REMOVING RAMSEY THEORY: LOWER BOUNDS WITH SMALLER DOMAIN SIZE

Jeff Edmonds
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract: Boppana [B89] proves a lower bound separating the PRIORITY and the COMMON PRAM models that is optimal to within a constant factor. However, an essential ingredient in his proof is a problem with an enormously large input domain. In this paper, I achieve the same lower bound with the improvement that it applies even when the computational problem is defined on a much more reasonably sized input domain. My new techniques provide a greater understanding of the partial information a processor learns about the input. In addition, I define a new measure of the dependency that a function has on a variable and develop new set theoretic techniques to replace the use of Ramsey theory (which had forced the domain size to be large).

1 Introduction

Ramsey Theory has been extremely useful in proving lower bounds for problems defined on huge input domains. (e.g, [B89]). Given a fixed algorithm, the input domain is restricted so that the given algorithm, when run on the restricted domain, falls within a simpler class of algorithms (e.g, the class of comparison based algorithms). A lower bound is then proved on the time required to solve the problem using an algorithm from this simpler class. If the initial input domain is too small, then this technique fails because a restriction of the input domain with the desired properties might not exist.

It is important to obtain lower bounds for problems defined on small domains. Such lower bounds can provide a deeper understanding into what can and cannot be done by the model. Sometimes, when a problem is restricted to a small domain, the time required to solve it strictly decreases. For example, consider the problem of finding the maximum element of a set of n numbers. This problem has time complexity $\Theta(\log\log n)$ on PRIORITY or COMMON PRAM for general inputs [FMW86], but when the elements composing the input are restricted to lie within the range $[1..n^k]$, it can be done in $O(k)$ time [FRW88].

The parallel random access machine (PRAM) is a natural model of parallel computation that is used both for algorithm design and for obtaining lower bounds. On this model, processors communicate with one another via shared memory. During each time step, each processor is able to write to one memory cell and read from another. We are interested in how quickly the processors are able to gain information. For lower bounds, the processors are allowed to do an unbounded amount of computation between communication steps and each memory cell is allowed to hold a value of unbounded size. This is not unreasonable, because in real computers a communication step takes thousands of CPU cycles and transfers large blocks of data. Besides, this assumption only makes the lower bound stronger.

The two models considered in this paper, PRIORITY and COMMON, are both concurrent read concurrent write (CRCW) PRAMs, which differ only in the way they resolve write conflicts. If a number of processors concurrently write to the same cell, then on PRIORITY, the processor

with the highest priority (i.e. lowest index) of those writing to a cell is able to write his value. On COMMON, the algorithm must guarantee that if a concurrent write occurs, then all the processors writing to the cell at this time must write the same value.

A tight lower bound of $\Theta(\log n)$ has been obtained on the time to compute the *OR* of n bits on a concurrent read exclusive write (CREW) PRAM [CDR86]. As well, PRIORITY (and hence COMMON) with $n^{O(1)}$ processors requires $\Theta(\log n / \log\log n)$ time steps to compute the *PARITY* of n bits [BH87]. However, neither of these methods appears to be useful in differentiating between different the write conflict resolution methods used in PRIORITY and COMMON.

A number of simulations of PRIORITY by COMMON have been obtained. [K88] gave a constant time general simulation of PRIORITY on COMMON which requires the number of processors to increase from $p = n$ to $p = n^2$, where n is the length of the input. This was improved [FRW88] to $p = n^{1+\epsilon}$ and later [CDHR88] to $p = n \log n$. When both models have $p = n$ processors, [FRW88-2] show that the one step of PRIORITY can be simulated in $O(\log n / \log\log n)$ time steps on COMMON. [B89] and [R] independently showed how these algorithms could be combined giving a tradeoff between n and p.

Fich, Meyer auf der Heide, and Wigderson [FMW86] first separated the models using the Element Distinctness problem, a problem closely related to sorting. An input $\langle x_1, \ldots, x_n \rangle \in [1..d]^n$ is said to be *element distinct* if each variable x_i has a distinct value. In other words, for all $i, j \in [1..n]$, if $i \neq j$ then $x_i \neq x_j$. This problem can be solved in constant time on PRIORITY with n processors. With this number of processors, [FMW86] proved a $\Omega(\log\log\log n)$ lower bound on COMMON. The lower bound was later improved to $\Omega(\sqrt{\log n})$ by Ragde, Steiger, Szemerédi, and Wigderson [RSSW88] and to $\Theta\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ by Boppana [B89], matching to within a constant factor the upper bound that follows the simulation of PRIORITY by COMMON. In addition, Boppana proves that if the number of memory cells is bounded as the input domain grows, then Element Distinctness takes just as long to solve on the PRIORITY model as on the COMMON model. In the present paper, I prove the same lower bounds. Theorem 1 is a lower bound for the PRIORITY model with bounded memory and Theorem 2 is a lower bound for the COMMON model with unbounded memory. The difference between these results a the previous ones is that the results in [RSSW88] and [B89] require the input domain to be huge, namely the n variables take on values in the range $[1..d]$, where d is a huge tower of exponentials, while in the present paper, the results are proved with a much smaller domain, namely $d \in 2^{2^{\Omega(n)}}$. Most importantly, the new techniques presented provide a greater understanding of the power of concurrent write shared memory in parallel computation.

All the lower bound results mentioned use the adversarial argument. The following is an outline of this technique with an emphasis on what is done differently within this paper. Given a fixed algorithm, if an insufficient number of time steps have been performed, the adversary finds a input that should be *accepted* and one that should be *rejected* which are indistinguishable to, say, processor P_1 (i.e., P_1 is in the same state on both of these inputs at the end of the computation). It is a reasonable restriction to require processor P_1 to know the solution, because if any other processor knew the answer, he could tell P_1 in one extra time step.

The adversary need not choose these two inputs until the end of the computation. Instead, she maintains a set of inputs $\mathcal{D}_t$ that are still being considered until time t. The actions that each processor takes during time step t depend on the input selected from this restricted domain $\mathcal{D}_t$. The adversary, however, has defined this domain $\mathcal{D}_t$ to have the property that, when restricted to

inputs from this domain, the actions of each processor for time t depends on only a small subset of the input variables. The adversary is then able to consider these actions as functions of these input variables. These functions, however, may be quite complex. Therefore, the adversary restricts the input domain further to a subdomain $\mathcal{D}_{t+1} \subseteq \mathcal{D}_t$ on which all these functions have a more simple structure. This simple structure ensures, among other things, that the actions of each processor at time $t+1$ depend only on a small, but slightly larger, subset of the input variables. This process continues one time step at a time.

It is sometimes more intuitive to consider what each processor "knows" about the input, than it is to consider the set of inputs $\mathcal{D}_t$. When a processor is in a particular state, he is formally said to **know** a fact if it is true for every input such that on this input the processor is in the state in question and the input is still considered possible (i.e. it is in $\mathcal{D}_t$). Some of the information known by a processor is said to be **fixed**. By this, I mean that, for all inputs considered possible by the adversary (i.e. in $\mathcal{D}_t$), this information is true. At time t, the processor may choose to take some action because of this fixed knowledge. Because these facts are true over all inputs considered possible, the action is performed on all of these inputs. We will say that the actions of the processor do not *depend* on such fixed information, but only on *non-fixed* information. Each time step, the processor gains more non-fixed information. The adversary will choose some information to reveal to all of the processors. Revealing information amounts to restricting the input domain $\mathcal{D}_t$ to those inputs consistent with the information. The purpose of doing this is two fold. Revealing information that a processor already knows makes this information fixed. Hence, his actions would no longer depend on this information. Revealing information that a processor does not know (intuitively the convex hull of his knowledge) can make it possible to define more succinctly what he does in fact know. This paper provides a better understanding of the knowledge gained by the processors.

[FMW86] showed that processors are essentially only able to gain knowledge in two ways. Processors gain one of the types of knowledge by reading the values written by other processors. In this way, a processor is able to learn the values of 2^t variables in t time steps. In $\log n$ time steps, he is able to learn the entire input and can then compute the answer to any problem in one additional time step. The other type of knowledge is gained by learning about the interactions between the processors. (For example, n processors can compute the OR of n boolean variables in one time step by having each processor write to cell 1 if and only if his variable has the value 1.) When a processor reads a value from a cell, he is said to interact with one of the processors who wrote this value. Which cell a processor reads or writes to at time t is defined by an addressing function that maps each input to the memory cell addressed. Instead of speaking of the two processors interacting, it is often easier to speak of their addressing functions interacting.

Because it is hard to pin down how much an individual processor knows about the interactions, it is helpful for the adversary to reveal all the interactions to all of the processors. However, the cell at which an interaction occurs is not included in this information. To understand how much relevant information the processors gain from this, it is important to understand the difference between two addressing functions accessing the same cell and these two addressing functions interacting. For example, if one addressing function maps the values of a variable x_α in a 1-1 way to memory cells and another uses the same mapping except with the value of x_β, then they access the same cell if and only if $x_\alpha = x_\beta$. If it is known that these addressing functions interacted, then it is known that they accessed the same cell and hence that $x_\alpha = x_\beta$. The known optimal algorithms for Element Distinctness use this fact. On the other hand, if many processors write concurrently to a cell, then a reader "interacts" with no more than one of them. Therefore, knowing that two addressing functions have not interacted does not necessarily imply that they have accessed different cells.

Hence, x_α and x_β may or may not be equal.

When the number of memory cells is bounded as in Theorem 1, the adversary can fix the interactions between the addressing functions by making all of them constant. Each addressing function can be made constant by reducing the input domain by a factor proportional to the memory size. However, if the number of memory cells is unbounded as in Theorem 2, then this cannot always be done without revealing the entire input. In this case, the adversary has four other ways of fixing the interactions between addressing functions. The first method finds a subdomain of inputs on which the addressing functions that depend on exactly the same set of variables are either equal or disjoint. Hence, they either always interact or never interact. (See Lemma 4.)

The second method ensures that if two addressing functions depend on different sets of variables, then there is a variable on which one depends heavily and on which the other is constant. It follows that these functions access the same cell on only a small proportion of the inputs. These inputs can be removed later. [FMW86], [RSSW88], and [B89] restrict the domain so that the addressing functions are either constant or 1-1. This paper defines b-varying which is a more general measure of the dependency a function has on a variable and is interesting in its own right. (See Lemma 5.)

Unlike the first two methods, the third method to ensure that two addressing functions do not interact does not ensure that they access different cells. If more than one processor concurrently wrote to the same cell on the COMMON model, then by the definition of the model, they must all write the same value. Later, when a processor reads this cell, the adversary chooses one of the writers and reveals that the reader read from this writer. The reader would have no way of knowing whether or not any other processor also wrote to the cell. In this way, the adversary has freedom to choose which processors interact. In fact, different readers might be chosen to interact with different writers, even if they all accessed the same cell. (See Lemma 6.) Note that on the PRIORITY model, this is not possible. Each processor reading a cell reads from and, hence, interacts with a specific processor: the one with the highest priority.

The final method uses a refinement of the element distinctness graph introduced in [FMW86]. As mentioned above, it is possible that two addressing functions access the same cell if and only if a pair of variables have the same value. If no other processor writes to this cell (see the third method), then the adversary must reveal whether or not they have accessed the same cell. This information is recorded by covering the edge between these two variables in the element distinctness graph. The adversary ensures that these addressing functions do not interact by not allowing these variables to be equal. (See Lemma 6.) Graph theoretic techniques prove that if an insufficient number of time steps have been performed, then some edge $\{x_\alpha, x_\beta\}$ remains uncovered. (See Lemma 7.) It will follow that no processor knows whether or not these variables are equal.

The adversaries in [RSSW88] and in [B89] use multi-variable Ramsey Theory at each time step to reduce the domain to a sub-cube S^n of inputs (where $S \subseteq [1..d]$). This has the effect of revealing how the processor interact, but it also restricts the domain a great deal. Thus, the initial domain must be very large. In this paper, the adversary's subdomain of inputs does not form a symmetric sub-cube as before, but is allowed to be a more general subset. The subdomain of inputs is described using a new representation of the set of possible processor states. This set of states is restricted as information is revealed (fixed) and is expanded as the processors gain information that has not been fixed.

Interesting new combinatorial techniques are developed to obtain and maintain the desired properties. As well, without the symmetry on the addressing functions and on the domain imposed

in [FMW86], [RSSW88], and [B89], processors are able to learn partial information about whether a particular pair of variables has the same value. I extend the notion of the element distinctness graph used in these papers and use it to record this partial information.

The remainder of this paper is organized as follows. Section 2 proves the lower bound for Element Distinctness on PRIORITY with bounded memory. Section 3 proves the lower bound on COMMON with unbounded memory. Lemmas 4, 5, 6, and 7 used in Section 3 are proved in sections 4, 5, 6, and 7. Some open problems are given in Section 8.

2 PRIORITY PRAMs with Bounded Memory

Theorem 1 *If $d > m^{2^{(1+\epsilon)n}}$, where $\epsilon > 0$ is a constant, then Element Distinctness defined on the input domain $[1..d]^n$ requires $\Omega\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ time steps on a PRIORITY PRAM with p processors and m memory cells.*

Before proving this theorem, some definitions are presented.

After only one time step on a PRAM, the state of a processor can depend on the value of every input variable (e.g, if the processors compute the *OR* function). Part of this information can be gained by knowing which cells other processors did or did not write to. If the input domain is restricted so that the cells addressed by each processor are fixed, then the set of possible states that a processor can be in is greatly restricted. An algorithm is said to be $(\mathcal{D}, t)$**-oblivious** if, for each processor, the cells that it addresses during the first t steps are the same for all inputs in $\mathcal{D}$.

If an algorithm is oblivious, then at time t in the computation, for each processor, there is a small set of variables on which the processor's state might depend. Boppana [B89] proved that such sets must have the property that they could be formed by t steps of a p processor **merging machine**. The sets of variables $\mathcal{V}_{\langle 1,t\rangle}, \ldots, \mathcal{V}_{\langle P,t\rangle} \subseteq \{x_1, \ldots, x_n\}$ are said to have this property if there exists a set $\mathcal{V}_{\langle P,t'\rangle}$ for each processor $P \in [1..p]$ and intermediate time step $t' \in [0..t-1]$ such that $\mathcal{V}_{\langle P,0\rangle}$ contains a single variable for each P and for each $t' \in [1..t]$, $\mathcal{V}_{\langle P,t'\rangle}$ is either $\mathcal{V}_{\langle P,t'-1\rangle}$ plus one extra variable or is the union of $\mathcal{V}_{\langle P,t'-1\rangle}$ and $\mathcal{V}_{\langle P',t''\rangle}$ for some other processor $P' \in [1..p]$ and some previous time step $t'' \in [0..t'-1]$.

Claim 1 *If a PRAM algorithm is $(\mathcal{D}, t)$-oblivious then for each processor P, there is a fixed set of inputs variables $\mathcal{V}_{\langle P,t\rangle}$ such that, for inputs in $\mathcal{D}$, the state of P at the end of step t is uniquely determined by the values of these variables. Furthermore, the sets of variables $\mathcal{V}_{\langle 1,t\rangle}, \ldots, \mathcal{V}_{\langle P,t\rangle}$ have the property that they could be formed by t steps of a p processor merging machine.*

Proof of Claim 1: For time $t = 0$, the set $\mathcal{V}_{\langle P,0\rangle}$ is defined to contain only the variable initially assigned to processor P. Because the cells addressed by the processors are fixed, it is fixed which cell processor P reads at time t. It is also fixed whether or not the cell had been previously written to. If not and if the cell initially contained the value of a variable, then the variable learned is added to $\mathcal{V}_{\langle P,t\rangle}$. If the cell had been written to, then the last time step $t_w \in [1..t]$ in which the cell was written to is also fixed. Let P_w be the processor with the highest priority of those who wrote to the cell at time t_w. Define $\mathcal{V}_{\langle P,t\rangle}$ to be the union of $\mathcal{V}_{\langle P,t-1\rangle}$ and $\mathcal{V}_{\langle P_w,t_w\rangle}$. Inductively, processor P's state can only depend on the values of the variables in $\mathcal{V}_{\langle P,t\rangle}$. ■

Consider a $(\mathcal{D}, t)$-oblivious algorithm. Processor P is said to **see** the variables in the set $\mathcal{V}_{\langle P,t\rangle}$ at time t. The adversary finds a partition $\Pi_t^1, \Pi_t^2, \ldots, \Pi_t^{q_t}$ of the input variables $\{x_1, \ldots, x_n\}$ with the property that no processor sees more than one variable per part Π_t^i. (If necessary, the adversary gives processors the values of more variables so that each sees the exactly one variable per part).

A part Π_t^i of the partition is referred to as a **subproblem**. In the inputs considered, two variables will have the same value only if they are in the same subproblem. In order to prove that this entire input is element distinct, it is necessary to prove that each of these subproblems is element distinct. This is difficult for the processors to do if each only sees one variable per subproblem. This notion was introduced by Ragde et al. [RSSW88].

A **vantage point** is any sequence of variables $\mathcal{V} = \left\langle x_{j_1}, \ldots, x_{j_{q_t}} \right\rangle$ such that $x_{j_i} \in \Pi_t^i$ for each $i \in [1..q_t]$. Thus, a vantage point contains exactly one variable from each of the subproblems and the order of the variables is specified by the fixed ordering of the subproblems. Note that the variables $\mathcal{V}_{\langle P,t\rangle}$ seen by processor P at time t form a vantage point if the variable are appropriately ordered. Hence, $\mathcal{V}_{\langle P,t\rangle}$ is referred to as the vantage point from which the processor sees the input. The notion of defining ordered tuples of variables was introduced by Ragde et al. [RSSW88]. This idea is extended in the next definition.

The **view** of an input $\langle v_1, \ldots, v_n\rangle$ seen from the vantage point $\left\langle x_{j_1}, \ldots, x_{j_{q_t}} \right\rangle$ is the sequence $\left\langle v_{j_1}, \ldots, v_{j_{q_t}} \right\rangle$ of the values of the variables that occur in the vantage point. The view seen from $\mathcal{V}_{\langle P,t\rangle}$ is said to be the view seen by processor P at time t. By Claim 1, the state of the processor at the end of time t is uniquely determined by the view that he sees. The adversary maintains a set of views $\mathcal{S}_t \in [1..d]^{q_t}$ to be used as a set of objects, each representing a state that a processor could be in. An example is given in Figure 1. Processor P_1's vantage point is $\mathcal{V}_{\langle P_1,t\rangle} = \langle x_2, x_3, x_6\rangle$ and P_2's is $\mathcal{V}_{\langle P_2,t\rangle} = \langle x_1, x_3, x_5\rangle$. Then, on the input $\langle 7,3,5,0,2,4,8\rangle$, processor P_1 sees the view $\langle 3,5,4\rangle$ and P_2 sees $\langle 7,5,2\rangle$. A key point is that the same view can be used to represent the

Vantage Point of P_1	x_2	x_3	x_6
Vantage Point of P_2	x_1	x_3	x_5
Subproblems Π	x_1 x_2	x_3 x_4	x_5 x_6 x_7
Specific Input	7 3	5 0	2 4 8
View Seen by P_1	3	5	4
View Seen by P_2	7	5	2

Figure 1: The View Seen by a Processor

state of each of the processors. However, being in the state represented by a particular view means something different for each processor. In the above example, the view $\langle 3,5,4\rangle$ is seen by P_1 on the input $\langle 7,3,5,0,2,4,8\rangle$ and by P_2 on the input $\langle 3,1,5,9,4,6,4\rangle$.

