ELSEVIER

Information Processing Letters 62 (1997) 103–110

Information Processing Letters

Transforming comparison model lower bounds to the parallel-random-access-machine

Dany Breslauer [a,1], Artur Czumaj [b,1,2], Devdatt P. Dubhashi [a,*,1], Friedhelm Meyer auf der Heide [b,c,1,2]

[a] *BRICS – Basic Research in Computer Science, Centre of the Danish National Research Foundation, Department of Computer Science, University of Aarhus, DK-8000 Aarhus C, Denmark*

[b] *Heinz Nixdorf Institute, University of Paderborn, D-33095 Paderborn, Germany*

[c] *Department of Mathematics and Computer Science, University of Paderborn, D-33095 Paderborn, Germany*

Received 19 July 1995
Communicated by F. Dehne

Abstract

We provide general transformations of lower bounds in Valiant's parallel-comparison-decision-tree model to lower bounds in the priority concurrent-read concurrent-write parallel-random-access-machine model. The proofs rely on standard Ramsey-theoretic arguments that simplify the structure of the computation by restricting the input domain. The transformation of comparison model lower bounds, which are usually easier to obtain, to the parallel-random-access-machine, unifies some known lower bounds and gives new lower bounds for several problems.

Keywords: Lower bounds; PRAM model; Comparison model; Ramsey theory

1. Introduction

Valiant's parallel-comparison-decision-tree model [26] is very attractive for studying parallel algorithms and lower bounds for *order invariant problems* whose solution depends on equality or order relations between the input variables. One of the major drawbacks of this model, however, is that the information obtained by determining the relations between input variables becomes "common knowledge" and the model fails to capture the difficulty in communicating information between various processing units that run in parallel. In this respect, the parallel-random-access-machine model (PRAM) is more realistic because it captures some issues of communication between the different processing units, that makes it a more natural model to describe parallel algorithms.

The two models, however, are not comparable in general. While some problems in the comparison model, e.g. finding the maximum [26], have similar algorithms in the PRAM model [25], for other prob-

* Corresponding author. Current address: SPIC Mathematical Institute, 92 G.N. Chetty Road, T. Nagar, Chennai, 600 017, India. E-mail: dubhashi@ssf.ernet.in.

[1] Partially supported by the ESPRIT Basic Research Action Program of the EC under contract #7141 (ALCOM II).

[2] Partially supported by DFG-Graduiertenkolleg "Parallele Rechnernetzwerke in der Produktionstechnik", ME 872/4-1, and by DFG Grant DI 412/2.

PII S0020-0190(97)00032-X

lems, e.g. finding the median [1,5], only much slower PRAM algorithms [7] are known for any polynomial number of processors. On the other hand, there exist problems, e.g. element distinctness, that have slow comparison model algorithms [6] and constant-time PRAM algorithms on integer input domains.

Since comparison model lower bounds are often easier to obtain than PRAM lower bounds, it can sometimes be useful to translate comparison model lower bounds into PRAM lower bounds. Clearly, if a PRAM algorithm can only access its input by determining the relations between the input variables, then the comparison model lower bounds will hold for the PRAM. However, this assumption prevents PRAM algorithms from using their powerful capabilities. Moreover, solutions to problems that are defined in the equality-comparison model sometimes benefit from the introduction of an arbitrary order on the input domain so that order comparisons can be used; e.g. element distinctness. Since the input variables on the PRAM are usually assumed to be integers, the input domain is naturally ordered. This makes lower bounds in the equality-comparison model inapplicable to the PRAM model.

This note gives two general translations of lower bounds in the order-comparison model to lower bounds in the priority CRCW-PRAM model:

(1) Any comparison model lower bound can be converted into a corresponding lower bound in the priority CRCW-PRAM with bounded memory. By bounded memory we mean that the memory size is not permitted to grow as a function of the input domain size.
(2) Any comparison model lower bound that holds if the input variables are known to be all *distinct* can be converted into a corresponding lower bound in the priority CRCW-PRAM with infinite memory.

