\Omega(\frac{n}{p})$, which is also trivial since it takes that long just to access all the variables. Thus from now on assume $\frac{n}{\log n} \leq p \leq n^2$.

Suppose we have a merging machine with p processors that sorts a variable set V of cardinality n. To prove the lower bound we will follow a particular path of its computation. We will maintain a layering ℓ of the current partial order Γ and variable sets $\{V_i\}$ for each processor. Define the comparison graph $C_\ell(V_i, V_j)$ on V consisting of those variable pairs $\{a, b\}$ such that $a \in V_i$, $b \in V_j$, $a \neq b$, and $\ell(a) = \ell(b)$. Since V_i and V_j are totally ordered in Γ, this graph is a matching.

We will strive to maintain the above objects so that the following invariants hold after time t for $m = 9 \log p$:

(A) The entropy of ℓ is at most $(L(\frac{pm}{n}) + 3)t$, and

(B) For every $1 \leq i < j \leq p$, the graph $C_\ell(V_i, V_j)$ has at most m edges.

Let us see why these invariants would yield the asserted lower bound. At the end of the computation path, when Γ should be a total order, the entropy of ℓ must be $\log_2 n$. Thus by invariant (A) the time taken is at least $\log_2 n/(L(\frac{pm}{n}) + 3)$ which gives the desired bound.

We will establish the invariants (A) and (B) by induction on the time t. The case $t = 0$ is trivial. Assuming the invariants at time t, we will maintain them for time $t+1$. Let $\{V_i'\}$ be the new variable sets formed by the merges at time $t+1$, where V_i' is the union of the old variable sets V_i and V_{j_i} (we ignore the case of adjoining only a single variable to V_i, which is even easier to handle.)

Step 1: (Refine Γ and ℓ to resolve the new comparisons.) Consider the graph G that is the union of the graphs $C_\ell(V_i, V_{j_i})$ for $1 \leq i \leq p$. By invariant (B), the graph G has at most pm edges. Thus by lemma 3.4, there is a coloring χ of G with entropy at most $L(\frac{pm}{n})$. Assign an arbitrary total order to the colors of χ. Define the function $\ell' = (\ell, \chi)$, ordering its range in lexical order. Define the partial order Γ' on V to be the refinement of Γ that decides the comparison of a in V_i and b in V_{j_i} according to the comparison of $\ell'(a)$ and $\ell'(b)$. Notice that ℓ' is a layering of Γ'.

Step 2: (Refine ℓ' further to maintain invariant (B).) Since every new variable set is the union of two previous ones, by invariant (B) the graph $C_{\ell'}(V_i', V_j')$ has at most $4m$ edges for every $1 \leq i < j \leq p$. We will reduce the bound from $4m$ to m. Choose a random function f from V to $\{1, 2, \ldots, 8\}$ (all 8^n possibilities being equally likely), and consider the function $\ell'' = (\ell', f)$, again ordering its range in lexical order. Since the graph $C_{\ell'}(V_i', V_j')$ is a matching, the events "$f(a) = f(b)$" over all of its edges $\{a, b\}$ are mutually independent events each having probability $\frac{1}{8}$ of occurring. By Chernoff's bound (1952) and the choice $m = 9 \log p$, we have

$$\Pr[|C_{\ell''}(V_i', V_j')| \geq m] \leq (\frac{1}{4})^{-m}(\frac{3}{4})^{-3m}(\frac{1}{8})^m(\frac{7}{8})^{3m} = (\frac{343}{432})^m \leq \frac{1}{p^2}.$$

Taking the disjunction over all $1 \leq i < j \leq p$ gives

$$\Pr[\exists i < j \text{ s.t. } |C_{\ell''}(V_i', V_j')| \geq m] \leq \binom{p}{2} \cdot \frac{1}{p^2} < 1.$$

Hence there is a fixed function f and corresponding function ℓ'' such that the graph $C_{\ell''}(V_i', V_j')$ has fewer than m edges for every $1 \leq i < j \leq p$.