Any set of views $\mathcal{S}_t$ can be expanded into the set of inputs such that, for each vantage point, the view seen is contained in $\mathcal{S}_t$. Each vantage point is considered, even if no processor has it. More formally, the set of inputs is defined to be

$$Expand(\mathcal{S}_t) = \Big\{ \langle v_1, \ldots, v_n\rangle \mid \forall \text{ vantage points } \mathcal{V} = \left\langle x_{j_1}, \ldots, x_{j_{q_t}} \right\rangle \text{ (i.e.} \forall i \in [1..q_t], x_{j_i} \in \Pi_t^i\text{)}, \text{ the view } \left\langle v_{j_1}, \ldots, v_{j_{q_t}} \right\rangle \text{ is contained in } \mathcal{S}_t \Big\}.$$

The inputs $\mathcal{D}_t = Expand(\mathcal{S}_t)$ are those considered possible by the adversary. By considering only these inputs, the adversary can ensure that every processor sees a view contained in $\mathcal{S}_t$, and hence is in one of the allowed states.

The proof of Theorem 1 uses an adversarial argument. Formally, an **adversary** is defined to be a function that maps a complete description of a PRIORITY PRAM algorithm running in time $T \in o\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ to two inputs Φ and Φ', one element distinct and the other not, such that processor P_1 is in the same state on both inputs after T steps of the given algorithm. To complete this task, the adversary defines the following constructs, for each time step $t \leq T$,

- a set of views $\mathcal{S}_t \subseteq [1..d]^{q_t}$,
- a partition $\Pi_t^1, \Pi_t^2, \ldots, \Pi_t^{q_t}$ of the input variables,

with the following properties:

1. The given PRAM algorithm is $(\mathcal{D}_t, t)$-oblivious, where $\mathcal{D}_t = Expand(\mathcal{S}_t)$.
2. For each processor P, the vantage point $\mathcal{V}_{\langle P,t\rangle}$ contains exactly one variable from each Π_t^i.
3. The entropy of the partition $\Pi_t^1, \Pi_t^2, \ldots, \Pi_t^{q_t}$ is at most $\left(L\left(\frac{9p\log p}{n}\right)+3\right)t$, where $L(x) = (x+1)\log_2(x+1) - x\log_2(x)$.
4. $|\mathcal{S}_t| \geq \frac{d^{q_t}}{m^{2p(2^{q_t}-2)}}$ (i.e., $\mathcal{S}_t \subseteq [1..d]^{q_t}$ is a large fraction of all possible views).

At the end of the induction, the final set of views $\mathcal{S}_T$ is expanded into a set of inputs $\mathcal{D}_T$. The following properties must hold:

5. $q_T < n$.
6. $|\mathcal{D}_T| \geq \frac{d^n}{m^{2p\ (2^n-2)}}$.

Entropy is defined as follows. Uniformly at random choose a variable $x \in \{x_1, \ldots, x_n\}$. The entropy $H(\Pi) = \sum_{i\in[1..q_t]} -\Pr\left[x \in \Pi_t^i\right]\log_2 \Pr\left[x \in \Pi_t^i\right]$ is the expected number of bits to specify which of the sets Π_t^i that x is contained in.

Initially, with $t = 0$, every processor sees at most one variable. Therefore, it is sufficient to have only one subproblem Π_0^1 (i.e., $q_0 = 1$) and the initial set of views is $\mathcal{S}_0 = \{1, 2, \ldots, d\}$. Hence, the entropy is 0, the size bound $|\mathcal{S}_0| \geq \frac{d}{1}$ is met, and $\mathcal{D}_0$ contains all d^n inputs.

Inductively, suppose that the adversary has defined these constructs for time step $t-1$ so that the induction hypothesis hold. The adversary then defines these constructs for time step t by restricting the set of views $\mathcal{S}_{t-1}$, refining the partition Π_{t-1}, and then expanding the set of views.

2.1 The Restricting Stage

To construct $\mathcal{S}_t$, the adversary constructs $\mathcal{S}_{t-1}^{oblivious}$ from $\mathcal{S}_{t-1}$. By Claim 1, on inputs in $\mathcal{D}_{t-1}$ the state of each processor at the end of step $t-1$ depends only on his view. Therefore, for every view

in $\mathcal{S}_{t-1}$ and for every processor, there is a unique cell that the processor writes to at time t when seeing the view and a unique cell that he reads. In all, each view in $\mathcal{S}_{t-1}$ specifies $2p$ cells to be addressed. There are only m cells; hence, there are m^{2p} addressing possibilities. The adversary fixes one possibility that is used for at least $\frac{|\mathcal{S}_{t-1}|}{m^{2p}}$ of the views and lets $\mathcal{S}_{t-1}^{oblivious} \subseteq \mathcal{S}_{t-1}$ be this subset of views. In order to be able to refer to them latter, define $Address(w, P, t)$ and $Address(r, P, t)$ to be the cells addressed for writing or reading.

2.2 The Refinement Stage

The subproblems need to be repartitioned. There are five requirements of the new partition Π_t^1, $\Pi_t^2, \ldots, \Pi_t^{q_t}$. To satisfy condition (2), it must have the property that no processor sees more than one variable from each part. It must be a refinement of the previous partition Π_{t-1}^1, $\Pi_{t-1}^2, \ldots,$ $\Pi_{t-1}^{q(t-1)}$. Its entropy must meet the bound required for condition (3). For technical reasons, it is necessary that at least one of the subproblems $\Pi_{t-1}^1, \Pi_{t-1}^2, \ldots, \Pi_{t-1}^{q(t-1)}$ is repartitioned. Finally, we require for condition (5) that, after T time steps, the number of parts q_T is strictly less than n. The proof that such a partition exists appears in Boppana's paper [B89], but is not presented here. Here, I will simply draw the connections between what is done there and what is done here.

Boppana's Lemma 3.5 in [B89] proves a lower bound on the time for a merging machine with p processors to sort n variables. Processor P knows the order of the variables in $\mathcal{V}_{\langle P,t \rangle}$. Therefore, the input must agree with the partial order on the variables that is the union of all the total orders of the variables in each $\mathcal{V}_P$. Boppana maintains a *layering* of this partial order. This consists of a mapping l from the n variables to q_t layers such that for each $i \in [1..q_t]$, the variables $l^{-1}(i)$ in the i^{th} layer are incomparable in the partial order. More to our purposes, this partitioning of the variables into layers has that property that no processor knows the value of more than one variable in any one layer. In other words, we can let $\Pi_t^i = l^{-1}(i)$ and this partition will meet our first requirement. Boppana's leveling l is always a refinement of the previous leveling. Therefore, this partition meets our second requirement. The third requirement bounds the **entropy** of the partition $\Pi_t^1, \Pi_t^2, \ldots, \Pi_t^{q_t}$. Boppana ensures that this third requirement is met by maintaining the property that the entropy of the leveling, and hence of the partition, is at most $\left(L\left(\frac{9p\log p}{n}\right) + 3\right)t$, where $L(x) = (x+1)\log_2(x+1) - x\log_2(x)$. The technical requirement is easy. For reasonable algorithms, each part will be repartitioned into many new subproblems. However, if this is not the case, then the adversary can repartition one of the parts anyway. Finally, Boppana ensures that $q_T < n$. His lower bound on the time required for the merging machine to sort is $\Omega\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$. In other words, if $T \in o\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$, then the merging machine does not know the total ordering on the variables. This means that Boppana's layering does not have each variable in its own layer. Therefore, $q_T < n$. It follows that this partition meets our requirements.

2.3 The Expanding Stage

During time step t, a processor can learn the values of more variables. Hence, the view a processor sees at time $t-1$ is a subsequence of his view at time t. As well, a larger number of views are needed to represent the larger number of states in which he may be in. To accommodate these two needs, the adversary **expands** the set of views $\mathcal{S}_{t-1}^{oblivious}$, to form a larger set $\mathcal{S}_t$ of longer views. Note that, even though the set of views $\mathcal{S}_t$ gets larger each time step, the set of inputs $\mathcal{D}_t = Expand(\mathcal{S}_t)$

keeps getting smaller.

The adversary ensures that $\mathcal{S}_t$ only contains views which are considered by all of the processors to be consistent with their knowledge. A processor, after reading a memory cell, considers a view to be consistent if both the part of the view that he saw before the read and the part of the view that the writer saw are consistent. (See Figure 2.) To be careful, a view is included in $\mathcal{S}_t$ if, for every vantage point from time $t-1$, the subview is contained in $\mathcal{S}_{t-1}^{oblivious}$. Note that some of these vantage points are not held by any processor. Define

$$\begin{aligned}\mathcal{S}_t \;=\; \Big\{ \Big\langle\, v^{1,1},\ldots,v^{1,\delta_1},v^{2,1},\ldots,v^{2,\delta_2},\;\ldots,v^{q_{(t-1)},1},\ldots,v^{q_{(t-1)},\delta_{q_{(t-1)}}} \,\Big\rangle \;\Big|\\ \text{for each } k_1\in[1..\delta_1],\; k_2\in[1..\delta_2],\ldots,k_{q_{(t-1)}}\in[1..\delta_{q_{(t-1)}}],\\ \text{the subview } \Big\langle\, v^{1,k_1},v^{2,k_2},\ldots,v^{q_{(t-1)},k_{q_{(t-1)}}} \,\Big\rangle \text{ is contained in } \mathcal{S}_{t-1}^{oblivious} \Big\}.\end{aligned}$$

For $i\in[1..q_{(t-1)}]$, δ_i is the number of new subproblems into which the subproblem Π_{t-1}^i is repartitioned. For $k\in[1..\delta_i]$, $v^{i,k}$ is the value that the view assigns to the k^{th} new subproblem $\Pi_t^{i,k}$. In the example in Figure 2, the processor P_r does not see a variable from the newly partitioned subproblem Π_t^4. The adversary could reveal the value of a variable from this set to P_r adding the variable to the set $\mathcal{V}_{\langle P_r,t\rangle}$ of those seen by P_r. At any rate, the adversary has complete freedom to choose the value (here 0) that is at this index in the view seen by P_r.

Old Subproblems	Π_{t-1}^1	Π_{t-1}^2	Π_{t-1}^3
Repartitioned Subproblems	Π_t^1 Π_t^2	Π_t^3 Π_t^4	Π_t^5 Π_t^6 Π_t^7
Subview Seen by P_r	3	5	4
Subview Seen by P_w	7	5	2
View Seen by P_r	7 3	5 0	2 4 8

Figure 2: Expanding the Set of Views

2.4 $\mathcal{S}_t$ is large

In order to make calculating the size of $\mathcal{S}_t$ easier, the subproblems $\Pi_{t-1}^1,\ldots,\Pi_{t-1}^{q_{(t-1)}}$ will be repartitioned one at a time. Consider expanding a set $\mathcal{S}^{pre}\subseteq[1..d]^q$ of views into the set $\mathcal{S}^{exp}\subseteq[1..d]^{q+\delta-1}$ in order to repartition the subproblem Π_{t-1}^i into δ new subproblems.

Let $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}=\{v \mid \langle\vec{u},v,\vec{u}'\rangle\in\mathcal{S}^{pre}\}$ be the set of values for the i^{th} component that are consistent with $\vec{u}$ in the first $i-1$ components and $\vec{u}'$ in the last $q-i$ components. Then $\mathcal{S}^{pre}=\bigcup_{<\vec{u},\vec{u}'>}\vec{u}\times\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}\times\vec{u}'$. It follows that $\mathcal{S}^{exp}=\bigcup_{<\vec{u},\vec{u}'>}\vec{u}\times\left(\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}\right)^{\delta}\times\vec{u}'=\bigcup_{<\vec{u},\vec{u}'>}\{\,\langle\vec{u},v_1,\ldots,v_\delta,\vec{u}'\rangle \mid v_k\in\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ for each $k\in[1..\delta]\,\}$.

Lemma 1 *If* $|\mathcal{S}^{pre}|\geq\frac{d^q}{m^e}$, *then* $|\mathcal{S}^{exp}|\geq\frac{d^{q+\delta-1}}{m^{e\delta}}$.

Proof of Lemma 1: $\sum_{\langle\vec{u},\vec{u}'\rangle}|\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}|$ has the fixed value $|\mathcal{S}^{pre}|$. Therefore, by convexity, the expression $|\mathcal{S}^{exp}|=\sum_{\langle\vec{u},\vec{u}'\rangle}|\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}|^{\delta}$ is minimized when $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ is the same size for each $\langle\vec{u},\vec{u}'\rangle$. There

are at most d^{q-1} different choices for $\langle \vec{u}, \vec{u}' \rangle$. Therefore, $|\mathcal{S}^{exp}| \geq d^{q-1} \left(\frac{|\mathcal{S}^{pre}|}{d^{q-1}} \right)^{\delta} \geq d^{q-1} \left(\frac{\frac{d^q}{m^e}}{d^{q-1}} \right)^{\delta} \geq \frac{d^{q+\delta-1}}{m^{e\delta}}$. ■

Lemma 2 $|\mathcal{S}_t| \geq \frac{d^{q_t}}{m^{2p\left(2^{q_t}-2\right)}}$.

Proof of Lemma 2: By condition (4) for step $t-1$,

$$|\mathcal{S}_{t-1}| \geq \frac{d^{q_{(t-1)}}}{m^{2p\left(2^{q_{(t-1)}}-2\right)}} \text{ and } |\mathcal{S}_{t-1}^{oblivious}| \geq \frac{|\mathcal{S}_{t-1}|}{m^{2p}} = \frac{d^{q_{(t-1)}}}{m^{2p\left(2^{q_{(t-1)}}-1\right)}}.$$

Applying Lemma 1 for each of the $q_{(t-1)}$ subproblems gives

$$|\mathcal{S}_t| \geq \frac{d^{q_t}}{m^{2p\left(2^{q_{(t-1)}}-1\right)\delta_1\delta_2\ldots\delta_{q_{(t-1)}}}}$$

Since $2^q\delta \leq 2^{q+\delta-1}$ for $\delta \geq 1$ and $q_t = q_{(t-1)} + (\delta_1 - 1) + \ldots + (\delta_{q_{(t-1)}} - 1)$, it follows that $2^{q_{(t-1)}}\delta_1\delta_2\ldots\delta_{q_{(t-1)}} \leq 2^{q_t}$. Furthermore, the technical condition on the refinement of the subproblems is that at least one of the subproblems is partitioned. The ensures that $\delta_1\delta_2\ldots\delta_{q_{(t-1)}} \geq 2$. Thus $\left(2^{q_{(t-1)}} - 1\right)\delta_1\delta_2\ldots\delta_{q_{(t-1)}} \leq \left(2^{q_t} - 2\right)$ and $|\mathcal{S}_t| \geq \frac{d^{q_t}}{m^{2p\left(2^{q_t}-2\right)}}$. ■

2.5 The PRAM Algorithm Is $(\mathcal{D}_t, t)$-Oblivious

Let $\mathcal{D}_t = Expand(\mathcal{S}_t)$. Then condition (1) for step t is satisfied.

Lemma 3 *The PRAM algorithm is $(\mathcal{D}_t, t)$-oblivious.*

Proof of Lemma 3: Let $\mathcal{D}_{t-1}^{oblivious} = Expand\left(\mathcal{S}_{t-1}^{oblivious}\right)$. The first step is to prove that $\mathcal{D}_t \subseteq \mathcal{D}_{t-1}^{oblivious}$. Let $\Phi \in \mathcal{D}_t$. By the definition of $\mathcal{D}_t$, every view $\vec{v}_t$ seen on input Φ is contained in $\mathcal{S}_t$. By the definition of $\vec{v}_t \subset \mathcal{S}_t$, every subview $\vec{v}_{t-1}$ of $\vec{v}_t$ is contained in $\mathcal{S}_{t-1}^{oblivious}$. It follows that every subview $\vec{v}_{t-1}$ seen on input Φ is contained in $\mathcal{S}_{t-1}^{oblivious}$. Therefore, $\Phi \in \mathcal{D}_{t-1}^{oblivious}$. Similarly, $\mathcal{D}_{t-1}^{oblivious} \subseteq \mathcal{D}_{t-1}$ follows easily from $\mathcal{S}_{t-1}^{oblivious} \subseteq \mathcal{S}_{t-1}$.

By condition (1) for step $t-1$, the PRAM algorithm is $(\mathcal{D}_{t-1}, t-1)$-oblivious. Because $\mathcal{D}_t \subseteq \mathcal{D}_{t-1}$, the algorithm is $(\mathcal{D}_t, t-1)$-oblivious. Hence, we only need to prove that the cells accessed at time t are independent of the input in $\mathcal{D}_t$. Let $\Phi \in \mathcal{D}_t$, and consider processor P. Because $\Phi \in \mathcal{D}_t \subseteq \mathcal{D}_{t-1}^{oblivious}$, the view seen by processor P at time $t-1$ is contained in $\mathcal{S}_{t-1}^{oblivious}$. Hence, it follows from the definition of $\mathcal{S}_{t-1}^{oblivious}$ that processor P accesses the cells $Address(w, P, t)$ and $Address(r, P, t)$ at time t. This is true for every processor. ■

2.6 $\mathcal{D}_T$ is Large

This completes the induction for time step t. It remains to show that if the total number of steps is $T \in o\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ then the bound on $|\mathcal{D}_T|$ for condition (6) is satisfied. Refine the subproblems $\Pi_T^1, \ldots, \Pi_T^{q_T}$ into the n singleton sets and expand the set of views $\mathcal{S}_T$ as before forming the set of long views $\mathcal{S}_*$. Each expanded view in $\mathcal{S}_*$ assigns a value to each variable and hence completely specifies an input. Comparing the definitions for $\mathcal{D}_T$ and $\mathcal{S}_*$ will reveal that $\mathcal{D}_T = \mathcal{S}_*$. Lemma 2 gives that $|\mathcal{D}_T| = |\mathcal{S}_*| \geq \frac{d^n}{m^{2p\ (2^n-2)}}$.

2.7 Conclusion

From these constructs, it is easy to find inputs Φ and $\Phi' \in \mathcal{D}_T$ such that Φ is element distinct and Φ' is not. The total number of inputs that are not element distinct inputs is at most $\binom{n}{2}d^{n-1}$. This is strictly less than $\frac{d^n}{m^{2p\ (2^n-2)}}$, the number inputs in $\mathcal{D}_T$, when $d > \binom{n}{2}m^{2p\ (2^n-2)}$. It follows that there exists an element distinct input $\Phi \in \mathcal{D}_T$.

Since $q_T < n$, there is a subproblem Π_T^i containing two different variables: x_α and x_β. After step T, processor P_1 sees at most one of these variables, say $x_\beta \notin \mathcal{V}_{\langle P_1,T\rangle}$. Let v be the value of x_α for Φ. Form the input Φ' from Φ by changing the value of x_β to v so that Φ' is not element distinct. From the definition of $\mathcal{D}_T$, it follows easily that $\Phi' \in \mathcal{D}_T$. By Claim 1, on inputs Φ and Φ', the state of P_1 at the end of step T depends only on the fixed set of input variables $\mathcal{V}_{\langle P_1,T\rangle}$ seen by him. P_1 does not see x_β and therefore cannot distinguish between Φ and Φ', which only differs in the value x_β. ■

3 COMMON PRAMs with Unbounded Memory

Theorem 2 *If $d \geq 2^{2^{(1+\epsilon)n}}$, then Element Distinctness defined on the input domain $[1..d]^n$ requires $\Theta\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ time steps on a COMMON PRAM with p processors and an unbounded number of memory cells.*

With an unbounded number of memory cells, Element Distinctness can be solved in constant time on the PRIORITY model. Therefore, this theorem provides a separation between the PRIORITY and COMMON models.