The proof techniques used are standard multivariable Ramsey-theoretic arguments that were developed by several authors for studying specific problems [9,19,21–24]. The main idea is that one can restrict the original input domain in such a way that the processors must communicate in a manner that depends only on the relative order between the input variables. This implies that the PRAM can only determine the relations between input variables that were communicated to a given processor and not by the communication pattern itself. We then apply comparison model lower bounds to obtain lower bounds on the PRAM.

The transformation of the comparison model lower bounds provides a unified way to obtain lower bounds for the PRAM. It generalizes previous results and provides PRAM lower bounds for problems that had only comparison model lower bounds. Some of the lower bounds obtainable by the transformation, for input of size n on a p processor PRAM, are:

(1) Sorting requires

$$\Omega\left(n/p + \log_{\lceil (p/n)\log n+1\rceil} n\right)$$

time [9,21].

(2) Element distinctness requires

$$\Omega\left(n/p + \log_{\lceil (p/n)\log n+1\rceil} n\right)$$

time if the memory size is bounded [9,16].

(3) Finding the maximum and merging require

$$\Omega\left(n/p + \log\log_{\lceil p/n+1\rceil} n\right)$$

time [19,24].

(4) String matching and some related problems on strings require

$$\Omega\left(n/p + \log\log_{\lceil p/n+1\rceil} n\right)$$

time, if the memory size is bounded.

(5) Finding an approximate maximum, namely, an element whose rank belongs in the top εn ranks, requires

$$\Omega\left(n/p + \log\log_{\lceil p/n+1\rceil}(1/\varepsilon) + \log^* n - \log^*(p/n)\right)$$

time [3], for $1/n \leqslant \varepsilon \leqslant 1/2$.

The last two lower bounds are new. The paper is organized as follows. Sections 2 and 3 review Valiant's parallel-comparison-decision-tree and the parallel-random-access-machine models. Section 4 gives the general PRAM lower bounds and Section 5 shows how these lower bounds are applied to specific problems. Conclusions and open problems are given in Section 6.

[3] Define $\log^{(0)} n = n$, $\log^{(i)} n = \log\log^{(i-1)} n$ and $\log^* n = \min\{i \mid \log^{(i)} n \leqslant 1\}$. In this paper $\log n = \max\{0, \log_2 n\}$.

2. Parallel comparison models

The input variables $x_1, \ldots, x_n$ are chosen from some infinite totally ordered domain $\mathscr{D}$. Denote by $\mathscr{D}^k$ the set of all k-tuples of elements of $\mathscr{D}$, by $\mathscr{D}^k_*$ the set of all k-tuples of $\mathscr{D}$ with no two equal elements and by $\mathscr{D}^k_<$ the set of all increasing k-tuples of $\mathscr{D}$.

Following the notation of [22], two tuples $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ are said to be *order equivalent* if $x_i < x_j \Leftrightarrow y_i < y_j$, for all $i, j = 1, \ldots, k$. (And hence $x_i = x_j \Leftrightarrow y_i = y_j$.) The equivalence class containing $x_1, \ldots, x_k$ is called the *order type* of $x_1, \ldots, x_k$. A decision problem $\mathscr{P}$ on the variables $x_1, \ldots, x_n$ partitions the inputs from $\mathscr{D}^n$ into classes $\mathscr{P}_1, \ldots, \mathscr{P}_q$. $\mathscr{P}$ is said to be *order invariant* if order equivalent tuples are always in the same class.

A comparison between two variables $x_i :: x_j$ determines if $x_i < x_j$, $x_i = x_j$ or $x_i > x_j$. In some cases, we will also consider the additional relations $x_i \neq x_j$, without order information. These inequality relations, which may not be established by comparisons, might instead be given *a priori*, as part of the definition of a problem. Such *a priori* restrictions are useful for problems that are defined on partial domains; e.g. in the merging problem the two lists to be merged are assumed to be sorted. Restricting the input so that all variables are distinct, i.e. $x_i \neq x_j$, for $i \neq j$, will be of particular interest in this paper. We refer to this restriction as the *distinctness assumption*.