Step 3: (Complete the induction.) We will show that the layering ℓ'' of the partial order Γ' and the new variable sets satisfy the invariants (A) and (B) for time $t + 1$. The construction of ℓ'' in Step 2 guarantees that it satisfies invariant (B). By the subadditivity property of entropy, it follows that

$$H(\ell'') \leq H(\ell) + H(\chi) + H(f) \leq (t+1) \cdot (L(\frac{pm}{n}) + 3),$$

which establishes invariant (A). This completes the inductive proof. □

4. THE COMMON MODEL

This section will show the following lower bound for Common machines solving the Element Distinctness function.

Theorem 4.1: *Every Common[p] machine that solves Element Distinctness[n] requires $\Omega(A(n, p))$ time for $p \geq n$.*

This theorem is again proved by first simulating such a PRAM on a structured machine called a merging and distinctness-checking machine (lemma 4.3), and then showing a lower bound for such structured machines (lemma 4.6). A *merging and distinctness-checking machine* consists of a set of processors $\{P_i\}$ operating on a set of variables V with all the capabilities of a merging machine plus one additional feature called distinctness-checking. The machine has a set of *rows*, each row r having an associated *coordinate set* $c(r)$. A processor P_i at a given time will, depending on the current partial order, choose a pair $(r, \vec{a})$ called its *access pair*, where r is a row and $\vec{a}$ in $V_i^{c(r)}$ is a vector of different variables. (Once an access pair is chosen, it may never be chosen at a later time.) The processor will then merge its variable set as in merging machines. A pair $\{\vec{a}, \vec{b}\}$ of vectors in $V^{c(r)}$ is said to *cover* a pair $\{a, b\}$ of variables if $\vec{a}$ and $\vec{b}$ agree on all coordinates but one, and on that coordinate they take the values a and b. The *distinctness graph DG* on

V consists of those variable pairs $\{a,b\}$ for which the following holds: There are two access pairs $(r,\vec{a})$ and $(r,\vec{b})$ chosen at different times in the computation such that the pair $\{\vec{a},\vec{b}\}$ covers $\{a,b\}$. The machine is said to *solve* Element Distinctness on V if every pair $\{a,b\}$ of variables is either comparable in the partial order Γ or is an edge in the distinctness graph DG.

The simulation uses a Ramsey-theoretic argument similar to one given by Ragde, Steiger, Szemerédi, and Wigderson (1988). Let $\mathcal{F}$ and $\mathcal{G}$ be two collections of D-functions for some totally-ordered set D. Say that $\mathcal{F}$ is *reducible* to $\mathcal{G}$ if for every function f in $\mathcal{F}$ and every order class O of $D^{var(f)}$, there is a function g in $\mathcal{G}$ and a one-to-one mapping $\rho: var(g) \rightarrow var(f)$ such that f agrees with g^ρ on the class O. (Here g^ρ refers to the function with domain $D^{var(f)}$ obtained in the natural way from g and ρ.) Say that $\mathcal{G}$ is *one-to-one* if (1) every function in $\mathcal{G}$ is one-to-one, and (2) every two distinct functions in $\mathcal{G}$ have disjoint ranges. Finally say that $\mathcal{F}$ is *canonical* if it is reducible to some one-to-one collection of D-functions. The following result is a finite version of a result due to Meyer auf der Heide and Wigderson (1985), which in turn is based on the canonical Ramsey theorem due to Erdős and Rado (see Graham, Rothschild, and Spencer (1980), lemma 5.5.3).

Lemma 4.2: *There is a function f_2 such that the following holds: If $\mathcal{F}$ is a collection of D-functions such that $|D| \geq f_2(|\mathcal{F}|, dim(\mathcal{F}), c)$, then there is a subset $C \subseteq D$ of cardinality at least c such that $\mathcal{F}|_C$ is canonical.*

With this Ramsey-theoretic background, the simulation will now be given.