The proof uses an adversary argument similar to the previous proof. As in that proof, a key concept is the vantage point $\mathcal{V}_{\langle P,t\rangle} = \left\langle x_{j_1}, \ldots, x_{j_{q_t}} \right\rangle$ seen by processor P at time t. In this proof there is another key concept. This is the addressing functions used by the processors and which of these functions interact. For a more detailed overview of the proof, see the Introduction, Section 1.

Fix a COMMON PRAM algorithm. For each processor P and time step t, the algorithm defines the addressing functions $f^w_{\langle P,t\rangle}$ and $f^r_{\langle P,t\rangle} : [1..d]^n \to \mathbb{N}$ which specify the cells that P on input Φ writes into and reads from at time t. Let $\mathcal{F}_t^{write}$ be the collection of addressing functions used by the p processors for writing during time t and $\mathcal{F}_{[1..t]}^{write}$ to be the collection of addressing functions used during the time interval $[1..t]$. Similarly, define $\mathcal{F}_t^{read}$, $\mathcal{F}_{[1..t]}^{read}$, and $\mathcal{F}_{[1..t]} = \mathcal{F}_{[1..t]}^{read} \cup \mathcal{F}_{[1..t]}^{write}$. Even if the addressing functions $f^w_{\langle P,t\rangle}$ and $f^w_{\langle P',t'\rangle}$ happen to be the same function $[1..d]^n \to \mathbb{N}$, they will be considered as separate objects in the collection $\mathcal{F}_{[1..t]}^{write}$ so that when needed we can refer to the unique processor and time step in which the addressing function was used. The subscript $\langle P,t\rangle$ will be dropped as in f_w, when the processor and the time step are irrelevant to the discussion at hand.

Given the addressing functions $\mathcal{F}_{[1..t]}$, one can determine how the processors interact on any input,during the time interval $[1..t]$. Here $t \leq T$ is our current place in the induction. Consider an input $\Phi \in [1..d]^n$ and a read function $f_r \in \mathcal{F}_{[1..t]}^{read}$. Associated with these is the cell c read and the time step at which the read occured. By considering all the write addressing functions in $\mathcal{F}_{[1..t]}^{write}$, we

can determine the last time step t' that this cell was written to. Let $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r) \subseteq \mathcal{F}^{write}_{t'}$ denote the set of write addressing functions that simultaneously wrote to the cell c at time t'. If the cell c read by f_r is blank, then the set of write addressing functions $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$ is defined to be the empty set.

The processors using the write addressing functions in the set $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$ simultaneously write to the same cell. Therefore, by the rules of the COMMON model, all of these processors must write the same value. When the processor using f_r reads this value, he need not be aware of more than one of these writers. Hence, the read function f_r is said to **interact** with only one of the write functions in the set $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$. The adversary is able to choose which of these write functions it will be. The adversary restricts the input domain to a subdomain $\mathcal{D}$ in order to fix which write function each read function interacts with. Let $\Gamma_{[1..t]} : \mathcal{F}^{read}_{[1..t]} \rightarrow \mathcal{F}^{write}_{[1..t]} \cup \{miss\}$ be some fixed function chosen by the adversary. An algorithm is said to be $(\mathcal{D}, t, \Gamma_{[1..t]})$**-oblivious** if for all $f_r \in \mathcal{F}^{read}_{[1..t]}$, either it is the case that for all $\Phi \in \mathcal{D}$, $\Gamma_{[1..t]}(f_r) \in \Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$ or it is the case that $\Gamma_{[1..t]}(f_r) = miss$ and for all $\Phi \in \mathcal{D}$, $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r) = \emptyset$. We say that on inputs in $\mathcal{D}$, the interactions in the computation up until time step t are consistent with $\Gamma_{[1..t]}$.

Claim 2 *If a PRAM algorithm is $(\mathcal{D}, t, \Gamma_{[1..t]})$-oblivious, then for each processor P there is a fixed set of inputs variables $\mathcal{V}_{\langle P,t\rangle}$ (vantage points) such that for inputs in $\mathcal{D}$, the state of P at the end of step t is uniquely determined by the values of these variables. Furthermore, the sets of variables $\mathcal{V}_{\langle P_1,t\rangle}, \ldots, \mathcal{V}_{\langle P_p,t\rangle}$ have the property that they could be formed by t steps of a p processor merging machine.*

Proof of Claim 2: The proof depends heavily on the definition of the COMMON model. When a set of processors simultaneously writes to the same cell, they must all write the same value. Therefore, the information written must be contained in the intersection of the knowledge of these writers. In the lower bound, when a processor reads this value using the addressing function f_r, the adversary reveals to the reader the identity of one of the processors that wrote the value and reveals all of the information that this processor has, i.e. the value of the variables seen by the processor indicated by $\Gamma_{[1..t]}(f_r)$. The reader is unable to discern any additional information from the read. For example, the processor cannot determine whether or not any other processor simultaneously wrote to the same cell as well. For more details see the proof of Claim 1. ■

As in the proof of Theorem 1, the adversary maintains a set of views $\mathcal{S}_t$ which are used to represent the states of the processors. However, in the proof of Theorem 2, the algorithm might not be oblivious on the input domain $\mathcal{D}_t = Expand(\mathcal{S}_t)$. There may be some bad inputs on which the addressing functions do not interact in the fixed way that they should. Instead, the adversary maintains the set of such bad inputs $Bad_{[1..t]} \subseteq Expand(\mathcal{S}_t)$ and proves that this set is not too big.

Consider an addressing function $f \in \mathcal{F}_{[1..t]}$. Suppose that processor P addresses using this function at time t. By Claim 2, when restricted to inputs in $\mathcal{D}_t - Bad_{[1..t]}$, the state of the processor, and hence the function depends only on the values of the variables in the vantage point $\mathcal{V}_{\langle P,t\rangle} = \left\langle x_{j_1}, \ldots, x_{j_{q_t}} \right\rangle$. Because $\mathcal{D}_t = Expand(\mathcal{S}_t)$, the possible tuples of values for this sequence of variables are the views in $\mathcal{S}_t$. Therefore, the input domain for f can be considered to be the set $\mathcal{S}_t$ when viewed as a set of values for the variables in $\mathcal{V}_{\langle P,t\rangle}$. It is interesting to observe that for a different addressing function, the input domain will also be considered to be $\mathcal{S}_t$, but for a different tuple of variables.

The previous papers [FMW86], [RSSW88], and [B89] restrict the input domain to a subdomain $\mathcal{D}_t$ such that for each of the addressing functions f and each of the variables x_j, either f depends in a 1-1 way on x_j or it does not depend on this variable at all. This is is an unreasonable requirement when the input domain is small, because such a subdomain might not exist. Instead, I define a more general measure of the dependency a function has on a variable. The precise definition of b-varying is defined in Section 4.1. The extent to which the function varies is parameterized by the integer b. My adversary maintains a set of views $\mathcal{S}_t$ and, for each addressing function $f \in \mathcal{F}_{[1..t]}$, a set of variables $\mathcal{X}(f) \subseteq \mathcal{V}_{\langle P,t\rangle}$. The required condition is that when viewing $\mathcal{S}_t$ as the input domain, f is b_t-varying with respect to each variable in $\mathcal{X}(f)$ for some integer b_t and is completely independent of the other variables. One problem that might arise is that an addressing function from an earlier time t' might be $b_{t'}$-varying on the set of views $\mathcal{S}_{t'}$. However, the same function might varying much less on the current set of views $\mathcal{S}_t$, i.e. is only b-varying for b considerably smaller than $b_{t'}$. To handle this problem, b_t is set to be a rapidly decreasing function of t ending with b_T being set to be the final value needed. Because b_t is set to be considerable smaller than $b_{t'}$, the adversary can maintain the property that all the function are at least b_t-varying with respect to the variables in $\mathcal{X}(f)$ on the set of views $\mathcal{S}_t$. This has the added benefit of ensuring that the set of variables $\mathcal{X}(f)$ does not change from one time step to the next.

The adversary classifies each read-write pair of addressing functions $\langle f_r, f_w\rangle \in \mathcal{F}^{read}_{[1..t]} \times \mathcal{F}^{write}_{[1..t]}$ based on the sets of variables $\mathcal{X}(f_r)$ and $\mathcal{X}(f_w)$ on which they depend. The functions in the pair are said to be **similar** if $\mathcal{X}(f_r) = \mathcal{X}(f_w)$. They are said to be $\{x_\alpha, x_\beta\}$**-covering** if $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \{x_\alpha\}$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \{x_\beta\}$. Otherwise, they are said to be **unrelated**.

Similar pairs of functions access their cells based on mutual information and hence know a priori whether or not they will access the same cell. $\{x_\alpha, x_\beta\}$-covering pairs could be used by the COMMON algorithm as follows. One of the functions addresses cells in a 1-1 way with the value of x_α. The other uses the same mapping except that it uses the variable x_β in place of x_α. The reader learns whether or not $x_\alpha = x_\beta$ by learning whether or not f_r and f_w access the same cell. Unrelated pairs do not seem to help the algorithm in any way.

The adversary is able choose which interactions $\Gamma_{[1..t]} : \mathcal{F}^{read}_{[1..t]} \rightarrow \mathcal{F}^{write}_{[1..t]} \cup \{miss\}$ that she wants between the addressing functions and then reveals this information to the processors. The input domain is restricted to those inputs on which these interactions occur. Similar pairs of addressing functions depend on the same set of variables $\mathcal{X}(f_r) = \mathcal{X}(f_w)$, so how these pairs interact partitions the set of views $\mathcal{S}_t$. If follows that the adversary is able to fix the interactions between these function to those that reduces $\mathcal{S}_t$ the least. In contrast, the adversary always will reveal that the $\{x_\alpha, x_\beta\}$-covering and the unrelated pairs do not interact. The set $Bad_{[1..t]}$ of inputs mentioned above are defined to be those on which $\{x_\alpha, x_\beta\}$-covering or unrelated pairs do interact. Because $Bad_{[1..t]}$ is defined in this way, saying that the algorithm is $(\mathcal{D}_t - Bad_{[1..t]}, t, \Gamma_{[1..t]})$-oblivious effectively only states that on the inputs in $\mathcal{D}_t$, the similar pairs of addressing functions interact as revealed.

As implied, there are two types of bad inputs, $Bad^{\text{unrelated}}_{[1..t]}$ and $Bad^{\{x_\alpha,x_\beta\}\text{-covering}}_{[1..t]} \subseteq Bad_{[1..t]}$. If, on input Φ, a pair $\langle f_r, f_w\rangle$ of unrelated addressing functions access the same cell, i.e. $f_r(\Phi) = f_w(\Phi)$, then this input is in $Bad^{\text{unrelated}}_{[1..t]}$. On the other hand, the adversary allows $\{x_\alpha, x_\beta\}$-covering pairs of addressing functions to access the same cell, even though the adversary must ensure that they do not interact. For example, f_r and f_w might access the same cell, but the reader might not read the value written by f_w, i.e. $f_w \notin \Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$, because the value was overwritten or written after the read. For a more complex second example, suppose that f_r does

read the value written by f_w. Suppose as well that there is another write function f'_w that writes to this cell at the same time as f_w, i.e. both f_w and f'_w are in $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$. If f'_w is similar to f_r, then the adversary can ensure that f_r interacts with f'_w, i.e. choose $\Gamma_{[1..t]}(f_r) = f'_w$. In this case, the interactions are still consistent with $\Gamma_{[1..t]}$, because the functions in the $\{x_\alpha, x_\beta\}$-covering pair $\langle f_r, f_w \rangle$ do not interact. However, if $f_w \in \Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$ and there is no write function in $\Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r)$ that is similar to f_r, then the input Φ is said to be in $Bad^{\{x_\alpha, x_\beta\}\text{–covering}}_{[1..t]}$.

Formally the adversary maintains, for each time step t, the following constructs:

- A set of "views", $\mathcal{S}_t \subseteq [1..d]^{q_t}$,
- A partition $\Pi^1_t, \Pi^2_t, \ldots, \Pi^{q_t}_t$ of the input variables,
- A function $\Gamma_{[1..t]} : \mathcal{F}^{read}_{[1..t]} \rightarrow \mathcal{F}^{write}_{[1..t]} \cup \{miss\}$ specifying the interactions,

From these the following constructs are derived:

- For each addressing function $f \in \mathcal{F}_{[1..t]}$, the set of variable $\mathcal{X}(f) \subseteq \mathcal{V}_{\langle P,t \rangle} \subseteq \{x_1, \ldots x_n\}$ on which the function depends.
- The function $Type$ specifying the type of every read-write pair of addressing functions. Specifically, for each $f_r \in \mathcal{F}^{read}_{[1..t]}$ and $f_w \in \mathcal{F}^{write}_{[1..t]}$,
  - if $\mathcal{X}(f_r) = \mathcal{X}(f_w)$ then $Type(f_r, f_w) =$ similar;
  - if $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \{x_\alpha\}$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \{x_\beta\}$ for $x_\alpha, x_\beta \in \Pi^i_t$ then $Type(f_r, f_w) = \{x_\alpha, x_\beta\}$-covering;
  - otherwise $Type(f_r, f_w) =$ unrelated.
- The set of inputs:
  - $Bad^{\text{unrelated}}_{[1..t]} = \left\{ \Phi \;\middle|\; \begin{array}{l} \exists f_r \in \mathcal{F}^{read}_{[1..t]} \text{ and } f_w \in \mathcal{F}^{write}_{[1..t]} \text{ such that} \\ Type(f_r, f_w) = \text{ unrelated and } f_r(\Phi) = f_w(\Phi) \end{array} \right\}$.
  - $Bad^{\{x_\alpha, x_\beta\}\text{–covering}}_{[1..t]} = \left\{ \Phi \;\middle|\; \begin{array}{l} \exists f_r \in \mathcal{F}^{read}_{[1..t]} \text{ and } f_w \in \mathcal{F}^{write}_{[1..t]} \text{ such that} \\ Type(f_r, f_w) = \{x_\alpha, x_\beta\}\text{-covering}; f_w \in \Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r); \\ \text{and there are no write functions in } \Gamma^{\text{alg}}_{[1..t]}(\Phi, f_r) \\ \text{that are similar to } f_r \end{array} \right\}$.
  - $Bad^{\text{covering}}_{[1..t]} = \bigcup_{\substack{\forall x_\alpha, x_\beta \in \Pi^i_t, \\ i \in [1..q_t]}} Bad^{\{x_\alpha, x_\beta\}\text{–covering}}_{[1..t]}$.
  - $Bad_{[1..t]} = Bad^{\text{unrelated}}_{[1..t]} \cup Bad^{\text{covering}}_{[1..t]}$.

The conditions inductively maintained are the following:

1. The PRAM algorithm is $(\mathcal{D}_t - Bad_{[1..t]}, t, \Gamma_{[1..t]})$-oblivious, where $\mathcal{D}_t = Expand(\mathcal{S}_t)$.

2. For each processor P, the vantage point $\mathcal{V}_{\langle P,t\rangle}$ contains exactly one variable from each Π_t^i.

3. The entropy of the partition $\Pi_t^1, \Pi_t^2, \ldots, \Pi_t^{q_t}$ is at most $\left(L\left(\frac{9p\log p}{n}\right)+3\right)t$.

4. $|\mathcal{S}_t| \geq \frac{d^{q_t}}{b^{(pn)^{2T}(2^{q_t}-2)}}$ where $b = 2(npT)^2$ and T is the total time for the algorithm.

5. On the domain $\mathcal{S}_t$, each addressing function $f \in \mathcal{F}_{[1..t]}$ is independent of the variables not in $\mathcal{X}(f)$ and is b_t-varying with respect to $x \in \mathcal{X}(f)$, where $b_t = b^{(pn)^{[T-t+2]}}$ and $b = 2(npT)^2$.

6. For each $f_r \in \mathcal{F}^{read}_{[1..t]}$, if $\Gamma_{[1..t]}(f_r) = f_w$, then $Type(f_r, f_w) =$ similar.

At the end of the induction, the final set of views $\mathcal{S}_T$ is expanded into a set of inputs. Because there are additional complications, a set of inputs $\mathcal{D}_T^i$ is formed for each subproblem Π_T^i. The following properties must hold:

7. The PRAM algorithm is $(\mathcal{D}_T^i - Bad_{[1..T]}, T, \Gamma_{[1..t]})$-oblivious.

8. The number of parts in the partition $\Pi_T^1, \Pi_T^2, \ldots, \Pi_T^{q_T}$ is at most $q_T < n^{0.1}$.

9. $|\mathcal{D}_T^i| \geq \frac{d^n}{b^{(pn)^{2T}\ 2^n}}$, where $b = 2(npT)^2$.

10. On the domain $\mathcal{D}_T^i$, each addressing function $f \in \mathcal{F}_{[1..T]}$ is independent of the variables not in $\mathcal{X}(f)$ and is b-varying with respect to $x \in \mathcal{X}(f)$, where $b = 2(npT)^2$.

11. The set of inputs $\mathcal{D}_T^i$ are balanced in the following sense. For each subproblem Π_T^i, $i \in [1..q_T]$, let $\delta_i = \left|\Pi_T^i\right|$, $\delta = \sum_{j\in[1..i-1]}\left|\Pi_T^j\right|$, and $\delta' = \sum_{j\in[i+1..q_T]}\left|\Pi_T^j\right|$. For each $\vec{u} \in [1..d]^{\delta}$ and each $\vec{u}' \in [1..d]^{\delta'}$, define $\mathcal{V}^i_{\langle\vec{u},\vec{u}'\rangle} \subseteq [1..d]$ to be the set of values such that the set of inputs decomposes as $\mathcal{D}_T^i = \bigcup_{<\vec{u},\vec{u}'>} \vec{u} \times \left(\mathcal{V}^i_{\langle\vec{u},\vec{u}'\rangle}\right)^{\delta_i} \times \vec{u}'$. The set of inputs $\mathcal{D}_T^i$ are balanced in the sense that each nonempty $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ has the same fixed size, i.e. $\exists z,\ \forall\langle\vec{u},\vec{u}'\rangle,\ |\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}| \in \{0,z\}$.

3.1 The Main Steps of the Proof

In the introduction, I list four ways in which the adversary restricts the input domain to fix the interactions between the addressing functions are fixed. The four methods correspond to the following four lemmas. Lemma 4 uses techniques similar to those used in Theorem 1 to handle the similar pairs of addressing functions. Lemma 5 handles the unrelated pairs. Lemmas 6 and 7 handle the $\{x_\alpha, x_\beta\}$-covering pairs. Below, the lemmas are stated and, from them, Theorem 2 is proved. The proofs to Lemmas 4, 5, 6, and 7 are found in Sections 4, 5, 6, 7 respectively.

Lemma 4 *There exists an adversary function whose input is a complete description of a COMMON PRAM algorithm that runs in time $T \in o\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ and whose output consists of the sets of inputs $\mathcal{D}_T^i$, the partition Π_T^1, $\Pi_T^2, \ldots, \Pi_T^{q_T}$ of the input variables, and the function $\Gamma_{[1..T]}$ specifying the interactions, such that the conditions (7-11) are met.*

The proof inductively maintains the conditions (1) and (6) for each time step, by restricting and expanding the set of views $\mathcal{S}_t$.