An order invariant problem $\mathscr{P}(x_1, \ldots, x_n)$ can be solved by comparing pairs of input variables until all input tuples satisfying the relations that were established are in the same class $\mathscr{P}_i$; e.g. in the problem of finding the maximum it suffices to discover that some $x_i \geqslant x_j$, for $j = 1, \ldots, n$, without caring about the relative order between the other variables. Clearly, sorting is the hardest problem is this sense since it determines the exact relations between all input variables and thus the order type of the input.

Valiant's parallel-comparison-decision-tree model [26] proceeds in rounds in which up to p pairwise comparisons of input variables are made simultaneously. According to the outcome of the comparisons, and the relations established in previous rounds, the comparison model algorithm decides which variables to compare in the next round, or it may decide to terminate with an answer. We denote by $\mathscr{C}_{\mathscr{P}}(n, p)$ the depth of the shallowest comparison decision tree that solves the problem $\mathscr{P}(x_1, \ldots, x_n)$ using p comparisons in each round. Clearly, comparing all $\binom{n}{2}$ pairs of variables gives complete information about their order type, and thus, for any problem $\mathscr{P}$, $\mathscr{C}_{\mathscr{P}}(n, \binom{n}{2}) \leqslant 1$.

Boppana [9], following Meyer auf der Heide and Wigderson [21], defines a similar comparison decision tree model that we call the *merging-comparison decision tree*. In the p processor merging-comparison model, each processor knows a certain subset of the input variables and their order type (initially these sets are empty). In every round, according the partial order that is formed by the order types of the subsets of variables known by all processors, the merging-comparison model decides whether to terminate with an answer or to continue, letting each processor *merge* its set of variables either with the set of variables known by some other processor at the end of the previous round or with a single input variable. We denote by $\mathscr{M}_{\mathscr{P}}(n, p)$ the depth of the shallowest p processor merging-comparison decision tree for the problem $\mathscr{P}(x_1, \ldots, x_n)$. The following lemma relates lower bounds in the parallel comparison model to the merging-comparison model.

Lemma 2.1. *Let*

$$\hat{\mathscr{C}}_{\mathscr{P}}(n, p) = \max\{t \mid \mathscr{C}_{\mathscr{P}}(n, 2^{2(t-1)}p) \geqslant t\}.$$

Then, $2 \cdot \mathscr{C}_{\mathscr{P}}(n, p) \geqslant \mathscr{M}_{\mathscr{P}}(n, p) \geqslant \hat{\mathscr{C}}_{\mathscr{P}}(n, p)$.

Proof. Clearly, every comparison model round with p comparisons can be simulated by at most two rounds of a p processor merging-comparison model, establishing that $2 \cdot \mathscr{C}_{\mathscr{P}}(n, p) \geqslant \mathscr{M}_{\mathscr{P}}(n, p)$. Inductively, the number of variables known by each processor after h rounds is at most 2^{h-1}. The relations established by merging two sets of at most 2^{h-1} variables can be determined by performing $2^{2(h-1)}$ comparisons. Hence, a p processor merging-comparison model can be simulated by a comparison model that makes $2^{2(h-1)}p$ comparisons in round number $h+1$. If there are at most t rounds, this can be overestimated by $2^{2(t-1)}p$ comparisons in each round. If $\mathscr{C}_{\mathscr{P}}(n, 2^{2(t-1)}p) \geqslant t$, then even with this

larger number of comparisons the solution of the problem $\mathscr{P}$ requires at least t rounds. □

We say that a comparison model lower bound $\mathscr{C}_{\mathscr{P}}(n, p)$ is *resilient* if $\alpha \cdot \hat{\mathscr{C}}_{\mathscr{P}}(n, p) + \beta \geqslant \mathscr{C}_{\mathscr{P}}(n, p)$ for some constants α and β and all $n, p \geqslant 1$. When this holds, we write $\mathscr{C}_{\mathscr{P}}(n, p) = \Theta(\hat{\mathscr{C}}_{\mathscr{P}}(n, p))$. Resilient comparison model lower bounds translate to the same lower bounds in the merging-comparison model (up to constants).