Lemma 4.3: *If there is a Common[p] machine that solves Element Distinctness[n] in time t, then there is a merging and distinctness-checking machine with p processors that solves Element Distinctness of n variables in time $2t$.*

Proof: Consider a particular PRAM that satisfies the hypothesis with variable set V and infinite domain D of integers. By doubling the amount of time, we can assume that the write-index function $W_{i,j}$ and the read-index function $R_{i,j}$ are identical for every i and j. Let $\mathcal{F} = \{W_{i,j}\}$ for $1 \leq i \leq p$ and $1 \leq j \leq t$. Applying lemma 4.2 it follows that there is a subset $C \subseteq D$ of cardinality n such that $\mathcal{F}|_C$ is canonical, *i.e.*, reducible to some one-to-one collection $\mathcal{G}$ of C-functions.

We now construct the merging and distinctness-checking machine. Each row of memory will be indexed by a different function in $\mathcal{G}$; the coordinate set of the row will be the variable set of the associated function. Suppose that we have constructed the machine for all times preceding j, and are trying to decide what a processor P_i should do at time j. Let x be some vector in C^V with distinct coordinates consistent with the current partial order Γ, and let O be the order class of C^V containing x. By the definition of "reducible", there is a function g in $\mathcal{G}$ and a one-to-one mapping $\rho: var(g) \rightarrow var(f)$ such that the function $W_{i,j}$ agrees with g^ρ on the class O. Identify the mapping ρ with a vector $\vec{k}$ of distinct variables from $V^{var(g)}$. If the pair $(g,\vec{k})$ has never been used before, call it a distinctness-checking case; otherwise call it a merging case. The construction divides into these two cases.

Merging case: As in lemma 3.3, consider the last processor P_l to use the pair $(g,\vec{k})$ before time j on input x (if there is more than one, choose one arbitrarily). In the merge step of the simulation, the new variable set of processor P_i will be the union of the old variable sets of processors P_i and P_l.

Distinctness-checking case: Processor P_i chooses the access pair $(g,\vec{k})$ in its distinctness-checking step and skips its merging step.

Why does the above merging and distinctness-checking machine solve Element Distinctness? Suppose that some path in its computation ends with the variable pair $\{i,j\}$ incomparable in the partial order Γ and not an edge in the distinctness graph DG. Let x be some vector in C^V whose coordinates are distinct, consistent with Γ, and such that no coordinate lies between x_i and x_j in value. Let x' be the same vector except that $x'_i = x'_j = x_i$. From the simulation above, it follows that the original PRAM does not distinguish between these two inputs, which contradicts the hypothesis that the original PRAM solves Element Distinctness. □

The proof of the lower bound for merging and distinctness-checking machines uses the notion of covering a graph with multipartite graphs. A *partial function* on a set V is a function f whose domain is a subset of V. The *graph* of f on V, written $K(f)$, consists of those pairs $\{a,b\}$ for which a and b are in the domain of f and $f(a) \neq f(b)$; notice that $K(f)$ is a complete multipartite graph on $dom(f)$. The *cost* of f is

$$cost(f) = \frac{|dom(f)|}{|V|} \cdot H(f).$$

Given a collection $\mathcal{F}$ of partial functions on V, define its *graph* (*cost*) to be the union (sum) of the graphs (costs) of the functions in $\mathcal{F}$. Finally given a graph G on V, define its *content* to be

$$content(G) = \min_{K(\mathcal{F}) \supseteq G} cost(\mathcal{F}).$$

A useful result due to Fredman and Komlós (1984) says that the content of the complete graph on n vertices is $\log_2 n$. (For another proof, see Körner (1986).)