After Lemma 4, the next step for the adversary is to ensure that there are not too many inputs in $Bad^{\text{unrelated}}_{[1..T]}$. Lemma 5 proves this by producing a sufficient number of inputs that are not in this bad set. The proof uses that fact that the unrelated pairs of addressing functions are b-varying on different sets of variables.

Lemma 5 also ensures that, for many pairs of variables $\{x_\alpha, x_\beta\}$, the processors have not gained too much information about whether or not $x_\alpha = x_\beta$. In the case that the two variables are contained in different subproblems Π^i_T, this cannot be done, because some processor might see both of the variables. On the other hand, if the variables are in the same subproblem, then no processor sees both of them. This would lead us to believe that no processor knows whether they have the same value. However, besides knowing the values of the variables in $\mathcal{V}_{P,T}$, each processor also knows how the addressing functions interacted. This information may have provided the processors with partial information about whether or not $x_\alpha = x_\beta$. Lemma 5 proves that learning the interactions between the similar and the unrelated pairs does not completely reveal this information. The lemma does this by demonstrating for each pair of variables $\{x_\alpha, x_\beta\}$, one element distinct input and one input in which $x_\alpha = x_\beta$, on which the iterations between the similar and the unrelated pairs of addressing functions are as revealed by the adversary. Ideally, there would be a single element distinct input Φ and for each pair of variables, the non-element distinct input would differ from Φ only in the two variables $\{x_\alpha, x_\beta\}$ in question. The lemma accomplishes this, except that there may be a different element distinct input Φ^i for each subproblem Π^i_T.

Lemma 5 *Given the above constructs for time T, there exists, for each subproblem Π^i_T, $i \in [1..q_T]$, an element distinct input Φ^i and there exists, for each pair of variables $\{x_\alpha, x_\beta\}$ contained in the subproblem Π^i_T, a non-element distinct input $\Phi^i_{x_\alpha = x_\beta = v_i}$ such that: $\Phi^i_{x_\alpha = x_\beta = v_i}$ assigns the value v_i to the variables x_α and x_β and is the same as Φ^i on every other variable; the processors interact according to $\Gamma_{[1..t]}$ on the input Φ^i; and if $\Phi^i_{x_\alpha = x_\beta = v_i} \notin Bad^{\{x_\alpha, x_\beta\}-covering}_{[1..T]}$, then the same is true on this input.*

After Lemma 5, what remains is to determine for which pairs of variables $\{x_\alpha, x_\beta\}$ the processors can differentiate between Φ^i and $\Phi^i_{x_\alpha = x_\beta = v_i}$ by knowing how the $\{x_\alpha, x_\beta\}$-covering pairs interact. This is done by considers the "element distinctness" graph on vertex set $\{x_1, \ldots, x_n\}$ and by covering the edge $\{x_\alpha, x_\beta\}$ if the corresponding inputs can be differentiated. The next lemma uses graph theoretic constructs to characterize those edges covered. The undefined terms will be defined in Section 6.

Lemma 6 *There exists a "collection of labeled tuple systems" $\left\{ \left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w} \right) \mid t_w \in [1..T] \right\}$ that cover the edge $\{x_\alpha, x_\beta\}$ if $\Phi^i_{x_\alpha = x_\beta = v_i} \in Bad^{\{x_\alpha, x_\beta\}-covering}_{[1..T]}$. In addition, $\sum_i |\mathcal{A}_i| \leq pT^2$ and $\sum_i |\mathcal{B}_i| \leq pT$.*

The key now is to find an uncovered edge. The proof uses entropy techniques and combines the ideas from Fredman and Komlós, Ragde et al. [RSSW88], and Boppana [B89].

Lemma 7 *There exists a pair of variables $\{x_\alpha, x_\beta\}$, such that: x_α and x_β are contained in the same subproblem Π^i_T for some $i \in [1..q_T]$; the edge $\{x_\alpha, x_\beta\}$ is not covered by the collection of labeled tuple systems; and neither variables are seen by processor P_1, i.e. $x_\alpha, x_\beta \notin \mathcal{V}_{\langle P_1, T \rangle}$.*

Theorem 2 follows easily from these lemmas. Let $\{x_\alpha, x_\beta\}$ be an edge with the properties stated in Lemma 7. From Lemmas 5 and 6, it follows that all the addressing functions interact as revealed by the adversary on the inputs Φ^i and $\Phi^i_{x_\alpha=x_\beta=v_i}$. Hence, by Claim 2, the state of P_1, for these two inputs, at the end of step T depends only on the fixed set of input variables $\mathcal{V}_{\langle P_1,T\rangle}$ seen by him. P_1 does not see x_α or x_β and therefore cannot distinguish between the inputs Φ^i and $\Phi^i_{x_\alpha=x_\beta=v_i}$, which differ only on these variables. Therefore, on these inputs, P_1 is unable to determine whether or not the input is element distinct. ■

4 Induction on Time Steps

Lemma 4 *There exists an adversary function whose input is a complete description of a COMMON PRAM algorithm that runs in time* $T \in o\left(\frac{n}{p}\frac{\log n}{\log(\frac{n}{p}\log n)}\right)$ *and whose output consists of the sets of inputs* $\mathcal{D}^i_T$*, the partition* Π^1_T, $\Pi^2_T, \ldots,$ $\Pi^{q_T}_T$ *of the input variables, and the function* $\Gamma_{[1..T]}$ *specifying the interactions, such that the conditions (7-11) are met.*

Inductively, suppose that the adversary has defined the above constructs for time step $t-1$. First, the adversary restricts the set of views to the subset $\mathcal{S}^{varying}_{t-1} \subseteq \mathcal{S}_{t-1}$ on which the addressing functions in $\mathcal{F}_t$ are $(pt+1)^p b^{n^2} b_t$-varying. Then, the set of views is restricted further to $\mathcal{S}^{similar}_{t-1} \subseteq \mathcal{S}^{varying}_{t-1}$ so that each similar pair interacts in an oblivious way. Finally, the set $\mathcal{S}^{similar}_{t-1}$ is expanded to $\mathcal{S}_t$. On this new set of views, each addressing function $f \in \mathcal{F}_{[1..t]}$ is still b_t-varying with respect to the variables in $\mathcal{X}(f)$.

4.1 The Varying Property

The b-varying property is a general measure of the dependency a function has on a variable. Let f be an addressing function with the view $\mathcal{V}_{\langle P,t\rangle} = \langle x_{j_1}, \ldots, x_{j_q}\rangle$ and let x_{j_l} be a variable within this view. The addressing function f is said to be **b-varying** with respect to the variable x_{j_l} on the set of views $\mathcal{S}$ if and only if $Ind^{f,x_{j_l}}_{\mathcal{S}} \leq \frac{|\mathcal{S}|}{b}$, where $Ind^{f,x_{j_l}}_{\mathcal{S}}$ is the size of the largest subset of $\mathcal{S}$ on which f is independent of x_{j_l}. If f is b-varying with respect to every variable in $\mathcal{X}(f) \subseteq \mathcal{V}_{\langle P,t\rangle}$ and is independent of the other variables, then f is simply said to be b-varying.

The following construction of such a largest subset is not necessary for the proof, but it may provide some insight into the definition of b-varying. To temporarily simplify the notation, consider a function f that depends on the variables $\langle y_1, \ldots, y_q\rangle$ that is defined on a domain $\mathcal{S}$. We construct as follows a set, $IND^{f,y_1}_{\mathcal{S}}$, that is a largest subset of $\mathcal{S}$ on which f is independent of the variable y_1. For each setting $\vec{u}$ of $\langle y_2, \ldots, y_q\rangle$, f must address a fixed cell on the subdomain $IND^{f,y_1}_{\mathcal{S}}$ in order to be independent of y_1. Consider the univariate function $f(y_1, \vec{u})$. The possible values for y_1 are $\{\ v |\ \langle\ v, \vec{u}\ \rangle \in \mathcal{S}\ \}$. Partition these values according to which cell is addressed. Let $C^{f,y_1}_{\mathcal{S}}(\vec{u})$ be the cell addressed by the largest number of values. The function $C^{f,y_1}_{\mathcal{S}}(y_2, \ldots, y_q)$ is independent of y_1, but may depend on the variables $\langle y_2, \ldots, y_q\rangle$. Let $IND^{f,y_1}_{\mathcal{S}} = |\{\ \langle\ v, \vec{u}\ \rangle \in \mathcal{S} \mid f(v, \vec{u}) = C^{f,y_1}_{\mathcal{S}}(\vec{u})\ \}|$. A diagram of this construction is given in figure 3. There is a column for each $\vec{u}$ and a row for each v. Each entry specifies the output of f (eg. c, d, e) on the input $\langle\ v, \vec{u}\ \rangle \in \mathcal{S}$. For each column, the values v are partitioned according to which cell is addressed and the largest such block is marked. Note that output of f need not be the same for different columns. The union of the marked areas is a maximum subset $IND^{f,y_1}_{\mathcal{S}} \subseteq \mathcal{S}$ on which f is independent of y_1.

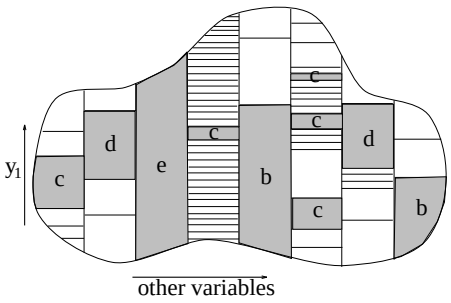

Figure 3: b-varying

4.2 Obtaining the Varying Property

Consider the cells addressed by the processors during time step t on inputs from the domain $\mathcal{D}_{t-1} - Bad_{[1..t]}$. By condition (1) and by Claim 2, the state of processor P at the end of step $t-1$ is uniquely determined by the values of the variables in the vantage point $\mathcal{V}_{\langle P,t-1\rangle}$. In other words, the view in $\mathcal{S}_{t-1}$ seen by the processor determines which cell is addressed at time t. This defines a collection of $2p$ new addressing functions $\mathcal{F}_t$ defined on the domain $\mathcal{S}_{t-1}$.

The adversary finds a subset $\mathcal{S}_{t-1}^{varying} \subseteq \mathcal{S}_{t-1}$ of the views on which these new addressing functions in $\mathcal{F}_t$ are $(pt+1)^p b^{n^2} b_t$-varying. This is done by restricting the set, once for each function-variable pair. A function-variable pair $\langle f, x_{j_l}\rangle$ is found for which the function is neither independent nor $(pt+1)^p b^{n^2} b_t$-varying with respect to the variable on the current set of views. The set is reduced to the largest subset on which the function is independent of the variable. Then another such function-variable pair is found and the set is reduced further. Because each set is a subset of the previous sets, once a function is independent of a variable, it remains independent. The process stops when no more such function-variable pairs exist. Let $\mathcal{S}_{t-1}^{varying}$ be the resulting set of views. On this set, for each addressing function in $\mathcal{F}_t$ and each variable, the function is either $(pt+1)^p b^{n^2} b_t$-varying with respect to the variable or independent of it.

How much does the set get reduced? Consider a set $\mathcal{S}$. Recall that $Ind_{\mathcal{S}}^{f,x_{j_l}}$ is the size of the largest subset of $\mathcal{S}$ on which f is independent of x_{j_l}. The set of views is reduced to such a subset. Because f is not $(pt+1)^p b^{n^2} b_t$-varying with respect to x_{j_l}, we know that the size of this largest subset is greater than $\frac{|\mathcal{S}|}{(pt+1)^p b^{n^2} b_t}$. There are at most $2p$ addressing functions in $\mathcal{F}_t$ and each of these depends on at most 2^t variables. Therefore, the set will be reduced in this way no more than $2p2^t$ times. We can conclude that $\mathcal{S}_{t-1}^{varying} > \frac{|\mathcal{S}_{t-1}|}{\left[(pt+1)^p b^{n^2} b_t\right]^{2p2^t}}$.

4.3 Ensuring that Similar Pairs Interact as Revealed

The following lemma ensures that condition (1) is true for time step t.

Lemma 4.1 *If the PRAM algorithm is* $(Expand\left(\mathcal{S}_{t-1}^{varying}\right) - Bad_{[1..t-1]}, t-1, \Gamma_{[1..t-1]})$*-oblivious, then there exists a set of views* $\mathcal{S}_{t-1}^{similar} \subseteq \mathcal{S}_{t-1}^{varying}$ *such that* $|\mathcal{S}_{t-1}^{similar}| \geq \frac{|\mathcal{S}_{t-1}^{varying}|}{(pt+1)^p}$ *and a* $\Gamma_{[1..t]}$ *for time* t *such that the PRAM algorithm is* $(Expand\left(\mathcal{S}_{t-1}^{similar}\right) - Bad_{[1..t]}, t, \Gamma_{[1..t]})$*-oblivious.*

Proof of Lemma 4.1: Consider an input $\Phi \in Expand\left(\mathcal{S}_{t-1}^{varying}\right) - Bad_{[1..t]}$ and any addressing function $f_r \in \mathcal{F}_t^{read}$. By the definition of $Bad_t = Bad_t^{\text{unrelated}} \cup Bad_t^{\text{covering}}$, either $\Gamma_{[1..t]}^{\text{alg}}(\Phi, f_r)$ contains a write addressing function that is similar to f_r or $\Gamma_{[1..t]}^{\text{alg}}(\Phi, f_r)$ is empty. In the first case, define $\Gamma'_{[1..t]}(\Phi, f_r)$ to be the write addressing function from $f^w \Gamma_{[1..t]}^{\text{alg}}(\Phi, f_r)$ that that is similar to f_r. If there is more than one possibility for f^w, break the tie by choosing the one used by the processor of highest priority. In the second case, let $\Gamma'_{[1..t]}(\Phi, f_r) = miss$.

Let $\vec{v} \in \mathcal{S}_{t-1}^{varying}$ be the view seen by the processor P using f_r on input Φ. We now prove that if $\Gamma'_{[1..t]}(\Phi, f_r) = f_w$, then $\Gamma'_{[1..t]}(\Phi', f_r) = f_w$ for every $\Phi' \in Expand\left(\mathcal{S}_{t-1}^{varying}\right) - Bad_{[1..t]}$ on which P sees the same view $\vec{v}$. Suppose by contradiction, that $\Gamma'_{[1..t]}(\Phi, f_r) = f_w$, and $\Gamma'_{[1..t]}(\Phi', f_r) = f'_w$. By the definition of $\Gamma'_{[1..t]}$, f_r is similar to both f_w and to f'_w. Therefore, $\mathcal{X}(f_r) = \mathcal{X}(f_w) = \mathcal{X}(f_{w'})$. The view $\vec{v}$ specifies the values of the variables in $\mathcal{X}(f_r) \subseteq \mathcal{V}_{\langle P, t-1\rangle}$. Hence, $\vec{v}$ specifies the cells addressed by f_r, f_w and $f_{w'}$. On input Φ, the addressing functions f_r, f_w and f'_w access cells so that f_r reads from f_w. Therefore, the same thing happens on input Φ', proving the claim. The effect of the claim is that, the function $\Gamma''_{[1..t]}(\vec{v}, f_r) = \Gamma'_{[1..t]}(\Phi, f_r)$ is well defined.

Define $\mathcal{S}_{t-1}^{similar}$ to be the largest subset of $\mathcal{S}_{t-1}^{varying}$ on which $\Gamma''_{[1..t]}(\vec{v}, f_r)$ is independent of $\vec{v}$ for each $f_r \in \mathcal{F}_t^{read}$. There are p such read functions f_r and the range of $\Gamma''_{[1..t]}(\vec{v}, f_r)$ is at most $|\mathcal{F}_{[1..t]}^{write} \cup \{miss\}| \leq pt + 1$. Therefore, $|\mathcal{S}_{t-1}^{similar}| \geq \frac{|\mathcal{S}_{t-1}^{varying}|}{(pt+1)^p}$. The adversary chooses $\Gamma_{[1..t]}(f_r)$ $= \Gamma''_{[1..t]}(\vec{v}, f_r)$ for these views $\vec{v}$. ■

The following example will demonstrate why this technique does not work for non-similar pairs. Suppose that there is only one subproblem $\Pi_1 = \{x_1, x_2\}$ and that the set of views is $\mathcal{S}_{t-1}^{varying} = \{1, 2\}$. Suppose that f_r and f_w both address cell c_1 when they see the view 1 and c_2 when seeing 2. Finally, suppose that f_r and f_w are not similar: $\mathcal{X}(f_r) = \{x_1\}$ and $\mathcal{X}(f_w) = \{x_2\}$. The domain of inputs consistent with the views is $Expand\left(\mathcal{S}_{t-1}^{varying}\right) = \{11, 12, 21, 22\}$. The problem is that on the inputs 11 and 22, f_r and f_w address the same cell, while on the inputs 12 and 21 they access different cells. Hence, whether they access the same cell does not depend on simply one view.

4.4 Keeping Functions Varying during the Restriction Stage

Lemma 4.2 *The addressing functions in $\mathcal{F}_t$ are $b^{n^2} b_t$-varying on the set $\mathcal{S}_{t-1}^{similar}$.*

Proof of Lemma 4.2: Consider any addressing function $f \in \mathcal{F}_t$ and any variable $x_{j_l} \in \mathcal{X}(f)$. Recall that $Ind_{\mathcal{S}_{t-1}^{similar}}^{f, x_{j_l}}$ is the size of the largest subset of $\mathcal{S}_{t-1}^{similar}$ on which f is independent of x_{j_l}. Clearly, $Ind_{\mathcal{S}_{t-1}^{similar}}^{f, x_{j_l}} \leq Ind_{\mathcal{S}_{t-1}^{varying}}^{f, x_{j_l}}$, because $\mathcal{S}_{t-1}^{similar} \subseteq \mathcal{S}_{t-1}^{varying}$. Furthermore, $Ind_{\mathcal{S}_{t-1}^{varying}}^{f, x_{j_l}} \leq \frac{|\mathcal{S}_{t-1}^{varying}|}{(pt+1)^p b^{n^2} b_t}$ because f is $(pt+1)^p b^{n^2} b_t$-varying with respect to x_{j_l} on the set $\mathcal{S}_{t-1}^{varying}$. Finally, because $\frac{|\mathcal{S}_{t-1}^{varying}|}{(pt+1)^p} \leq |\mathcal{S}_{t-1}^{similar}|$, we can conclude that $Ind_{\mathcal{S}_{t-1}^{similar}}^{f, x_{j_l}} \leq \frac{|\mathcal{S}_{t-1}^{similar}|}{b^{n^2} b_t}$. Therefore, f is $b^{n^2} b_t$-varying with respect to x_{j_l} on the set $\mathcal{S}_{t-1}^{similar}$. ■

Lemma 4.3 *The addressing functions in $\mathcal{F}_{[1..t-1]}$ are $b^{n^2}b_t$-varying on the set $\mathcal{S}_{t-1}^{similar}$.*

Proof of Lemma 4.3: First recall that $|\mathcal{S}_{t-1}^{similar}| \geq \frac{|\mathcal{S}_{t-1}^{varying}|}{(pt+1)^p} \geq \frac{|\mathcal{S}_{t-1}|}{(pt+1)^p\left[(pt+1)^p b^{n^2} b_t\right]^{2p2^t}}$ $= \frac{b^{n^2}|\mathcal{S}_{t-1}|}{\left[(pt+1)^p b^{n^2}\right]^{2p2^t+1}[b_t]^{2p2^t}}$. Because generously $(pt+1)^p b^{n^2} \leq b^{(pn)^2} \leq b^{(pn)^{[T-t+2]}} = b_t$ and $t \in o(\log n)$, it follows that $|\mathcal{S}_{t-1}^{similar}| \geq \frac{b^{n^2}|\mathcal{S}_{t-1}|}{b_t^{4p2^t+1}} \geq \frac{b^{n^2}|\mathcal{S}_{t-1}|}{b_t^{pn-1}}$. Finally, $b_{t-1} = b_t^{pn}$ gives $|\mathcal{S}_{t-1}^{similar}| \geq \frac{b^{n^2}b_t|\mathcal{S}_{t-1}|}{b_{t-1}}$.