3. The parallel-random-access-machine

In this paper we consider a powerful version of the *priority concurrent-read concurrent-write parallel-random-access-machine* (*CRCW-PRAM*). The model consists of p synchronous processors that communicate via a shared memory with cells of *unlimited size*. Processors are allowed to read and write simultaneously at the same memory location; write conflicts are resolved by accepting the value that is written by the processor with the highest preassigned priority.

We assume that the execution of a PRAM program proceeds in rounds. Each round consists of a computation phase in which every processor can make *any* computation on the information it has obtained before, followed by a write phase and then by a read phase. Note that these assumptions result in an extremely powerful model that can compute *any function* in O(log n) steps using $n/\log n$ processors. Hence, lower bounds in this model emphasize the limits of the interprocessor communication. We say that the PRAM solves an order invariant problem $\mathscr{P}$, if for each pair of order types in different $\mathscr{P}$-equivalence classes, there exists at least one processor that is able to distinguish between them.

Let $R_{i,t}$, for $i \in [p]$, denote the *read access* function of processor number i at round t; i.e. at round t, processor i reads the memory cell whose address is given by $R_{i,t}$. Similarly, let $W_{i,t}$, for $i \in [p]$, denote the *write access* function of processor i at round t; i.e. at round t, processor i writes the value $X_{i,t}$ into the memory cell whose address is $W_{i,t}$. $R_{i,t}$, $W_{i,t}$ and $X_{i,t}$ are functions of the *state* of processor i at round t. The lower bound arguments given next show that by restricting the input domain, it is possible to simplify the interaction between processors so that the state of each processor depends only on the input variables it "knows".

4. Lower bounds

In this section we show that, under certain assumptions, a PRAM algorithm for an order invariant problem can be simulated by a merging-comparison decision tree on some restricted input domain. This allows us to transform lower bounds from the merging-comparison model to lower bounds in the PRAM model. The arguments are essentially the same as those used by Meyer auf der Heide and Wigderson [21] and Boppana [9]; we observe that these arguments are more generally applicable than to the problems considered in those papers and even when input variables are allowed to be equal. In Section 4.1 we summarize the Ramsey-theoretic components of the proofs. In Section 4.2 we give the lower bounds for PRAM algorithms with bounded memory and in Section 4.3 for PRAM algorithms with unbounded memory, under the distinctness assumption.

4.1. Ramsey theory

Let $\mathscr{D}$ be an infinite totally ordered set. We say that a function f is a $\mathscr{D}$ *fixed order type* function if f is defined on tuples from $\mathscr{D}^k$ of a fixed order type, for some $k \geqslant 0$. The *standard form* of f is obtained by removing all but one representative of equal variables (since the domain of f has a fixed order type, equal variables are equal on all the domain); removing all variables that f does not depend on; and reordering the remaining variables in increasing order according to their order type. Hence the standard form of f is defined on the domain $\mathscr{D}_{<}^{l}$, for some $l \leqslant k$.

Given a function f that is defined on some subset of $\mathscr{D}^k$, we denote by $f|_{\mathscr{E}}$, for $\mathscr{E} \subseteq \mathscr{D}$, the restriction of f to $\mathscr{E}^k$. Similarly, if $\mathscr{F}$ is a family of functions, we use the notation $\mathscr{F}|_{\mathscr{E}} := \{f|_{\mathscr{E}} \mid f \in \mathscr{F}\}$. The following lemma can be derived from the "canonical" Ramsey theorem due to Erdös and Rado [17]. (See also Section 5.5 in [20].)

Lemma 4.1. *Let $\mathcal{F}$ be a finite collection of $\mathcal{D}$ fixed order type functions. Then, there exists an infinite subset $\mathcal{E} \subseteq \mathcal{D}$, such that the standard form of every function in $\mathcal{F}|_{\mathcal{E}}$ is injective and every pair of functions in $\mathcal{F}|_{\mathcal{E}}$ either have identical standard forms or disjoint ranges. In particular, if the range of the functions in $\mathcal{F}$ is finite, then the functions in $\mathcal{F}|_{\mathcal{E}}$ are constant.*

4.2. Bounded memory

In this section we assume that the memory size m is fixed. Hence all read and write access functions have the finite range $[m]$. We will construct a merging-comparison decision tree that will simulate the computation of the PRAM.