The above notions will be generalized to multivariate functions. A *partial multivariate function* on V is a

function f whose domain is a subset of $V^{c(f)}$ for some set $c(f)$ called the *coordinate set* of f. A set of vectors in $V^{c(f)}$ is said to be *i-adjacent* (or just *adjacent*) if the vectors agree on all coordinates but the ith one. The *multivariate graph* of f on $V^{c(f)}$, written $\vec{K}(f)$, consists of those adjacent pairs $\{\vec{a}, \vec{b}\}$ in the domain of f for which $f(\vec{a}) \neq f(\vec{b})$. The *graph* of f on V, written $K(f)$, consists of those pairs $\{a, b\}$ that are covered (see the definition of merging and distinctness-checking machines) by some edge $\{\vec{a}, \vec{b}\}$ in $\vec{K}(f)$. The *cost* and *profit* of f are

$$cost(f) = \frac{|dom(f)|}{|V|} \cdot H(f),$$

$$profit(f) = \frac{|dom(f)|^{3/2}}{|V|} \cdot \sqrt{2|c(f)|}.$$

Given a collection $\mathcal{F}$ of partial multivariate functions on V, define its *graph* (*cost*, *profit*) to be the union (sum, sum) of the graphs (costs, profits) of the functions in $\mathcal{F}$. The following auxiliary lemma is used to prove lemma 4.5.

Lemma 4.4: *For every subset S of a set V^c, there is a function h with domain S such that (i) if $h(\vec{a}) = h(\vec{b})$ then $\vec{a}$ and $\vec{b}$ are adjacent, and (ii) the graph $\vec{K}(h)$ has at most $|S|^{3/2}\sqrt{2|c|}$ edges.*

Proof: For each positive integer k, define iteratively the set S_k to be the largest adjacent set in $S - \bigcup_{j<k} S_j$ (breaking ties arbitrarily). Given a vector $\vec{a}$ in S, define $h(\vec{a})$ to be the value of k for which $\vec{a} \in S_k$. As the sets S_k are adjacent, part (i) is trivially satisfied.

To establish part (ii), orient each edge $\{\vec{a}, \vec{b}\}$ of $\vec{K}(h)$ from $\vec{a}$ to $\vec{b}$ provided $h(\vec{a}) < h(\vec{b})$. It suffices to show that each vector $\vec{a}$ in S has indegree at most $\sqrt{2|S||c|}$. Given a vector $\vec{a}$ in S and i in c, let n_i be the number of i-adjacent in-neighbors of $\vec{a}$. By the Cauchy-Schwarz inequality, it suffices to show that $\sum_{i \in c} n_i^2 \leq 2|S|$.

To establish this assertion, label the i-adjacent in-neighbors of $\vec{a}$ by $\vec{b}_{i,j}$ for $1 \leq j \leq n_i$. Abbreviate $h(\vec{b}_{i,j})$ by $g(i,j)$. Notice that the $g(i,j)$ are distinct for every choice of i and j (or else $\vec{a}$ would have been included in $S_{g(i,j)}$). Without loss of generality, assume the $\vec{b}_{i,j}$ were ordered so that $g(i,n_i) < \cdots < g(i,1)$. Since $\{\vec{b}_{i,j}, \ldots, \vec{b}_{i,1}, \vec{a}\}$ is an adjacent set, the maximum property of S_k implies that $|S_{g(i,j)}| \geq j+1$. Taking the sum over all i in c and $1 \leq j \leq n_i$ gives

$$|S| \geq \sum_{i,j} |S_{g(i,j)}| \geq \sum_{i,j} j + 1$$

$$= \sum_i \frac{n_i(n_i+3)}{2},$$

which establishes the assertion of the last paragraph and hence completes the proof. □

The following lemma, a multipartite extension of a result in Ragde, Steiger, Szemerédi, and Wigderson (1988), shows how to reduce with small penalty a collection of multivariate functions to a collection of univariate functions.

Lemma 4.5: *If $\mathcal{F}$ is a collection of partial multivariate functions on a set V, then*

$$content(K(\mathcal{F})) \leq cost(\mathcal{F}) + L(profit(\mathcal{F})).$$

Proof: By lemma 3.4, it suffices to show there is a collection $\mathcal{G}$ of partial univariate functions on V such that $cost(\mathcal{G}) \leq cost(\mathcal{F})$ and the graph $K(\mathcal{F}) - K(\mathcal{G})$ has at most $profit(\mathcal{F}) \cdot |V|$ edges. Furthermore, it suffices to show this assertion for a singleton collection $\mathcal{F} = \{f\}$.