Consider any addressing function $f \in \mathcal{F}_{[1..t-1]}$ and any variable $x_{j_l} \in \mathcal{X}(f)$. Clearly, $Ind^{f,x_{j_l}}_{\mathcal{S}_{t-1}^{similar}} \leq Ind^{f,x_{j_l}}_{\mathcal{S}_{t-1}}$, because $\mathcal{S}_{t-1}^{similar} \subseteq \mathcal{S}_{t-1}$. Furthermore, $Ind^{f,x_{j_l}}_{\mathcal{S}_{t-1}} \leq \frac{|\mathcal{S}_{t-1}|}{b_{t-1}}$ because f is b_{t-1}-varying with respect to x_{j_l} on the set $\mathcal{S}_{t-1}$. From above, we have $|\mathcal{S}_{t-1}| \leq \frac{b_{t-1}}{b^{n^2}b_t}|\mathcal{S}_{t-1}^{similar}|$. It follows that $Ind^{f,x_{j_l}}_{\mathcal{S}_{t-1}^{similar}} \leq \frac{|\mathcal{S}_{t-1}^{similar}|}{b^{n^2}b_t}$. We can conclude that f is $b^{n^2}b_t$-varying with respect to x_{j_l} on the set $\mathcal{S}_{t-1}^{similar}$. ■

4.5 The $\mathcal{S}$ Expanding Stage

The subproblems $\Pi_{t-1}^1, \ldots, \Pi_{t-1}^{q_{t-1}}$ are refined as was done in Theorem 1, Section 2.2. Then, the set of views $\mathcal{S}_{t-1}^{similar}$ is expanded to form the larger set $\mathcal{S}_t$ of longer views. The new difficulty is how the expanding effects the varying property. To simplify the process, the subproblems are repartitioned one at at time, expanding $\mathcal{S}_{t-1}^{similar}$ each time.

As done in Section 2.4, consider expanding a set $\mathcal{S}^{pre} \subseteq [1..d]^q$ of views into the set $\mathcal{S}^{exp} \subseteq [1..d]^{q+\delta-1}$ while repartitioning the subproblem Π_{t-1}^i into δ new subproblems. For each setting $\langle \vec{u}, \vec{u}' \rangle \in [1..d]^{q-1}$, the subset of longer views $\vec{u} \times \left(\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}\right) \times \vec{u}' \subseteq \mathcal{S}^{pre}$ is expanded into the subset of views $\vec{u} \times \left(\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}\right)^{\delta} \times \vec{u}' \subseteq \mathcal{S}^{exp}$. If for some of the settings of $\langle \vec{u}, \vec{u}' \rangle$, the set of values $\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}$ is much larger than for the other settings, than the corresponding set of views would expand into far more views than for the other settings. In such a case, I will say that the set of views $\mathcal{S}^{pre}$ expands unevenly.

The problem with $\mathcal{S}^{pre}$ expanding unevenly is the following. Suppose that the functions in $\mathcal{F}_{[1..t]}$ are b'-varying on the views $\mathcal{S}^{pre}$. As this set of views expands, the cells addressed by a function f do not change and the subsets of $\mathcal{S}^{pre}$ on which f is independent of a variable x_{j_l} remain intact. However, if the set does not expand evenly, then one of these subsets $Ind^{f,x_{j_l}}_{\mathcal{S}^{pre}}$ may expand too much in proportion to the rest. In this case, f may no longer be b'-varying.

The solution to this problem is to first find a subset $\mathcal{S}^{bal} \subseteq \mathcal{S}^{pre}$ of the views such that each nonempty $\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}$ has the same size, i.e. $\exists z,\ \forall \langle \vec{u}, \vec{u}' \rangle,\ |\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}| \in \{0, z\}$. It follows from the next lemma that a set of size $|\mathcal{S}^{bal}| \geq \frac{|\mathcal{S}^{pre}|}{b^n}$ exists with this property. If the addressing functions are $b^n b'$-varying on $\mathcal{S}^{pre}$ then they will be at least b'-varying on the set $\mathcal{S}^{bal}$. This new set, $\mathcal{S}^{bal}$, is then expanded exactly as was done in Theorem 1 to form $\mathcal{S}^{exp}$. Because it expands evenly, the functions are still b'-varying on $\mathcal{S}^{exp}$.

4.6 Forming the Balanced Set $\mathcal{S}^{bal}$

Lemma 4.4 *Suppose that $d_u \in [1..d]$, for each $u \in [1..d^{q-1}]$. Let $A = \sum_u d_u$ and suppose that $A \geq \frac{d^q}{B}$. Then there exists values n' and d' such that there are at least n' indexes u for which $d_u \geq d'$ and for which $n'd' = \frac{A}{2\ln(B)}$.*

Proof of Lemma 4.4: Without loss of generality, assume that the d_u are sorted in decreasing order. Suppose that for all $u \in [1..d^{q-1}]$, $d_u < \frac{A}{2\ln B}\frac{1}{u}$. Then

$$\begin{aligned}
\sum_u d_u &< \sum_{u\in[1..\lfloor\frac{A}{2d\ln B}\rfloor]} d + \sum_{u\in[\lfloor\frac{A}{2d\ln B}\rfloor+1..d^{q-1}]} \frac{A}{2\ln B}\frac{1}{u} \\
&\leq \frac{A}{2\ln B} + \frac{A}{2\ln B}\int_{\frac{A}{2d\ln B}}^{d^{q-1}} \frac{1}{u}\,\delta u \\
&= \frac{A}{2\ln B}\left[1 + \ln\left(d^{q-1}\right) - \ln\left(\frac{A}{2d\ln B}\right)\right] \\
&\leq \frac{A}{2\ln B}\left[1 + \ln\left(d^{q-1}\right) - \ln\left(\frac{d^q}{B}\right) + \ln\left(2\ln B\right) + \ln\left(d\right)\right] \\
&= \frac{A}{2\ln B}\left[1 + \ln\left(B\right) + \ln\left(2\ln B\right)\right] < A.
\end{aligned}$$

This contradicts the fact that $\sum_u d_u = A$. It follows that there exists an index $u' \in [1..d^{q-1}]$ for which $d_{u'} \geq \frac{A}{2\ln B}\frac{1}{u'}$ and that for all $u \in [1\ldots u']$, d_u is at least this size. Then, $n' = u'$ and $d' = \frac{A}{2\ln B}\frac{1}{u'}$ meet the requirements. ■

Using this lemma, we are now able to define the subset $\mathcal{S}^{bal} \subseteq \mathcal{S}^{pre}$. For $u \in [1..d^{q-1}]$, let $d_u = \left|\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}\right|$, where $\langle\vec{u},\vec{u}'\rangle$ is the u^{th} vector in $[1..d]^{q-1}$. Recall that $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle} = \{v \in [1..d] \mid \langle\vec{u}, v, \vec{u}'\rangle \in \mathcal{S}^{pre}\}$. Therefore, $d_u \leq d$ and $\sum_u d_u = |\mathcal{S}^{pre}| \geq \frac{d^q}{B}$ for some B. Applying Lemma 4.4 gives the stated values n' and d'. For each $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ that is no smaller than d', let $\mathcal{V}'_{\langle\vec{u},\vec{u}'\rangle}$ be an arbitrary subset of $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ of size d'. For those $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ which are smaller than d', let $\mathcal{V}'_{\langle\vec{u},\vec{u}'\rangle} = \emptyset$. Finally, let $\mathcal{S}^{bal} = \bigcup_{\langle\vec{u},\vec{u}'\rangle} \vec{u} \times \mathcal{V}'_{\langle\vec{u},\vec{u}'\rangle} \times \vec{u}'$. Note that $|\mathcal{S}^{bal}| = n'd' = \frac{|\mathcal{S}^{pre}|}{2\ln B}$.

4.7 The Size of $\mathcal{S}_t$

Recall that $\mathcal{S}^{similar}_{t-1} \subseteq \mathcal{S}^{varying}_{t-1} \subseteq \mathcal{S}_{t-1}$ is found and Lemma 4.3 shows that $|\mathcal{S}^{similar}_{t-1}| \geq \frac{b^{n^2} b_t |\mathcal{S}_{t-1}|}{b_{t-1}}$ $\geq \frac{|\mathcal{S}_{t-1}|}{b_{t-1}}$. Plugging in $b_t = b^{(pn)^{[T-t+2]}}$ for $t \geq 1$, gives $|\mathcal{S}^{similar}_{t-1}| \geq \frac{|\mathcal{S}_{t-1}|}{b^{(pn)^{2T}-n^2}}$. By condition (4), $|\mathcal{S}_{t-1}| \geq \frac{d^{q(t-1)}}{b^{(pn)^{2T}\left(2^{q(t-1)}-2\right)}}$. Therefore, $|\mathcal{S}^{similar}_{t-1}| \geq \frac{d^{q(t-1)}}{b^{(pn)^{2T}\left(2^{q(t-1)}-1\right)-n^2}}$.

This set of views is expanded as the subproblems $\Pi^1_{t-1}, \ldots, \Pi^{q_{t-1}}_{t-1}$ are refined one at a time. Suppose that $\mathcal{S}^{pre}$ is the set of views before one of the subproblems is refined and that $|\mathcal{S}^{pre}| \geq \frac{d^q}{b^e}$. Lemma 4.4 gives that $|\mathcal{S}^{bal}| = \frac{|\mathcal{S}^{pre}|}{2\ln B} \geq \frac{d^q}{b^e\ 2\ln B}$ and Lemma 1 gives that $|\mathcal{S}^{exp}| \geq \frac{d^{q+\delta-1}}{(b^e\ 2\ln B)^\delta}$.

The first step is to bound B for each such refinement. When the first subproblem is expanded, $\mathcal{S}^{pre}$ is the set $\mathcal{S}^{similar}_{t-1}$. Hence, B is such that $|\mathcal{S}^{pre}| = |\mathcal{S}^{similar}_{t-1}| = \frac{d^{q(t-1)}}{B}$. Hence, $2\ln B = 2\ln\left(b^{(pn)^{2T}\left(2^{q(t-1)}-1\right)-n^2}\right)$. Because $b = 2(npT)^2$, $T \leq \log n$, and $q \leq \frac{n}{2}$, it follows that $2\ln B \leq b^n$. When refining the remaining subproblems $\Pi^2_{t-1}, \ldots, \Pi^{q_{t-1}}_{t-1}$, B is larger. However, it is easy to see that for each refinement the bound $2\ln B \leq b^n$ holds.

Therefore, if $|\mathcal{S}^{pre}| \geq \frac{d^q}{b^e}$, then $|\mathcal{S}^{bal}| \geq \frac{|\mathcal{S}^{pre}|}{b^n} \geq \frac{d^q}{b^{e+n}}$ and $|\mathcal{S}^{exp}| \geq \frac{d^{q+\delta-1}}{b^{[e+n]\delta}}$. Applying this for each of the $q_{(t-1)}$ subproblems proves that if the pre-expanded set $\mathcal{S}_{t-1}^{similar}$ is no smaller than $\frac{d^{q_{(t-1)}}}{b^e}$, then the expanded set $\mathcal{S}_t$ is no smaller than $\frac{d^{q_t}}{b^{[\ldots[[e+n]\delta_1+n]\delta_2\cdots+n]\delta_{q_{(t-1)}}}} \geq \frac{d^{q_t}}{b^{[e+nq_{(t-1)}]\delta_1\delta_2\cdots\delta_{q_{(t-1)}}}}$. It follows that $|\mathcal{S}_t| \geq \frac{d^{q_t}}{b^{\left\lceil (pn)^{2T}\left(2^{q_{(t-1)}}-1\right)-n^2+nq_{(t-1)}\right\rceil\delta_1\delta_2\cdots\delta_{q_{(t-1)}}}} \geq \frac{d^{q_t}}{b^{(pn)^{2T}\left(2^{q_{(t-1)}}-1\right)\delta_1\delta_2\cdots\delta_{q_{(t-1)}}}}$. By the two claims used in Lemma 2, this size is no smaller than $\frac{d^{q_t}}{b^{(pn)^{2T}\left(2^{q_t}-2\right)}}$, meeting the bound for condition (4).

4.8 Keeping Functions Varying during the Expanding Stage

By Lemmas 4.2 and 4.3, the addressing functions in $\mathcal{F}_{[1..t]}$ are $b^{n^2}b_t$-varying on the input domain $\mathcal{S}_{t-1}^{similar}$. In Section 4.5, $\mathcal{S}_{t-1}^{similar}$ is expanded into $\mathcal{S}_t$. For condition (5), we require that the addressing functions in $\mathcal{F}_{[1..t]}$ be b_t-varying on $\mathcal{S}_t$. The difficulty is that the views in $\mathcal{S}_t$ are longer tuples of values than those in $\mathcal{S}_{t-1}^{similar} \subseteq \mathcal{S}_{t-1}$. Hence, for $\mathcal{S}_t$, the addressing functions need to be considered to be functions on a larger list of variables. The additional variables are seen by the processor at the end of time step t, but not at the end of time step $t-1$. Hence, the functions in $\mathcal{F}_{[1..t]}$ will not actually depend on these extra variables. In fact, each function f will still be varying with respect to each of the variables in its fixed set $\mathcal{X}(f)$ and independent of the other variables.

To be more precise, recall that at the end of time step $t-1$, processor P sees the variables $\mathcal{V}_{\langle P,t-1\rangle}$ consisting of one variable from each of the subproblems $\Pi_{t-1}^1, \ldots, \Pi_{t-1}^{q_{t-1}}$; the views in $\mathcal{S}_{t-1}^{similar}$ assign a value to each of these variables; it is on these values that the state of the processor, hence its addressing functions, depends. In Section 4.5, the subproblems $\Pi_{t-1}^1, \ldots, \Pi_{t-1}^{q_{t-1}}$ are refined one at a time. A view in the intermediate set $\mathcal{S}^{bal} \subseteq \mathcal{S}^{pre}$ assigns a value to each of the current subproblems. Each of the addressing functions in $\mathcal{F}_{[1..t]}$ is considered to be a function on one variable from each of these current subproblems.

Lemma 4.5 *If the addressing functions in $\mathcal{F}_{[1..t]}$ are b'-varying on $\mathcal{S}^{bal}$, then they are still b'-varying on the expanded set $\mathcal{S}^{exp}$.*

Proof of Lemma 4.5: Suppose that after some of the subproblems are refined, the current list of subproblems is $\Pi^1, \ldots, \Pi^q$. Suppose as well that the subproblem Π^i is currently being refined into the smaller subproblems $\Pi^{i,1}, \ldots, \Pi^{i,\delta}$. As this is done, $\mathcal{S}^{bal}$ is expanded into the larger set of longer views $\mathcal{S}^{exp}$. Consider any addressing function $f \in \mathcal{F}_{[1..t]}$. Let $\widehat{f}$ and $\widetilde{f}$ be the same addressing function as f except considered as a function on the domains $\mathcal{S}^{bal}$ and $\mathcal{S}^{exp}$. The views in $\mathcal{S}^{bal}$ assign a value to each of the subproblems $\Pi^1, \ldots, \Pi^q$ and in this way $\widehat{f}$ is a function on the variables $\{x_{j_1}, \ldots, x_{j_q}\}$, where $x_{j_l} \in \Pi^l$ for each $l \in [1..q]$. The views in $\mathcal{S}^{exp}$ assign a value to each of the subproblems $\Pi^1, \ldots, \Pi^{i-1}, \Pi^{i,1}, \ldots, \Pi^{i,\delta}, \Pi^{i+1}, \ldots, \Pi^q$ and in this way $\widetilde{f}$ is a function on the variables $\{x_{j_1}, \ldots, x_{j_{i-1}}, z_{(i,1)}, \ldots, z_{(i,k-1)}, x_{j_i}, z_{(i,k+1)}, \ldots, z_{(i,\delta)}, x_{j_{i+1}}, \ldots, x_{j_q}\}$, where k is such that $x_{j_i} \in \Pi^{i,k} \subseteq \Pi^i$ and $z_{(i,k')} \in \Pi^{i,k'}$ for the other $k' \in [1..\delta]$. Note that f does not depend on the $z_{(i,k')}$ variables. Hence, for every value of the z variables, $\widehat{f}(x_{j_1}, \ldots, x_{j_q}) = \widetilde{f}(x_{j_1}, \ldots, x_{j_{i-1}}, z_{(i,1)}, \ldots, z_{(i,k-1)}, x_{j_i}, z_{(i,k+1)}, \ldots, z_{(i,\delta)}, x_{j_{i+1}}, \ldots, x_{j_q})$.

Let $l \in [1..q]$. If $x_{j_l} \notin \mathcal{X}(f)$, then by the assumption of the lemma, $\widehat{f}$ is constant on $\mathcal{S}^{bal}$ with respect x_{j_l}. It is not hard to see that $\widetilde{f}$ is then also constant on $\mathcal{S}^{exp}$ with respect to x_{j_l}. As well, $z_{(i,l)} \notin \mathcal{X}(f)$ and $\widehat{f}$ is independent of this variable. Finally, suppose that $x_{j_l} \in \mathcal{X}(f)$. By the

assumption of the lemma, $\widehat{f}$ is b'-varying with respect to x_{j_l} on $\mathcal{S}^{bal}$. Our goal is to prove that $\widetilde{f}$ is b'-varying with respect to x_{j_l} on the expanded set $\mathcal{S}^{exp}$. There are two cases, namely $l = i$ and $l \neq i$. Because the proofs of these two cases are similar, we will only prove the second case.

As before, we use v's to denote the values assigned to the repartitioned subproblems and u's to denote the values assigned to the other subproblems. Specifically, let v_{j_i} denote the value of the expanded variable x_{j_i} on which f depends and $\vec{v}$ the values of $\langle\, z_{(i,1)}, \ldots, z_{(i,k-1)}, z_{(i,k+1)}, \ldots, z_{(i,\delta)} \,\rangle$. Let u_{j_l} denote the value of the variable x_{j_l} with respect to which $\widetilde{f}$ must be b'-varying and $\vec{u}$ the values of $\langle\, x_{j_1}, \ldots, x_{j_{l-1}}, x_{j_{l+1}}, \ldots, x_{j_{i-1}}, x_{j_{i+1}}, \ldots, x_{j_q} \,\rangle$. Using this notation, we can decompose $\mathcal{S}^{bal}$ and $\mathcal{S}^{exp}$ as done before.