The construction proceeds step by step. We will maintain the set of variable indices $V_{i,t} \subseteq [n]$ known by processor $i \in [p]$ at round t and an infinite set $S_t \subseteq \mathbb{N}$, such that the input tuples will be restricted to S_t^n. Initially $V_{i,0} = \emptyset$ and $S_0 = \mathbb{N}$, and for $t \geqslant 1$, $V_{i,t-1} \subseteq V_{i,t}$ and $S_t \subseteq S_{t-1}$, for $i \in [p]$. Also, we shall maintain that for inputs in S_t, the state of the processor i in the original PRAM, and therefore $R_{i,t}$, $W_{i,t}$ and $X_{i,t}$, are functions of the variables whose indices are in $V_{i,t}$.

Suppose by induction that we have described the behavior of the merging-comparison model up to round t. Thus, at the beginning of round t, each processor $i \in [p]$ knows the variables whose indices are in $V_{i,t}$ and their order type. Consider the collection of access functions $\mathcal{F}|_{S_t}$ consisting of the access functions $R_{i,t}$ and $W_{i,t'}$, for $i \in [p]$ and $0 \leqslant t' \leqslant t$. Since we are at a specific node of the merging-comparison decision tree in which the order types of $V_{i,t}$ are fixed, these access functions are S_t fixed order type functions. By Lemma 4.1, there exists an infinite subset $S_{t+1} \subseteq S_t$, such that the access functions in $\mathcal{F}|_{S_{t+1}}$ are constant. (If a processor does not write, it does not write on all the restricted inputs.)

Let $c := R_{i,t}|_{S_{t+1}}$ be a read access function and let $W_{i',t'}|_{S_{t+1}} = c$ be the write access function (provided it exists), that corresponds to what $R_{i,t}$ actually reads (lexicographically maximal $\langle t', i' \rangle$, corresponding to the most recent write by the highest priority processor). If such $W_{i',t'}$ exists, then processor i reads the value $X_{i',t'}$ which is also a function of $V_{i',t'}$; we can assume that it actually reads $V_{i',t'} \subseteq V_{i',t}$ and computes $X_{i',t'}$ by itself. Define $V'_{i,t}$ to be $V_{i',t}$ if such $W_{i',t'}$ exists, otherwise $V'_{i,t} := \{c'\}$ if memory cell number c initially contains the input variable $x_{c'}$. If neither of these conditions hold, then $V'_{i,t} := \emptyset$. The merging-comparison model merges in round t the set of variables $V_{i,t}$ known by processor $i \in [p]$ with $V'_{i,t}$. This completely defines the behavior of the merging-comparison model in round t. Since the PRAM interprocessor communication pattern is fixed, the state of the processors can depend only on the variables they know.

Suppose the simulating merging-comparison algorithm cannot solve the problem $\mathcal{P}$ in T rounds. Then there are two order types in different $\mathcal{P}$-equivalence classes that it cannot distinguish from each other. Since the domain S_T is large enough, for each processor i, there are input tuples of these two order types that agree on the variables in $V_{i,T}$. But by the invariant maintained, for inputs in S_T, the state of each processor in the original PRAM can only depend on the variables it knows, namely $V_{i,T}$. Hence, the original PRAM cannot distinguish between the inputs either. We conclude:

Theorem 4.2. *If a p processor merging-comparison model requires $\mathcal{M}_{\mathcal{P}}(n, p)$ rounds to solve the problem $\mathcal{P}(x_1, \ldots, x_n)$, then a bounded memory PRAM must take $\mathcal{M}_{\mathcal{P}}(n, p)$ time as well.*

4.3. Unbounded memory, distinct inputs

In this section we assume that the input variables are distinct. As in the previous section, we will construct inductively a merging-comparison decision tree that will simulate the computation of the PRAM.