To establish this assertion, apply lemma 4.4 to $S = dom(f)$ to obtain the promised function h. Let $\mathcal{G} = \{g_k\}$ be the collection of partial multivariate functions obtained by setting the function g_k, for each k in the range of h, to be the restriction of the function f to the domain $h^{-1}(k)$. Since $h^{-1}(k)$ is an adjacent set, the collection $\mathcal{G}$ is isomorphic to a collection of partial univariate functions. Since $\mathcal{G}$ is a partition of f, it follows that $cost(\mathcal{G}) \leq cost(f)$. Since $\vec{K}(h)$ has at most $|dom(f)|^{3/2}\sqrt{2|c(f)|}$ edges, the graph $K(f) - K(\mathcal{G}) \subseteq K(h)$ has at most $profit(f) \cdot |V|$ edges, which completes the proof. □

With the above background on multipartite covering, the lower bound for merging and distinctness-checking machines will finally be proved.

Lemma 4.6: *Every merging and distinctness-checking machine with p processors that solves the Element Distinctness[n] function requires at least $\Omega(A(n,p))$ time for $p \geq n$.*

Proof: For $p \geq n \log n$ the lower bound is trivial, so assume that $p \leq n \log n$. Given a machine satisfying the hypothesis with variable set V running in time t, we will follow one path of its computation. The merging steps will be handled as in the proof of lemma 3.5. That proof gave a layering ℓ of the partial order Γ obeying invariant (A). Let CG be the comparability graph of Γ. Since ℓ is a coloring of CG, it follows that

$$content(CG) \leq H(\ell) \leq \left[L(\frac{9p \log p}{n}) + 3 \right] t.$$

To handle the distinctness-checking steps, we will assign a collection of partial multivariate functions to the computation. Each row r of the machine will have its own partial multivariate function f_r with coordinate

set $c(r)$. The function f_r will map a vector $\vec{a}$ in $V^{c(r)}$ to the time when the pair $(r, \vec{a})$ was chosen as an access pair (if it ever was chosen). Setting $\mathcal{F} = \{f_r\}$, it follows from the definition of distinctness graph that $K(\mathcal{F}) = DG$. Since a processor accesses only one cell per time step, the cost of $\mathcal{F}$ is at most $\frac{pt}{n} \cdot \log_2 t$. Since a processor's variable set can at most double at each step, the coordinate sets $c(r)$ will have cardinality at most 2^t. It follows that the profit of $\mathcal{F}$ is at most $\frac{\sqrt{2^{t+1}}}{n} \cdot (pt)^{3/2}$. Applying lemma 4.5 gives

$$content(DG) \leq \frac{pt}{n} \cdot \log_2 t + L(\frac{\sqrt{2^{t+1}}}{n} \cdot (pt)^{3/2}).$$

We can now complete the lower bound. Since the union of the graphs CG and DG is the complete graph, the result of Fredman and Komlós (1984) implies that either $content(DG) \geq \frac{3}{4} \log_2 n$ or $content(CG) \geq \frac{1}{4} \log_2 n$. The first case gives $pt \log_2 t = \Omega(n \log n)$, which is equivalent to $t = \Omega(A(n,p))$. The second case gives $t = \Omega(B(n,p))$, which implies $t = \Omega(A(n,p))$ for $p \geq n$. □

5. SIMULATIONS

This section will show how to simulate a Priority machine on a Common machine.

Theorem 5.1: *One step of a Priority[n] machine can be simulated with $O(A(n,p))$ steps of a Common[p] machine.*

Proof: The simulation first performs some precomputation. Choose t to be the least positive integer satistfying $d\lceil \frac{n}{t} \rceil \leq p$ for $d = \lceil \log_t n \rceil$; this choice of t is $O(A(n,p))$. Choose a mapping f from $\{1,\ldots,n\} \times \{1,\ldots,d\}$ to $\{1,\ldots,t\}$ such that (a) if $1 \leq i < j \leq n$ then the string $f(i,1)\ldots f(i,d)$ strictly precedes $f(j,1)\ldots f(j,d)$ in lexical order, and (b) if $1 \leq s \leq t$ then $|f^{-1}(s)| \leq p$; such a mapping is easy to construct.