Define $\mathcal{V}_{\langle u_{j_l}, \vec{u}\rangle} = \{v_{j_i} \mid \langle\, v_{j_i}, u_{j_l}, \vec{u} \,\rangle \in \mathcal{S}^{bal}\}$. Then $\mathcal{S}^{bal} = \bigcup_{u_{j_l},\vec{u}} \left\{ \langle\, v_{j_i}, u_{j_l}, \vec{u} \,\rangle \mid v_{j_i} \in \mathcal{V}_{\langle u_{j_l},\vec{u}\rangle} \right\}$ and $\mathcal{S}^{exp} = \bigcup_{u_{j_l},\vec{u}} \left\{ \langle\, v_{j_i}, \vec{v}, u_{j_l}, \vec{u} \,\rangle \mid v_{j_i} \in \mathcal{V}_{\langle u_{j_l},\vec{u}\rangle} \text{ and } \vec{v} \in \left(\mathcal{V}_{\langle u_{j_l},\vec{u}\rangle}\right)^{\delta-1} \right\}$. Because $\mathcal{S}^{bal}$ was formed using Lemma 4.4, we know that $|\mathcal{V}_{\langle u_{j_l},\vec{u}\rangle}| = d'$ for each $\langle u_{j_l}, \vec{u}\rangle$ and hence $|\mathcal{S}^{bal}| = \sum_{u_{j_l},\vec{u}} \left|\mathcal{V}_{\langle u_{j_l},\vec{u}\rangle}\right| = n'd'$ and $|\mathcal{S}^{exp}| = \sum_{u_{j_l},\vec{u}} \left|\mathcal{V}_{\langle u_{j_l},\vec{u}\rangle}\right|^{\delta} = n'd'^{\delta} = d'^{\delta-1}|\mathcal{S}^{bal}|$.

The next step of the proof is to prove that $Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}} \leq d'^{\delta-1} Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}}$, where $Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}}$ is the size of the largest subset of $\mathcal{S}^{bal}$ on which $\widehat{f}$ is independent of χ_{j_l} and $Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}}$ is the size of the largest subset of $\mathcal{S}^{exp}$ on which $\widetilde{f}$ is independent of χ_{j_l}.

Fix some subset of $\mathcal{S}^{exp}$ on which $\widetilde{f}$ is independent of x_{j_l} of the maximum size $Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}}$. Because $\widetilde{f}$ is independent of x_{j_l} on this subset, it is well defined to define the function $\widetilde{C}(v_{j_i}, \vec{v}, \vec{u}) = \widetilde{f}(v_{j_i}, \vec{v}, u_{j_l}, \vec{u})$ to be the cell addressed on this subset. Recall that $\widehat{f}(v_{j_i}, u_{j_l}, \vec{u}) = \widetilde{f}(v_{j_i}, \vec{v}, u_{j_l}, \vec{u})$. Hence, we can define $\widehat{C}(v_{j_i}, \vec{u}) = \widetilde{C}(v_{j_i}, \vec{v}, \vec{u})$. Consider the subset of $\mathcal{S}^{bal}$ on which $\widehat{f}$ accesses the cell specified by $\widehat{C}$. $\widehat{f}$ is independent of x_{j_l} on this subset. Hence, the size of this set is no more than $Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}}$. To prove $Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}} \leq d'^{\delta-1} Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}}$, what remains is to compare the sizes of these two subsets of views.

$$
\begin{aligned}
Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}} &= \left| \left\{ \langle v_{j_i}, \vec{v}, u_{j_l}, \vec{u}\rangle \in \mathcal{S}^{exp} \mid \widetilde{f}(v_{j_i}, \vec{v}, u_{j_l}, \vec{u}) = \widetilde{C}(v_{j_i}, \vec{v}, \vec{u}) \right\} \right| \\
&= \left| \left\{ \langle v_{j_i}, \vec{v}, u_{j_l}, \vec{u}\rangle \mid v_{j_i} \in \mathcal{V}_{\langle u_{j_l},\vec{u}\rangle},\ \vec{v} \in \left(\mathcal{V}_{\langle u_{j_l},\vec{u}\rangle}\right)^{\delta-1},\ \text{and } \widehat{f}(v_{j_i}, u_{j_l}, \vec{u}) = \widehat{C}(v_{j_i}, \vec{u}) \right\} \right| \\
&= \sum_{u_{j_l},\vec{u}} \left| \left\{ v_{j_i} \mid v_{j_i} \in \mathcal{V}_{\langle u_{j_l},\vec{u}\rangle} \text{ and } \widehat{f}(v_{j_i}, u_{j_l}, \vec{u}) = \widehat{C}(v_{j_i}, \vec{u}) \right\} \right| \left| \mathcal{V}_{\langle u_{j_l},\vec{u}\rangle} \right|^{\delta-1} \\
&= d'^{\delta-1} \sum_{u_{j_l},\vec{u}} \left| \left\{ v_{j_i} \mid \langle v_{j_i}, u_{j_l}, \vec{u}\rangle \in \mathcal{S}^{bal} \text{ and } \widehat{f}(v_{j_i}, u_{j_l}, \vec{u}) = \widehat{C}(v_{j_i}, \vec{u}) \right\} \right| \\
&= d'^{\delta-1} \left| \bigcup_{u_{j_l},\vec{u}} \left\{ \langle v_{j_i}, u_{j_l}, \vec{u}\rangle \in \mathcal{S}^{bal} \mid \widehat{f}(v_{j_i}, u_{j_l}, \vec{u}) = \widehat{C}(v_{j_i}, \vec{u}) \right\} \right| \leq d'^{\delta-1} Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}}
\end{aligned}
$$

We can now complete the proof. By the statement of the lemma, $\widehat{f}$ is b'-varying with respect to x_{j_l} on $\mathcal{S}^{bal}$. Therefore, by definition, $Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}} \leq \frac{|\mathcal{S}^{bal}|}{b'}$. If follows that $Ind^{\widetilde{f},x_{j_l}}_{\mathcal{S}^{exp}} \leq d'^{\delta-1} Ind^{\widehat{f},x_{j_l}}_{\mathcal{S}^{bal}} \leq$

$d'^{\delta-1}\frac{|\mathcal{S}^{bal}|}{b'} = \frac{|\mathcal{S}^{exp}|}{b'}$. This proves that $\widetilde{f}$ is b'-varying with respect to x_{j_l} on $\mathcal{S}^{exp}$. ■

We now are able to obtain condition (5).

Lemma 4.6 *The addressing functions in $\mathcal{F}_{[1..t]}$ are b_t-varying on the set $\mathcal{S}_t$.*

Proof of Lemma 4.6: By Lemmas 4.2 and 4.3, the addressing functions in $\mathcal{F}_{[1..t]}$ are $b^{n^2}b_t$-varying on the input domain $\mathcal{S}_{t-1}^{similar}$. Consider a functions $f \in \mathcal{F}_{[1..t]}$ and a variable $x_{j_l} \in \mathcal{X}(f)$. Suppose that before one of the subproblems $\Pi_{t-1}^1, \ldots, \Pi_{t-1}^{q_{t-1}}$ is refined, f is $b^n b'$-varying with respect to x_{j_l} on $\mathcal{S}^{pre}$. Hence, by definition, $Ind_{\mathcal{S}^{pre}}^{f,x_{j_l}} \leq \frac{|\mathcal{S}^{pre}|}{b^n b'}$. Lemma 4.4 found the balanced subset of views $\mathcal{S}^{bal} \subseteq \mathcal{S}^{pre}$ such that $\frac{|\mathcal{S}^{pre}|}{b^n} \leq \mathcal{S}^{bal}$. It follows that $Ind_{\mathcal{S}^{bal}}^{f,x_{j_l}} \leq Ind_{\mathcal{S}^{pre}}^{f,x_{j_l}} \leq \frac{|\mathcal{S}^{pre}|}{b^n b'} \leq \frac{|\mathcal{S}^{bal}|}{b'}$ and that f is b'-varying with respect to x_{j_l} on $\mathcal{S}^{bal}$. Lemma 4.5 then proves that f is b'-varying with respect to x_{j_l} on $\mathcal{S}^{exp}$. The conclusion is that the refining of one subproblem looses at most a factor of b^n in the amount the functions vary. Hence, refining all q_t subproblems looses at most a total factor of $(b^n)^{q_t} \leq b^{n^2}$. The addressing functions in $\mathcal{F}_{[1..t]}$ are $b^{n^2}b_t$-varying on the initial pre-expanded set $\mathcal{S}_{t-1}^{similar}$. Therefore, they are still b_t-varying on the final expanded set $\mathcal{S}_t$. ■

4.9 The final sets of inputs $\mathcal{D}_T^i$

The above steps complete all the induction hypothesis for time step t. This is repeated until time final time step T. Note, in order to satisfy condition (8), $q_T < n^{0.1}$, the computation needs to be stopped sooner than was done in Theorem 1. Since the time bound is logarithmic in n, this effects the time by only a constant factor. What remains is to form the sets of inputs $\mathcal{D}_T^i$ satisfying conditions (7-11).

As done in Section 2.6, the set of inputs is formed by refining the subproblems $\Pi_T^1, \ldots, \Pi_T^{q_T}$ into the n singleton sets and expanding the final set of views $\mathcal{S}_T$ to form the set of longer views. These longer views are in fact inputs, because they assign a value to each of the variables. However, as done in Section 4.5, the subproblems are refined one at a time. Each time a subproblem Π_T^j is refined, the set of views $\mathcal{S}^{pre}$ is restricted to a subset $\mathcal{S}^{bal}$ and then this balanced set of views is expanded to form $\mathcal{S}^{exp}$. Because of these balancing steps, the set of views\inputs obtained depends on the order in which the subproblems $\Pi_T^1, ..., \Pi_T^{q_T}$ are refined. For each of the subproblems Π_T^i, let $\mathcal{D}_T^i$ be the set of inputs formed from $\mathcal{S}_T$ by refining the subproblems in an order in which Π_T^i is refined last.

By condition (1), the PRAM algorithm is $(Expand(\mathcal{S}_T) - Bad_{[1..T]}, T, \Gamma_{[1..t]})$-oblivious. Because $\mathcal{D}_T^i \subseteq Expand(\mathcal{S}_T)$, it follows that the PRAM algorithm is $(\mathcal{D}_T^i - Bad_{[1..T]}, T, \Gamma_{[1..t]})$-oblivious. The calculations in Section 4.7 give that $|\mathcal{D}_T^i| \geq \frac{d^n}{b^{(pn)^{2^T}\ 2^n}}$.

By condition (5), the addressing functions $\mathcal{F}_{[1..T]}$ are b_T-varying on the domain $\mathcal{S}_T$, where $b_T = b^{(pn)^{[T-T+2]}} \geq 2^{n^2}b$. It follows by the same proof as in Lemma 4.6 that these functions are b-varying on the domain $\mathcal{D}_T^i$.

Because $\mathcal{D}_T^i$ is the set inputs formed when the subproblem Π_T^i has been repartitioned last, it follows that the sets $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}^i$ associated with variables $x_{j_i} \in \Pi_T^i$ are balanced before this last repartitioning. It follows that each nonempty $\mathcal{V}_{\langle\vec{u},\vec{u}'\rangle}$ has the same fixed size.

This completes all of the induction hypothesis. ■

5 Handling the Unrelated Pairs

Lemma 5 *Given the above constructs for time T, there exists, for each subproblem Π_T^i, $i \in [1..q_T]$, an element distinct input Φ^i and there exists, for each pair of variables $\{x_\alpha, x_\beta\}$ contained in the subproblem Π_T^i, a non-element distinct input $\Phi^i_{x_\alpha=x_\beta=v_i}$ such that: $\Phi^i_{x_\alpha=x_\beta=v_i}$ assigns the value v_i to the variables x_α and x_β and is the same as Φ^i on every other variable; the processors interact according to $\Gamma_{[1..t]}$ on the input Φ^i; and if $\Phi^i_{x_\alpha=x_\beta=v_i} \notin Bad^{\{x_\alpha,x_\beta\}-covering}_{[1..T]}$, then the same is true on this input.*

Proof of Lemma 5: For each final subproblem Π_T^i, $i \in [1..q_T]$, consider the set of inputs $\mathcal{D}_T^i = \bigcup_{<\vec{u},\vec{u'}>} \vec{u} \times \left(\mathcal{V}^i_{\langle\vec{u},\vec{u'}\rangle}\right)^{\delta_i} \times \vec{u'}$, constructed in Lemma 4. Randomly, choose the input Φ^i uniformly from this set. Let $\vec{u} \in [1..d]^\delta$ be the values assigned by Φ^i to the variables contained in $\cup_{j\in[1..i-1]}\Pi_T^j$ and let $\vec{u'} \in [1..d]^{\delta'}$ be the values assigned to those in $\cup_{j\in[i+1..q_T]}\Pi_T^j$. $\mathcal{V}^i_{\langle\vec{u},\vec{u'}\rangle}$ is the set values v such that $\vec{u} \times \left(\mathcal{V}^i_{\langle\vec{u},\vec{u'}\rangle}\right)^{\delta_i} \times \vec{u'}$ is a subset of the input domain $\mathcal{D}_T^i$. Randomly choose the value v_i from this set $\mathcal{V}^i_{\langle\vec{u},\vec{u'}\rangle}$. For each pair of variables $\{x_\alpha, x_\beta\}$ in the subproblem Π_T^i, let $\Phi^i_{x_\alpha=x_\beta=v_i}$ be the input that is identical to Φ^i, except for the variables x_α and x_β which have the value v_i. Clearly, the inputs $\{\ \Phi^i_{x_\alpha=x_\beta=v_i} \mid x_\alpha, x_\beta \in \Pi_T^i\ \}$ are all contained within the domain $\mathcal{D}_T^i$.

For each of the requirements of Lemma 5, a sub-lemma below proves that it is with small probability that the inputs chosen do not meet the requirement. Summing these probabilities, we get that the total probability is strictly less than 1 that one of these bad properties occurs. Therefore, there exists a choice for Φ^i and v_i for which none of these things happen. Fix such a choice for each subproblem. ■

Lemma 5.1 *The probability that the input Φ^i is not element distinct is very small.*

Proof of Lemma 5.1: The size $|\mathcal{D}_T^i| \geq \frac{d^n}{b^{(pn)^{2T}\ 2^n}}$ given in condition (9) is much larger than the number $\binom{n}{2}d^{n-1}$ of non-element distinct inputs when $d \geq 2^{2^{(1+\epsilon)n}}$. ■

Lemma 5.2 *The probability is no more than $\frac{(pT)^2}{b}$ that $\Phi^i \in Bad_{[1..T]}$. Because $b = 2(npT)^2$ this probability is very small.*

Proof of Lemma 5.2: Consider any read-write unrelated or covering pair f_r and f_w. Because there are at most $(pT)^2$ such read-write pairs, it is sufficient to prove that this pair accesses the same cell on no more than $\frac{|\mathcal{D}_T^i|}{b}$ of the inputs in $\mathcal{D}_T^i$. Because $\mathcal{X}(f_r) \neq \mathcal{X}(f_w)$, there must exist some variable x_k on which only one of the functions depends. Without loss of generality, assume that f_r, but not f_w, depends on the variable x_k. The cell accessed by f_w is independent of x_k. Therefore, the set of bad inputs on which f_r accesses the same cell as f_w forms a subdomain of $\mathcal{D}_T^i$ on which the function f_r is independent of x_k. Because f_r is b-varying with respect to x_k on the domain $\mathcal{D}_T^i$, this subdomain is no larger than $\frac{|\mathcal{D}_T^i|}{b}$. ■

Lemma 5.2 proved that the addressing functions f_r and f_w do not access the same cell on most inputs in $\mathcal{D}_T^i$. The goal of the next lemma is to prove that the same is true for most non-element distinct inputs. More precisely, consider some pair of variables $\{x_\alpha, x_\beta\} \in \Pi_T^i$. Lemma 5.3 proves that if f_r and f_w are not a $\{x_\alpha, x_\beta\}$-covering pair, then they do not access the same cell on

most inputs in $\mathcal{D}_T^i$ for which $x_\alpha = x_\beta$. Note that this lemma might not be true if f_r and f_w are a $\{x_\alpha, x_\beta\}$-covering pair, i.e. $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \{x_\alpha\}$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \{x_\beta\}$. For example, if f_r addresses memory with the value of x_α using a 1-1 mapping and f_w uses the same mapping except with the value of x_β, then they access the same cell if and only if $x_\alpha = x_\beta$.

Lemma 5.3 *If Φ^i and v_i are chosen at random as described, then the probability is no more than $\frac{(npT)^2}{b}$ that there exists a pair of variables $x_\alpha, x_\beta \in \Pi_T^i$ such that the input $\Phi^i_{x_\alpha = x_\beta = v_i}$ is contained in $Bad^{unrelated}_{[1..T]} \cup \bigcup_{\forall x'_\alpha, x'_\beta \in \Pi_t^i, \{x'_\alpha, x'_\beta\} \neq \{x_\alpha, x_\beta\}} Bad^{\{x_\alpha, x_\beta\}-covering}_{[1..T]}$. Because $b = 2(npT)^2$ this probability is less than $\frac{1}{2}$.*

Proof of Lemma 5.3: Consider any pair of variables $x_\alpha, x_\beta \in \Pi_T^i$ and any read-write unrelated or covering pair f_r and f_w which is not a $\{x_\alpha, x_\beta\}$-covering pair. Because there are at most n^2 variable pairs and $(pT)^2$ read-write pairs, it is sufficient to prove that f_r and f_w access the same cell on the input $\Phi^i_{x_\alpha = x_\beta = v_i}$ for no more than a $\frac{1}{b}$ fraction of the choices for Φ^i and v_i.

The pair f_r and f_w are not similar, (i.e. $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \emptyset$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \emptyset$) and are not $\{x_\alpha, x_\beta\}$-Covering Pairs, (i.e. $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \{x_\alpha\}$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \{x_\beta\}$). Because x_α and x_β are contained in the same subproblem, it is neither the case that both x_α and x_β are contained in $\mathcal{X}(f_r)$ nor both in $\mathcal{X}(f_w)$. Therefore, there are two remaining cases.

Case 1 There exists another variable x_k (not the same variable as x_α or x_β) on which one, but not both, of the two functions depends, (i.e. x_k is contained in either $\mathcal{X}(f_r) - \mathcal{X}(f_w)$ or in $\mathcal{X}(f_w) - \mathcal{X}(f_r)$)

Case 2 One of the functions f_r or f_w depends on one of the variables x_α (or x_β), however, the other function depends on neither of them.

For case 1, assume without loss of generality that f_r, but not f_w, depends on x_k. As well, f_r cannot depend on both x_α and x_β. Without loss of generality, assume that it does not depend on x_β. If f_r tries to mimic f_w using the value of x_α instead of x_β so that they access the same cell if and only if $x_\alpha = x_\beta$, then adjusting the value of x_k will change the cell addressed by f_r but not by f_w. Because f_r is b-varying with respect to x_k, f_w will manage to mimic f_w on no more than a $\frac{1}{b}$ fraction of the inputs.

Case 1 will now be broken into two sub-case depending on whether x_k is contained in the subproblem Π_T^i. Besides differences in notation, the case 1.1 and 1.2 differ very little. However, to be formal they are both included.