Suppose by induction that we have described the behavior of the merging-comparison model up to round t and let $\mathcal{F}|_{S_t}$ be the collection of the access functions $R_{i,t}$ and $W_{i,t'}$, for $i \in [p]$ and $0 \leqslant t' \leqslant t$. Since the order types of $V_{i,t}$ are fixed, these access functions are S_t fixed order type functions and by Lemma 4.1, there exists an infinite subset $S_{t+1} \subseteq S_t$, such that the standard forms of the access functions in $\mathcal{F}|_{S_{t+1}}$ are injective and that every pair of functions in $\mathcal{F}|_{S_{t+1}}$ either have identical standard forms or disjoint ranges.

Since the input variables are distinct, identical standard forms of access functions in $\mathcal{F}|_{S_{t+1}}$ are

either functions of exactly the same variables, in which case they are *always* equal, or have *always* disjoint ranges. Hence, given the standard form of each read access function $R_{i,t}|_{S_{t+1}}$ one can determine precisely the unique write access function $W_{i',t'}|_{S_{t+1}}$ that corresponds to what $R_{i,t}$ actually reads. As in the previous section, this defines $V'_{i,t}$ and therefore, the complete behavior of the merging-comparison decision tree. This leads to the following theorem:

Theorem 4.3. *If the p processor merging-comparison model requires $\mathcal{M}_{\mathcal{P}}(n, p)$ rounds to solve the problem $\mathcal{P}(x_1, \ldots, x_n)$ under the distinctness assumption, then a PRAM must take $\mathcal{M}_{\mathcal{P}}(n, p)$ time as well.*

5. Applications

We give some applications of the general lower bounds from the previous section. Some of these bounds were known previously, but we present them in a unified fashion. We also give some new bounds.

In our discussion below we assume that the input has size n and the number of processors p satisfies $1 \leqslant p \leqslant \binom{n}{2}$. Since all the problems we consider depend on most of their input variables, any PRAM algorithm for these problems must read $\Omega(n)$ variables and thus must takes at least $\Omega(n/p)$ time.

5.1. Element distinctness and sorting

The $\Omega(\log_{\lceil p/n+1 \rceil} n)$ comparison model lower bound for sorting [6] gives only an $\Omega(\sqrt{\log n})$ lower bound in the n processor merging-comparison model. However, Boppana [9] gives better direct lower bounds for sorting in the merging-comparison model, from which he derives the PRAM lower bounds that are stated next.

Lemma 5.1 (Boppana [9]). *Sorting in the p processors merging-comparison model requires*

$$\Omega\left(\log_{\lceil (p/n)\log n+1 \rceil} n\right)$$

rounds.

Since the merging-comparison model lower bound holds also under the distinctness assumption, it translates to an $\Omega(\log_{\lceil (p/n)\log n+1 \rceil} n)$ time lower bound for sorting on the PRAM [9]. It is straightforward to establish that sorting comparison model lower bounds hold for the element distinctness problem. Hence, the merging-comparison model lower bound translates to an $\Omega(\log_{\lceil (p/n)\log n+1 \rceil} n)$ time lower bound for the element distinctness problem in the PRAM with bounded memory [9,16].

5.2. Finding the maximum and related problems

Several problems have $\Omega(\log \log_{\lceil p/n+1 \rceil} n)$ rounds lower bounds in the parallel comparison model. The list includes:

- finding the maximum [26];
- merging two lists of equal length [10];
- string-matching [12];
- two-dimensional array-matching [12,14];
- testing if a string is square-free [4] and
- finding initial palindromes in a string [13].

The following lemma, whose proof is similar to Lemma 5.3 below, shows that these lower bounds can be transformed into $\Omega(\log \log_{\lceil p/n+1 \rceil} n)$ lower bounds for the PRAM model.