The nontrivial part of the simulation is to eliminate the Priority writes. Suppose that at a given time the Priority processor P_i attempts to write into a cell $c_i \in N$. In the Common machine, there will be a set of ordinary memory cells indexed by $N \times \{1,\ldots,t\}^{<d}$, and a set of special memory cells indexed by $\{1,\ldots,n\}$; for convenience assume these cells are initialized to 0 although this assumption can be eliminated. For $1 \leq k \leq d$, define cell $c_{i,k}$ to be $(c_i, f(i,1)\ldots f(i,k-1))$. Again for convenience, assume there are nd Common processors $Q_{i,k}$ where $1 \leq i \leq n$ and $1 \leq k \leq d$; later the number of processors will be reduced to p. At time s, say that processor $Q_{i,k}$ is *active* if $f(i,k) = s$. An active processor $Q_{i,k}$ will first read ordinary cell $c_{i,k}$. If it read a '1', then it will write a '1' to special cell i; otherwise it will write a '1' to ordinary cell $c_{i,k}$. At the end of this computation, the special cell i has a '0' in it iff processor P_i is a lexically-first writer. Thus it is easy to determine which Priority processors' values should be written.

Although the above simulation used nd processors, notice that at most p processors are active at any particular time. Consequently only p processors are needed, each imitating one of the active processors $Q_{i,k}$. □

6. REFERENCES

M. Ajtai, D. Karabeg, J. Komlós, and E. Szemerédi (1988), "Sorting in average time $o(\log n)$," preprint.

P. Beame and J. Hastad (1987), "Optimal bounds for decision problems on the CRCW PRAM," *19th STOC*, 83–93.

H. Chernoff (1952), "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Stat.* 23, 493–509.

B. S. Chlebus, K. Diks, T. Hagerup, and T. Radzik (1988), "Efficient simulations between concurrent-read concurrent-write PRAM models," *13th Symp. Math. Found. Comp. Sci., Lecture Notes in Comp. Sci.* 324, Springer-Verlag, 231–239.

S. A. Cook, C. Dwork, and R. Reischuk (1986), "Upper and lower bounds for parallel random access machines without simultaneous writes," *SIAM J. Comp.* 15, 87–97.

I. Csiszár and J. Körner (1982), *Information Theory: Coding Theorems for Discrete Memoryless Systems*, Academic Press.

F. E. Fich, F. Meyer auf der Heide, and A. Wigderson (1986), "Lower bounds for parallel random-access machines with unbounded shared memory," *Advances in Computing Research.*

F. E. Fich, P. L. Ragde, and A. Wigderson (1988), "Simulations among concurrent-write PRAMs," *Algorithmica* 3, 43–52.

M. Fredman and J. Komlós (1984), "On the size of separating systems and families of perfect hash functions," *SIAM J. Alg. Disc. Meth.* 5, 61–68.

R. L. Graham, B. L. Rothschild, and J. H. Spencer (1980), *Ramsey Theory*, John Wiley and Sons.

V. Grolmusz and P. Ragde (1987), "Incomparability in parallel computation," *28th FOCS*, 89–98.

J. Körner (1986), "Fredman-Komlós bounds and information theory," *SIAM J. Alg. Disc. Meth.* 7, 560–570.

L. Kučera (1982), "Parallel computation and conflicts in memory access," *Inf. Proc. Letters* 14, 93–96.

F. Meyer auf der Heide and A. Wigderson (1985), "The complexity of parallel sorting," *25th FOCS*, 532–540.

P. Ragde, W. L. Steiger, E. Szemerédi, and A. Wigderson (1988), "The parallel complexity of element distinctness is $\Omega((\log n)^{1/2})$," *SIAM J. Disc. Math.* 1, 399–410.