Case 1.1 $x_k \in \Pi_T^i$:

Recall $\vec{u} \in [1..d]^\delta$ denotes the values assigned by Φ^i to the variables contained in $\cup_{j \in [1..i-1]} \Pi_T^j$ and $\vec{u}' \in [1..d]^{\delta'}$ denotes the values assigned to those in $\cup_{j \in [i+1..q_T]} \Pi_T^j$. Let v_α, v_β, and v_k denote the values assigned by Φ^i to the variables $x_\alpha, x_\beta, x_k \in \Pi_T^i$ and finally let $\vec{v}$ denote the values assigned to the remaining variables in Π_t^i. The probability space that we are considering is a random choice for $\Phi^i = \langle \vec{u}, v_\alpha, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i$ and a random choice for $v_i \in \mathcal{V}^i_{\langle \vec{u}, \vec{u}' \rangle}$. The size of this sample space is

$$\left| \left\{ \langle \vec{u}, v_\alpha, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u}, \vec{u}' \rangle} \right\} \right| = \sum_{\langle \vec{u}, \vec{u}' \rangle} \left| \left\{ \langle \vec{u}, v_\alpha, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i \right\} \right| \times \left| \mathcal{V}^i_{\langle \vec{u}, \vec{u}' \rangle} \right|.$$

By condition (11), each nonempty $\mathcal{V}_{\langle \vec{u},\vec{u}' \rangle}$ has the same fixed size d'. Hence, this amount can be factored out, giving that the size of the sample space is $d'|\mathcal{D}_T^i|$.

The set of bad $\langle \Phi^i, v_i \rangle$ samples for which f_r and f_w access the same cell on the input $\Phi^i_{x_\alpha = x_\beta = v_i}$ is

$$\left\{ \langle \vec{u}, v_\alpha, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle} \mid f_r(\vec{u}, v_i, v_i, v_k, \vec{v}, \vec{u}') = f_w(\vec{u}, v_i, v_i, v_k, \vec{v}, \vec{u}') \right\}.$$

We will transform this set into a subdomain of $\mathcal{D}_T^i$ on which the function f_r is independent of the value v_k of the variable x_k. Two difficult changes are required to complete the transformation: the input to which f_r and f_w are applied much be transformed from a non-element distinct input to a general input from $\mathcal{D}_T^i$ and because $n+1$ different values are being considered, a d must be factored out.

Because f_r does not depend on x_β, it follows that f_r will access the same cell whether x_β is set to have the same value v_i which x_α is set to or to a different value v_β. More precisely, $f_r(\vec{u}, v_i, v_i, v_k, \vec{v}, \vec{u}') = f_r(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}')$. We cannot do the same trick for f_w, because it might depend on x_β. However, we do not really need to consider the function f_w itself. Instead, define C' to be a function, such that, for every value $v_\beta \in [1..d]$, $C'(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}') = f_w(\vec{u}, v_i, v_i, v_k, \vec{v}, \vec{u}')$. Note that even if f_w depends on the value of x_β, the function C' does not. This can be carried a step further by noting that C' is independent of the value v_k of x_k because f_w is. Therefore, define C to be such that, for every value $v_k \in [1..d]$, $C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}') = C'(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}')$. This gives that the number of bad $\langle \Phi^i, v_i \rangle$ samples is

$$\left| \left\{ \langle \vec{u}, v_\alpha, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle} \mid f_r(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}') \right\} \right|$$

It remains to remove the extra factor of d. It is interesting that the value v_α is factored out, instead of the v_i as might have been expected.

$$\begin{aligned}
&= \sum_{\langle \vec{u},\vec{u}' \rangle} \left| \left\{ v_\alpha, v_\beta, v_k, v_i \in \mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle},\ \vec{v} \in \left(\mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle}\right)^{\delta_i - 3} \mid f_r(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}') \right\} \right| \\
&= |\mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle}| \sum_{\langle \vec{u},\vec{u}' \rangle} \left| \left\{ v_\beta, v_k, v_i \in \mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle},\ \vec{v} \in \left(\mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle}\right)^{\delta_i - 3} \mid f_r(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}') \right\} \right| \\
&= d' \left| \left\{ \langle \vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}' \rangle \in \mathcal{D}_T^i \mid f_r(\vec{u}, v_i, v_\beta, v_k, \vec{v}, \vec{u}') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}') \right\} \right|
\end{aligned}$$

This last set is a subdomain of $\mathcal{D}_T^i$ on which the function f_r is independent of the value v_k of the variable x_k. Because f_r is b-varying with respect to x_k, it follows that this subdomain is no larger than $\frac{|\mathcal{D}_T^i|}{b}$. We can conclude that the number of such bad $\langle \Phi^i, v_i \rangle$ samples is no more than $d'\frac{|\mathcal{D}_T^i|}{b}$, which is no more than a $\frac{1}{b}$ fraction of the sample space.

Case 1.2 $x_k \notin \Pi_T^i$:

With out loss of generality, assume that $x_k \in \cup_{j \in [i+1..q_T]} \Pi_T^j$ and let u_k denote the value assigned by Φ^i to x_k and let $\vec{u}'' \in [1..d]^{\delta'-1}$ denotes the values assigned to the remaining variables in $\cup_{j \in [i+1..q_T]} \Pi_T^j$. Using this notation, the size of this sample space is

$$\left| \left\{ \langle \vec{u}, v_\alpha, v_\beta, \vec{v}, u_k, \vec{u}'' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u},\vec{u}' \rangle} \right\} \right| \ = \ d'|\mathcal{D}_T^i|.$$

The size of the set of bad $\langle \Phi^i, v_i \rangle$ samples for which f_r and f_w access the same cell on the input $\Phi^i_{x_\alpha = x_\beta = v_i}$ is

$$\left|\left\{ \langle \vec{u}, v_\alpha, v_\beta, \vec{v}, u_k, \vec{u}'' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle} \mid f_r(\vec{u}, v_i, v_i, \vec{v}, u_k, \vec{u}'') = f_w(\vec{u}, v_i, v_i, \vec{v}, u_k, \vec{u}'') \right\}\right|$$

$$= \left|\left\{ \langle \vec{u}, v_\alpha, v_\beta, \vec{v}, u_k, \vec{u}'' \rangle \in \mathcal{D}_T^i,\ v_i \in \mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle} \mid f_r(\vec{u}, v_i, v_\beta, \vec{v}, u_k, \vec{u}'') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}'') \right\}\right|$$

$$= \sum_{\langle \vec{u}, u_k, \vec{u}'' \rangle} \left|\left\{ v_\alpha, v_\beta, v_i \in \mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle},\ \vec{v} \in \left(\mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle}\right)^{\delta_i - 2} \mid f_r(\vec{u}, v_i, v_\beta, \vec{v}, u_k, \vec{u}'') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}'') \right\}\right|$$

$$= |\mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle}| \sum_{\langle \vec{u}, u_k, \vec{u}'' \rangle} \left|\left\{ v_\beta, v_i \in \mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle},\ \vec{v} \in \left(\mathcal{V}^i_{\langle \vec{u}, u_k, \vec{u}'' \rangle}\right)^{\delta_i - 2} \mid f_r(\vec{u}, v_i, v_\beta, \vec{v}, u_k, \vec{u}'') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}'') \right\}\right|$$

$$= d' \left|\left\{ \langle \vec{u}, v_i, v_\beta, \vec{v}, u_k, \vec{u}'' \rangle \in \mathcal{D}_T^i \mid f_r(\vec{u}, v_i, v_\beta, \vec{v}, u_k, \vec{u}'') = C(\vec{u}, v_i, v_\beta, \vec{v}, \vec{u}'') \right\}\right|$$

This completes case 1. The proof of case 2 is very similar. Because neither function depends on x_β, its value can be adjusted to either be equal to x_α or not. ■

6 The Element Distinctness Graph

What remains is to determine for which pairs of variables $\{x_\alpha, x_\beta\}$ the processors can differentiate between Φ^i and $\Phi^i_{x_\alpha = x_\beta = v_i}$ by knowing how the $\{x_\alpha, x_\beta\}$-covering pairs interact. This is done by considers the "element distinctness" graph G on vertex set $\{x_1, \ldots, x_n\}$ and by covering the edge $\{x_\alpha, x_\beta\}$ if the corresponding inputs can be differentiated. Lemma 6 uses graph theoretic constructs to characterize those edges covered.

Tuples and systems of tuples are defined in [RSSW88] to cover the edges $\{x_\alpha, x_\beta\}$ of the element distinctness graph for which the PRAM has learned that $x_\alpha \neq x_\beta$. A tuple is a sequence of variables $\langle\, x_{j_1}, \ldots, x_{j_{q_T}} \,\rangle$, one for each subproblem $\langle\, \Pi_T^1, \ldots, \Pi_T^{q_T} \,\rangle$, (i.e. a vantage point). Unlike in [RSSW88], in the present paper, the PRAM may have gained partial information about whether $x_\alpha = x_\beta$. Therefore, the concept of a tuple needs to be extended. A **labeled tuple** is the same as before, except now each variable is labeled with a memory cell $c \in \mathbb{R}$. For each addressing function $f \in \mathcal{F}_{[1..T]}$, the adversary forms the labeled tuple $\mathcal{T}_f$ as follows. Let $\mathcal{V} = \langle\, x_{j_1}, \ldots, x_{j_{q_T}} \,\rangle$ be the vantage point of the processor using f. For each $i \in [1..q_T]$, let $\Phi^i_{x_{j_i} = v_i}$ be the same input as Φ^i except that $x_{j_i} = v_i$. Define c_i to be the cell $f\left(\Phi^i_{x_{j_i} = v_i}\right)$ addressed by f on this input. The labeled tuple associated with the addressing function f is $\mathcal{T}_f = \langle\, (x_{j_1}, c_1), (x_{j_2}, c_2), \ldots, (x_{j_{q_T}}, c_{q_T}) \,\rangle$.

The edges of the element distinctness graph are covered by pairs of labeled tuples. We say that the pair of labeled tuples $(\mathcal{T}_{f_r}, \mathcal{T}_{f_w})$ covers the edge $\{x_\alpha, x_\beta\}$ of G, if there exists a coordinate $i \in [1..q_T]$ such that the variable in the i^{th} coordinate of $\mathcal{T}_{f_r}$ is x_α and of $\mathcal{T}_{f_w}$ is x_β and for each of the other coordinates the variable in $\mathcal{T}_{f_r}$ and in $\mathcal{T}_{f_w}$ are the same. In addition, the label of x_α in $\mathcal{T}_{f_r}$ must be the same as the the label of x_β in $\mathcal{T}_{f_w}$.

For example, $\mathcal{T}_{f_r} = \langle (x_1, c_8), (x_2, c_4), (x_\alpha, c), (x_4, c_9) \rangle$ and
$\mathcal{T}_{f_w} = \langle (x_1, c_4), (x_2, c_3), (x_\beta, c), (x_4, c_8) \rangle$ cover the edge $\{x_\alpha, x_\beta\}$.

Lemma 6.1 *Suppose that f_r and f_w form a $\{x_\alpha, x_\beta\}$-Covering Pair of addressing functions. If these functions access the same cell c on the input $\Phi^i_{x_\alpha = x_\beta = v_i}$, then the pair of labeled tuples $(\mathcal{T}_{f_r}, \mathcal{T}_{f_w})$ covers the edge $\{x_\alpha, x_\beta\}$.*

Proof of Lemma 6.1: Because $\mathcal{X}(f_r) - \mathcal{X}(f_w) = \{x_\alpha\}$ and $\mathcal{X}(f_w) - \mathcal{X}(f_r) = \{x_\beta\}$, the variables in the two labeled tuples will be the same, except for x_α and x_β. Because these two variables are in the same subproblem, the order of the variables in the two tuples will be the same. In addition, because $x_\beta \notin \mathcal{X}(f_r)$, $f_r(\Phi^i_{x_\alpha=v_i}) = f_r(\Phi^i_{x_\alpha=x_\beta=v_i}) = c$. Similarly, $f_w(\Phi^i_{x_\beta=v_i}) = f_w(\Phi^i_{x_\alpha=x_\beta=v_i}) = c$. Therefore, the label of x_α in $\mathcal{T}_{f_r}$ and of x_β in $\mathcal{T}_{f_w}$ are both the same cell c. ■

The most obvious way to proceed is to cover the edge $\{x_\alpha, x_\beta\}$ of the element distinctness graph G if there exists addressing functions $f_r \in \mathcal{F}^{read}_{[1..T]}$ and $f_w \in \mathcal{F}^{write}_{[1..T]}$ such that the associated pair of labeled tuples $(\mathcal{T}_{f_r}, \mathcal{T}_{f_w})$ covers the edge. The problem with this technique, however, is that if this were done, every edge of G could be covered in one time step on both the PRIORITY and on the COMMON model. This is in fact is the essence of the constant time PRIORITY algorithm. The solution, as stated before, is that the adversary does allow the $\{x_\alpha, x_\beta\}$-covering pairs of addressing functions to access the same cell as long as she can ensure that they do not interact.

A **labeled tuple system** $(\mathcal{A}, \mathcal{B})$ is a pair of sets of labeled tuples. It is important for the graph theoretic result that the sets $\mathcal{A}$ and $\mathcal{B}$ are disjoint. The system is said to cover the edge $\{x_\alpha, x_\beta\}$ if there exists labeled tuples $\mathcal{T}_\mathcal{A} \in \mathcal{A}$ and $\mathcal{T}_\mathcal{B} \in \mathcal{B}$ such that the pair $(\mathcal{T}_\mathcal{A}, \mathcal{T}_\mathcal{B})$ covers the edge. The adversary forms a labeled tuple system for each time step $t_w \in [1..T]$. Let $\mathcal{A}_{[t_w..T]} = \{\mathcal{T}_{f_r} \mid f_r \in \mathcal{F}^{read}_{[t_w..T]}\}$ and $\mathcal{B}_{t_w} = \{\mathcal{T}_{f_w} \mid f_w \in \mathcal{F}^{write}_{t_w}\}$. Note that $\sum_i |\mathcal{A}_i| \leq pT^2$ and $\sum_i |\mathcal{B}_i| \leq pT$. The sets $\mathcal{A}_{[t_w..T]}$ and $\mathcal{B}_{t_w}$ might not be disjoint. Therefore, the ${t_w}^{th}$ labeled tuple system is defined to be $\left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w}\right)$. We are now ready to prove the main lemma of this section.

Lemma 6 *If* $\Phi^i_{x_\alpha=x_\beta=v_i} \in Bad^{\{x_\alpha,x_\beta\}-covering}_{[1..T]}$, *then the collection of labeled tuple systems* $\left\{\left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w}\right) \mid t_w \in [1..T]\right\}$ *covers the edge* $\{x_\alpha, x_\beta\}$. *Note as well that* $\sum_i |\mathcal{A}_i| \leq pT^2$ *and* $\sum_i |\mathcal{B}_i| \leq pT$ *and that the tuples have length* q_T.

Proof of Lemma 6: Suppose that $\Phi^i_{x_\alpha=x_\beta=v_i} \in Bad^{\{x_\alpha,x_\beta\}-\text{covering}}_{[1..T]}$. Then, by definition, there exists addressing functions $f_r \in \mathcal{F}^{read}_{[1..T]}$ and $f_w \in \mathcal{F}^{write}_{[1..T]}$ such that: $Type\,(f_r, f_w) = \{x_\alpha, x_\beta\}$-covering and $f_w \in \Gamma^{\text{alg}}_{[1..t]}\left(\Phi^i_{x_\alpha=x_\beta=v_i}, f_r\right)$. In addition, there are no write functions in $\Gamma^{\text{alg}}_{[1..t]}\left(\Phi^i_{x_\alpha=x_\beta=v_i}, f_r\right)$ that are similar to f_r. Recall that $\Gamma^{\text{alg}}_{[1..t]}\left(\Phi^i_{x_\alpha=x_\beta=v_i}, f_r\right)$ contains the set of write functions from which f_r reads on this input.

Because f_r reads from f_w on input $\Phi^i_{x_\alpha=x_\beta=v_i}$, they must access the same cell on this input. Therefore, by Lemma 6.1, the pair of tuples $(\mathcal{T}_{f_r}, \mathcal{T}_{f_w})$ covers the edge $\{x_\alpha, x_\beta\}$. In addition, $\mathcal{T}_{f_r} \in \mathcal{A}_{[t_w..T]}$ and $\mathcal{T}_{f_w} \in \mathcal{B}_{t_w}$, for some time step t_w. Therefore, to show that the edge $\{x_\alpha, x_\beta\}$ is covered by the tuple system $\left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w}\right)$, it is sufficient to show that $\mathcal{T}_{f_r} \notin \mathcal{B}_{t_w}$.

Suppose by contradiction that $\mathcal{T}_{f_r} \in \mathcal{B}_{t_w}$. Then, by definition of $\mathcal{B}_{t_w}$, there exists a write addressing function $f_{hit} \in \mathcal{F}^{write}_{t_w}$ used during time step t_w which contributes the labeled tuple $\mathcal{T}_{f_r}$. Because f_r and f_{hit} contribute the same labeled tuple, they must depend on the same set of variables, i.e. $\mathcal{X}(f_r) = \mathcal{X}(f_{hit})$, and hence must be similar. In addition, both functions give the same label to x_α, so they must access the same cell on the input $\Phi^i_{x_\alpha=v_i}$ and therefore on $\Phi^i_{x_\alpha=x_\beta=v_i}$. It follows that f_w and f_{hit} write to the same cell during time step t_w. To conclude, if f_w is contained in $\Gamma^{\text{alg}}_{[1..t]}\left(\Phi^i_{x_\alpha=x_\beta=v_i}, f_r\right)$, then so is f_{hit}. This contradicts the fact that there are no write functions in $\Gamma^{\text{alg}}_{[1..t]}\left(\Phi^i_{x_\alpha=x_\beta=v_i}, f_r\right)$ that are similar to f_r. ■

7 Graph Theory

Lemma 7 *There exists a pair of variables $\{x_\alpha, x_\beta\}$, such that: x_α and x_β are contained in the same subproblem Π_T^i for some $i \in [1..q_T]$; the edge $\{x_\alpha, x_\beta\}$ is not covered by the collection of labeled tuple systems $\left\{\left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w}\right) \mid t_w \in [1..T]\right\}$; and neither variables are seen by processor P_1, i.e. $x_\alpha, x_\beta \notin \mathcal{V}_{\langle P_1,T\rangle}$.*

Another combinatorial object is required. A **semi-partition** $\Pi = \langle \pi_1, \ldots, \pi_r \rangle$ of a set $V = \{x_1, \ldots, x_n\}$ is a set of disjoint subsets of V. Boppana thinks of it as a partial function from V to $[1..r]$. The semi-partition covers the edge $\{x_\alpha, x_\beta\}$ if x_α and x_β are contained in different subsets $\pi_1, \ldots, \pi_r$. Note that this forms a complete multipartite graph. The *size* of Π is $|\Pi| = \sum_i |\pi_i|$. The *entropy* of Π is defined as follows. Uniformly at random choose a variable x from $\bigcup_{i\in[1..r]} \pi_i$. The entropy $H(\Pi) = \sum_{i\in[1..q_t]} -\Pr[x \in \pi_i] \log_2 \Pr[x \in \pi_i] = -\sum_i \frac{|\pi_i|}{|\Pi|} \log_2 \left(\frac{|\pi_i|}{|\Pi|}\right)$ is the expected number of bits to specify which of the sets π_i that x is contained in. The *cost* of Π is defined in the same way except that x is randomly chosen from V, giving $cost(\Pi) = \frac{|\Pi|}{|V|} H(\Pi)$. The size of a collection of semi-partitions is the sum of the individual sizes and the cost of the collection is the sum of the individual costs. A useful result due to Fredman and Komlós (1984) says that if a collection of semi-partitions covers the complete graph on n vertices then the cost of the collection is at least $\log_2 n$. (For another proof, see Körner (1986).)