Lemma 5.2. *The lower bounds listed above are resilient.*

The comparison model lower bounds for finding the maximum and merging hold under the distinctness assumption. Hence, these lower bounds can be transformed into PRAM lower bounds [19,24]. The comparison model lower bounds for string-matching and the related problems mentioned above do not hold under the distinctness assumption, and therefore, these lower bounds translate only to lower bounds in the PRAM with bounded memory.

5.3. Finding an approximate maximum

Alon and Azar [2,3] give tight lower and upper bounds on the comparison complexity of several approximation problems. We consider as an example the problem of finding an approximate maximum

(AM); namely, an element whose rank belongs in the top εn ranks, $1/n \leqslant \varepsilon \leqslant 1/2$. Alon and Azar prove that under the distinctness assumption,

$$\mathscr{C}_{\mathrm{AM}}(n, p) = \Omega\big(\log\log_{\lceil p/n+1\rceil}(1/\varepsilon) + \log^* n - \log^*(p/n)\big). \quad (1)$$

We prove next that this lower bound is resilient. Hence the same lower bound holds in the merging-comparison and the PRAM models.

Lemma 5.3. *The lower bound* (1) *for* $\mathscr{C}_{\mathrm{AM}}(n, p)$ *is resilient.*

Proof. We prove that

$$\log\log_{\lceil p/n+1\rceil}(1/\varepsilon) + \log^* n - \log^*(p/n)$$

is resilient (since constants do not matter). If

$$\log\log_{\lceil p/n+1\rceil}(1/\varepsilon) \leqslant \log^* n - \log^*(p/n) \leqslant \log^* n,$$

then

$$\log^* n - \log^*(p/n) = \Theta\big(\log^* n - \log^*(2^{\log^* n} p/n)\big).$$

If $\log^* n - \log^*(p/n) \leqslant \log\log_{\lceil p/n+1\rceil}(1/\varepsilon) \leqslant \log\log(1/\varepsilon)$, we proceed with two cases. If $1 \leqslant p \leqslant n\log(1/\varepsilon)$, then since $\log^* n \leqslant 1/\varepsilon$, we get that for large enough n,

$$\log\log(1/\varepsilon) \leqslant 2\log\log_{\lceil \log^2(1/\varepsilon)+1\rceil}(1/\varepsilon) \leqslant 2\log\log_{\lceil (p/n)\log(1/\varepsilon)+1\rceil}(1/\varepsilon).$$

If $n\log(1/\varepsilon) \leqslant p \leqslant \binom{n}{2}$, then

$$\log\log_{\lceil (p/n)+1\rceil}(1/\varepsilon) \leqslant \log\log_{\lceil (p/n)\log(1/\varepsilon)+1\rceil}(1/\varepsilon) + 1.$$

Putting the inequalities above together, we establish that

$$\log\log_{\lceil p/n+1\rceil}(1/\varepsilon) = \Theta\big(\log\log_{\lceil (p/n)\log(1/\varepsilon)+1\rceil}(1/\varepsilon)\big). \quad \square$$

6. Conclusions

By using a finite version of the Erdös–Rado Theorem [18], it is possible to replace the infinite Ramsey-theoretic arguments in Section 4.1 by finite ones. However, the lower bounds obtained by the general transformation would still require that the input domain is huge. In some cases, direct lower bounds that were given for specific problems require an input domain that is much smaller [8,16,19]. It would be of interest to extend the lower bounds given here to smaller domains.

In another direction, one could hope to strengthen some of the lower bounds for PRAMs with unbounded memory. However, on a PRAM with unbounded memory (and concurrent-read and concurrent-write with arbitrary write conflict resolution), the following reduction transforms any problem $\mathscr{P}(x_1, \ldots, x_n)$, whose result depends on equality of the variables but not on their order (e.g. element distinctness and string-matching), from an unbounded input domain to a linear sized domain: processors number i, $i \in [n]$, reads input x_i and then writes its index i into memory cell number x_i; next, processor i reads cell x_i and sets y_i to the value read. Note that $y_i \in [n]$ and because the solution to problem $\mathscr{P}$ depends only on the equality between input variables, the solution for $P(y_1, \ldots, y_n)$ is equivalent to the solution for the original input. This observation indicates that the general Ramsey-theoretic techniques of this paper cannot help in obtaining such extensions.