In order to prove Lemma 7, we cover all the unacceptable edges of G with four collections of semi-partitions, $\Pi_\Pi, \Pi_\mathcal{V}, \Pi_{\langle A,B\rangle}$, and Π_E. If by contradiction there is no uncovered edge, then the sum cost of these collections is at least $\log_2 n$. It is sufficient then to show that if the cost of $\Pi_\Pi, \Pi_\mathcal{V}$, or $\Pi_{\langle A,B\rangle}$ is at least $.1 \log_2 n$ or if the cost of Π_E is at least $.7 \log_2 n$, then $T \in \Omega\left(\frac{n}{p} \frac{\log n}{\log(\frac{n}{p} \log n)}\right)$.

Note that the subproblems $\langle \Pi_T^1, \ldots, \Pi_T^{q_T} \rangle$ themselves form a semi-partition. This is denoted Π_Π. If $\{x_\alpha, x_\beta\}$ is not covered by this semi-partition then x_α and x_β are contained in the same subproblem Π_T^i. $|\Pi_\Pi| = |V|$. Therefore, $cost(\Pi_\Pi) = H(\Pi_\Pi)$. condition (3) bounds the entropy $H(\Pi_\Pi)$. As stated in Section 2, Boppana proves that if $H(\Pi_\Pi)$ is at least $.1 \log_2 n$, than T is as required.

In order to ensure that $x_\alpha, x_\beta \notin \mathcal{V}_{\langle P_1,T\rangle}$, we use the semi-partition $\left\langle \mathcal{V}_{\langle P_1,T\rangle}, V - \mathcal{V}_{\langle P_1,T\rangle} \right\rangle$ and form a collection of semi-partitions to cover all the edges in the clique $\mathcal{V}_{\langle P_1,T\rangle}$. The entropy of any random variable that takes on only two values is at most 1. Hence, the cost of $\left\langle \mathcal{V}_{\langle P_1,T\rangle}, V - \mathcal{V}_{\langle P_1,T\rangle} \right\rangle$ is at most 1. By condition (8), $\left|\mathcal{V}_{\langle P_1,T\rangle}\right| = q_T < n^{0.1}$. The result by Fredman and Komlós is tight so there is a collection with cost $\log_2(n^{0.1}) = 0.1 \log_2 n$ that covers the clique.

What remains is forming two collections of semi-partitions, $\Pi_{\langle A,B\rangle}$ and Π_E, that cover the edges covered by the collection of labeled tuple systems $\left\{\left(\mathcal{A}_{[t_w..T]} - \mathcal{B}_{t_w}, \mathcal{B}_{t_w}\right) \mid t_w \in [1..T]\right\}$.

Lemma 4.1 *Given a collection of labeled tuple systems $\mathcal{T} = \{\langle \mathcal{A}_i, \mathcal{B}_i \rangle \mid i \in [1..r]\}$ with tuples of length at most q, there exists a collection of semi-partitions $\Pi_{\langle A,B\rangle} = \{\langle A_i, B_i \rangle \mid i \in [1..r']\}$ that covers all the edges covered by $\mathcal{T}$ except for the edges in a set E of size $q\sqrt{n}\log(|\mathcal{T}|)|\mathcal{T}| + \frac{|\mathcal{T}|^2}{\sqrt{n}}$. Furthermore, $\sum_i |A_i| \le \sum_i |\mathcal{A}_i|$ and $\sum_i |B_i| \le \sum_i |\mathcal{B}_i|$.*

As q becomes larger, this bound becomes weaker. It is conjectured that the bound remains the same regardless of the length of the tuples.

The proof of this lemma is taken from [RSSW88]. In that setting the tuples are not labeled. However, the fact that the tuples are labeled does not effect the proof at all.

Proof of Lemma 4.1: Initially, let $\Pi_{\langle A,B\rangle}$ and E be empty. Each labeled tuple system $\langle \mathcal{A}_i, \mathcal{B}_i\rangle \in \mathcal{T}$ will add a number of semi-partitions to $\Pi_{\langle A,B\rangle}$ and a number of edges to E. A pair of labeled tuples is said to be a *covering* pair if they cover an edge. We say that $\langle \mathcal{A}_i, \mathcal{B}_i\rangle$ is *sparse* if it contains at most $\frac{|\mathcal{A}_i||\mathcal{B}_i|}{\sqrt{n}}$ covering pairs. If it is not sparse, then there exists some tuple $\tau \in \mathcal{A}_i$, which forms a covering pair with at least $\frac{|\mathcal{B}_i|}{\sqrt{n}}$ tuples in $\mathcal{B}_i$. Since τ is a tuple of length at most q, there is some coordinate p such that at least $\frac{|\mathcal{B}_i|}{q\sqrt{n}}$ tuples in $\mathcal{B}_i$ differ from τ only in position p and have the same label on this coordinate. Let $\mathcal{A}'_i$ be the subset (including τ) of $\mathcal{A}_i$ with this property and $\mathcal{B}'_i$ be the subset of $\mathcal{B}_i$. Let A be the set of vertices occurring at position p in the tuples of $\mathcal{A}'_i$ and similarly B of $\mathcal{B}'_i$. The label of these vertices are all same and hence can be removed. Therefore, $\langle A, B\rangle$ is a semi-partition which covers the same edges as the labeled tuple system $\langle \mathcal{A}'_i, \mathcal{B}'_i\rangle$.

We remove $\mathcal{A}'_i$ from $\mathcal{A}_i$ and $\mathcal{B}'_i$ from $\mathcal{B}_i$. We have not accounted for edges covered by pairs of tuples between $\mathcal{A}'_i$ and $\mathcal{B}_i - \mathcal{B}'_i$ or between $\mathcal{B}'_i$ and $\mathcal{A}_i - \mathcal{A}'_i$. But any tuple in $\mathcal{B}_i - \mathcal{B}'_i$ covers at most one edge with tuples in $\mathcal{A}'_i$, for it can differ from a tuple in $\mathcal{A}'_i$ only in a coordinate $p' \neq p$ and tuples in $\mathcal{A}'_i$ have the same variable in position p' and different variables in position p. Similarly, any tuple in $\mathcal{A}_i - \mathcal{A}'_i$ covers at most one edge with tuples in $\mathcal{B}'_i$. Thus we have neglected at most $|\mathcal{A}_i| + |\mathcal{B}_i|$ edges. These are added to E.

If $\langle \mathcal{A}_i, \mathcal{B}_i\rangle$ is not yet sparse, we repeat the process. Each time, we shrink $\mathcal{B}_i$ by at least a factor of $\mu = 1 - \frac{1}{q\sqrt{n}}$. In at most $\log_{\frac{1}{\mu}}(|\mathcal{B}_i|)$ iterations, $\langle \mathcal{A}_i, \mathcal{B}_i\rangle$ becomes sparse (note that an empty tuple system is sparse). The total number of pairs we have added to E is $\log_{\frac{1}{\mu}}(|\mathcal{B}_i|)(|\mathcal{A}_i| + |\mathcal{B}_i|) \leq q\sqrt{n}\log(|\mathcal{B}_i|)(|\mathcal{A}_i| + |\mathcal{B}_i|)$. When $\langle \mathcal{A}_i, \mathcal{B}_i\rangle$ becomes sparse, we simply add to E the edges covered by this system. This adds at most $\frac{|\mathcal{A}_i||\mathcal{B}_i|}{\sqrt{n}}$ more edges. When this process is completed, it is clear that $\sum_i |A_i| \leq \sum_i |\mathcal{A}_i|$ and $\sum_i |B_i| \leq \sum_i |\mathcal{B}_i|$. Furthermore, $|E| \leq \sum_i \left[q\sqrt{n}\log(|\mathcal{B}_i|)(|\mathcal{A}_i| + |\mathcal{B}_i|) + \frac{|\mathcal{A}_i||\mathcal{B}_i|}{\sqrt{n}}\right] \leq q\sqrt{n}\log(|\mathcal{T}|)|\mathcal{T}| + \frac{|\mathcal{T}|^2}{\sqrt{n}}$. ■

The next step is to form a collection of semi-partitions Π_E that cover the edges in E. Boppana, Lemma 3.4 [B89], proves that every graph with n vertices and $|E|$ edges has a coloring with entropy of at most $\log_2\left(\left(\frac{|E|}{n} + 1\right)e\right)$. The semi-partition Π_E is formed by putting together vertices with the same color. The edges of E are guaranteed to be covered by Π_E, because they are bi-chromatic. By Lemma 6, $\sum_i |\mathcal{A}_i| \leq pT^2$ and $\sum_i |\mathcal{B}_i| \leq pT$. Therefore, $|\mathcal{T}| = \sum_i |\mathcal{A}_i| + |\mathcal{B}_i| \leq 2pT^2$. If $p \geq n\log n$, then the lower bound in Theorem 2 becomes $\Omega(1)$. Therefore, assume $p \leq n\log n$. If $T > \log n$, then Theorem 2 follows. Therefore, assume that $T \leq \log n$. By condition (8), the tuples have length $q_T < n^{0.1}$. Therefore, by Lemma 4.1, $|E| \leq q\sqrt{n}\log(|\mathcal{T}|)|\mathcal{T}| + \frac{|\mathcal{T}|^2}{\sqrt{n}} \leq n^{1.6}$. The cost of Π_E is equal to its entropy, which is less than $\log_2\left(\left(\frac{|E|}{n} + 1\right)e\right) \leq 0.7\log n$.

What remains is to bound the cost of the collection of semi-partitions $\Pi_{\langle A,B\rangle}$. Ragde et. al. [RSSW88], Boppana [B89], and I have different ways of doing this. Ragde et. al. use the fact that $\sum_i |A_i| + |B_i| \leq 2pT^2$ and that $H(\langle A_i, B_i\rangle) \leq 1$ to conclude that $cost\left(\Pi_{\langle A,B\rangle}\right) = \sum_i \frac{|\langle A_i,B_i\rangle|}{|V|} H(\langle A_i, B_i\rangle) \leq \frac{2pT^2}{n}(1)$. This gives the weaker result of $T \in \Omega\left(\sqrt{\frac{n}{p}\log n}\right)$. Boppana uses a semi-partition with T parts and of size $2pT$. The cost of this is at most $\frac{2pT}{n}(\log T)$, which give the required T. My method uses that fact that the semi-partitions are unbalanced, namely

that the sets A_i are bigger than the sets B_i. Generally this is the case since Lemma 6 gives that $\sum_i |A_i| \leq pT^2$ and $\sum_i |B_i| \leq pT$. I will show that unbalanced semi-partitions are not able to cover edges effectively.

As a first step of this proof, consider the situation in which $|A_i| = T|B_i|$, for each i. Given this situation, the entropy is easy to compute. $H(\langle A_i, B_i \rangle) = -\frac{|A_i|}{|A_i|+|B_i|} \log_2 \left(\frac{|A_i|}{|A_i|+|B_i|}\right) - \frac{|B_i|}{|A_i|+|B_i|} \log_2 \left(\frac{|B_i|}{|A_i|+|B_i|}\right) = -\frac{T}{T+1} \log_2 \left(\frac{T}{T+1}\right) - \frac{1}{T+1} \log_2 \left(\frac{1}{T+1}\right)$. Note that $-\log_2 \left(\frac{T}{T+1}\right) \leq \frac{1}{T}$. Therefore, $H(\langle A_i, B_i \rangle) \leq \frac{1}{T+1} + \frac{\log_2(T+1)}{T+1} \leq \frac{2\log_2(T)}{T}$. It follows that $cost\left(\Pi_{\langle A,B \rangle}\right) = \sum_i \frac{|\langle A_i, B_i \rangle|}{|V|} H(\langle A_i, B_i \rangle) \leq \left(\frac{2pT^2}{n}\right)\left(2\frac{\log_2 T}{T}\right) = 4\frac{p}{n} T \log_2 T$. Therefore, if $cost\left(\Pi_{\langle A,B \rangle}\right)$ is more than $.1 \log_2 n$, then $T \in \Omega\left(\frac{n}{p} \frac{\log n}{\log(\frac{n}{p} \log n)}\right)$.

This bounds $cost\left(\Pi_{\langle A,B \rangle}\right)$ under the assumption that $|A_i| = T|B_i|$, for each i. The next lemma proves that $cost\left(\Pi_{\langle A,B \rangle}\right)$ is maximized when this assumption is true.

Lemma 4.2 *$C = \sum_i \frac{|A_i|+|B_i|}{|V|} H(\langle A_i, B_i \rangle)$ is maximized when the semi-partitions are as balanced as possible, i.e. $|A_i| = T|B_i|$, for each i.*

Proof of Lemma 4.2: For each $i \in [1..r']$, let $c_i = |A_i| + |B_i|$, $\alpha_i = \frac{|A_i|}{c_i}$, and $1 - \alpha_i = \frac{|B_i|}{c_i}$. The following notation will be helpful. Let $h(x) = x \log_2(x) + (1-x)\log_2(1-x)$. Substituting these into the equation gives $C = -\sum_i \frac{c_i}{n}[\alpha_i \log_2(\alpha_i) + (1-\alpha_i)\log_2(1-\alpha_i)] = -\sum_i \frac{c_i}{n} h(\alpha_i)$. The claim is that this is maximized when all the α_i have the same value. Suppose that C is maximized with the values $\alpha_1, \ldots, \alpha_{r'}$ and suppose by way of contradiction, that there are indexes $j, k \in [1..r']$ for which $\alpha_j \neq \alpha_k$. Keep everything fixed except for α_j and α_k. This gives

$$C = \frac{-c_j}{n} h(\alpha_j) + \frac{-c_k}{n} h(\alpha_k) + \sum_{i \notin \{j,k\}} \frac{-c_i}{n} h(\alpha_i) \qquad \text{and}$$

$$\frac{dC}{d\alpha_j} = \frac{-c_j}{n} h'(\alpha_j) + \frac{-c_k}{n} h'(\alpha_k) \frac{d\alpha_k}{d\alpha_j} + 0$$

Since $\sum_i |A_i| = pT^2$ and $c_i \alpha_i = |A_i|$, we can conclude that $\alpha_k = \frac{|A_k|}{c_k} = \frac{pT^2 - c_j\alpha_j - \sum_{i \notin \{j,k\}} c_i \alpha_i}{c_k}$ and that $\frac{d\alpha_k}{d\alpha_j} = \frac{-c_j}{c_k}$. Therefore, setting $\frac{dC}{d\alpha_j}$ to zero gives $-h'(\alpha_j) + h'(\alpha_k) = 0$. Because $h'(x) = \log_2(x) - \log_2(1-x)$, it follows that, $\log_2(\alpha_j) - \log_2(1-\alpha_j) = \log_2(\alpha_k) - \log_2(1-\alpha_k)$. This gives $\log_2(\alpha_j - \alpha_j\alpha_k) = \log_2(\alpha_k - \alpha_j\alpha_k)$ and $\alpha_j - \alpha_j\alpha_k = \alpha_k - \alpha_j\alpha_k$. We can concluded that C is maximized when $\alpha_j = \alpha_k$. ■

We can conclude that the costs of the four collections of semi-partitions $\Pi_\Pi, \Pi_\mathcal{V}, \Pi_{\langle A,B \rangle}$, and Π_E are less than either $.1 \log_2 n$ or $.7 \log_2 n$. Therefore, these semi-partitions cannot cover all the edges of the element distinctness graph G. Hence, an edge exists meeting the requirements of Lemma 7. ■.

As proved in Section 3.1, the Lemmas 4, 5, 6, and 7 together prove Theorem 2. Let $\{x_\alpha, x_\beta\}$ be an edge with the properties stated in Lemma 7. From Lemmas 5 and 6, it follows that all the addressing functions interact as revealed by the adversary on the inputs Φ^i and $\Phi^i_{x_\alpha = x_\beta = v_i}$. Hence, by Claim 2, the state of P_1, for these two inputs, at the end of step T depends only on the fixed set of input variables $\mathcal{V}_{\langle P_1, T \rangle}$ seen by him. P_1 does not see x_α or x_β and therefore cannot distinguish between the inputs Φ^i and $\Phi^i_{x_\alpha = x_\beta = v_i}$, which differ only on these variables. Therefore, on these

inputs, P_1 is unable to determine whether or not the input is element distinct. ■

8 Open Problems

Finding lower bounds for Element Distinctness when defined on even smaller domains is still open. For example, nothing is known when the variables take on values up to n^2. It is also open whether PRIORITY and COMMON can be separated when the number of memory cells is bounded.

Acknowledgments

I would like to thank my supervisor, Faith Fich, for her inspiration and guidance. I would also like to thank my Paul Beame, Toni Pitassi, and an unknown referee for their useful comments.

References

[B89] R. B. Boppana, "Optimal Separations Between Concurrent-Write Parallel Machines," *Proc. 21st ACM Symposium on Theory of Computing*, pp. 320-326, 1989.

[BH87] P. Beame and J. Hastad, "Optimal bounds for decision problems on the CRCW PRAM," *Proc. 19th ACM Symposium on Theory of Computing*, pp. 83-93, 1987.

[CDHR88] B. S. Chlebus, K. Diks, T. Hagerup, and T. Radzik, "Efficient simulations between concurrent-read concurrent-write PRAM models," *13th Symp. Math. Found. Comp. Sci., Lecture Notes in Comp. Sci.* vol. 324, Springer-Verlag, pp. 231-239, 1988.

[CDR86] S. A. Cook, C. Dwork, and R. Reischuk, "Upper and lower bounds for parallel random access machines without simultaneous writes," *SIAM J. Comp.* vol. 15, pp. 87-97, 1986.

[FMRW85] F. E. Fich, F. Meyer Auf Der Heide, P. L. Ragde, and A. Wigderson, "One, Two, ... Infinity: Lower Bounds for Parallel Computation," *Proc. 17th ACM Symposium on Theory of Computing*, pp. 48-58, 1985.

[FMW86] F. E. Fich, F. Meyer Auf Der Heide, and A. Wigderson, "Lower bounds for parallel random-access machines with unbounded shared memory," *Advances in Computing Research.*, vol. 4, pp. 1-15, 1986.

[FRW88] F. E. Fich, P. L. Ragde, and A. Wigderson, "Relations Between Concurrent-Write Models of Parallel Computation," *SIAM J. Comp.*, vol. 17, pp. 606-627, 1988.

[FRW88-2] F. E. Fich, P. L. Ragde, and A. Wigderson, "Simulations among concurrent-write PRAMs," *Algorithmica* 3, pp. 43-52, 1988.

[FK84] M. Fredman and J. Komlós, "On the size of separating systems and families of perfect hash functions," *SIAM J. Alg. Disc. Meth.* 5, 61-68, 1984.

[K88] L. Kucera (1982), "Parallel computation and conflicts in memory access," *Inf. Proc. Letters*, vol. 14, pp. 93-96, 1988.

[R] P. L. Ragde, "Processor-Time Tradeoffs in PRAM Simulations," to appear in *J. Comput. Syst. Sciences*.

[RSSW88] P. L. Ragde, W. L. Steiger, E. Szemerédi, and A. Wigderson, "The parallel complexity of element distinctness is $\Omega((\log n)^{\frac{1}{2}})$," *SIAM J. Disc. Math.*, vol. 1, pp. 399-410, 1988.