Some of the results reported in this note were originally observed independently in [11] and [15].

Acknowledgement

We thank an anonymous referee for comments that helped improve the presentation above.

References

[1] M. Ajtai, J. Komlós, W.L. Steiger and Szemerédi, Optimal parallel selection has complexity O(log log n), *J. Comput. System Sci.* 38 (1989) 125–133.

[2] N. Alon and Y. Azar, Finding an approximate maximum, *SIAM J. Comput.* 18 (2) (1989) 258–267.

[3] N. Alon and Y. Azar, Parallel comparison algorithms for approximation problems, *Combinatorica* 11 (2) (1991) 97–122.

[4] A. Apostolico and D. Breslauer, An optimal O(log log n) time parallel algorithm for detecting all squares in a string, *SIAM J. Comput.*, to appear.

[5] Y. Azar and N. Pippenger, Parallel selection, *Discrete Appl. Math.* 27 (1990) 49–58.

[6] Y. Azar and U. Vishkin, Tight comparison bounds on the complexity of parallel sorting, *SIAM J. Comput.* 16 (3) (1987) 458–464.

[7] P. Beame and J. Hastad, Optimal bound for decision problems on the CRCW-PRAM, *J. ACM* 36 (3) (1989) 643–670.

[8] O. Berkman, Y. Matias and P. Ragde. Triply-logarithmic upper and lower bounds for minimum, range minima, and related problems with integer inputs, in: *Proc. 3rd Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, Vol. 709 (Springer, Berlin, 1993).

[9] R. Boppana, Optimal separation between concurrent-write parallel machines, in: *Proc. 21st ACM Symp. on Theory of Computing* (1989) 320–326.

[10] A. Borodin and J.E. Hopcroft, Routing, merging and sorting on parallel models of comparison, *J. Comput. System Sci.* 30 (1985) 130–145.

[11] D. Breslauer and D.P. Dubhashi, Tranforming comparison model lower bounds to the parallel-random-access-machine, Tech. Rept. RS-95-10, BRICS, Dept. of Computer Science, University of Aarhus, DK-8000 Aarhus C, Denmark, 1995.

[12] D. Breslauer and Z. Galil, A lower bound for parallel string matching, *SIAM J. Comput.* 21 (5) (1992) 856–862.

[13] D. Breslauer and Z. Galil, Finding all periods and initial palindromes of a string in parallel, *Algorithmica*, to appear.

[14] R. Cole, Z. Galil, R. Hariharan, S. Muthukrishnan and K. Park, Parallel two dimensional witness computation, Manuscript, 1993.

[15] A. Czumaj and F. Meyer auf der Heide, A lower bound for string matching on PRAM, Tech. Rept. tr-rf-95-011, University of Paderborn, 1995.

[16] J. Edmonds, Lower bounds with smaller domain size on concurrent write parallel machines, in: *Proc. 6th Ann. Conf. on Structures in Complexity Theory* (1991) 322–331.

[17] P. Erdös and R. Rado, A combinatorial theorem, *J. London Math. Soc.* 25 (1950) 249–255.

[18] P. Erdös and R. Rado, Combinatorial theorems on classifications of subsets of a given set, *Proc. London Math. Soc.* 2 (1952) 417–439.

[19] F.E. Fich, F. Meyer auf der Heide and A. Wigderson, Lower bounds for parallel random-access machines with unbounded shared memory, *Adv. in Comput. Research* 4 (1987) 1–15.

[20] R.L. Graham, B.L. Rothschild and J.H. Spencer, *Ramsey Theory* (John Wiley & Sons, New York, 2nd ed., 1990).

[21] F. Meyer auf der Heide and A. Wigderson, The complexity of parallel sorting, *SIAM J. Comput.* 16 (1) (1987) 100–107.

[22] S. Moran, M. Snir and U. Manber, Applications of Ramsey's theorem to decision tree complexity, *J. ACM* 32 (4) (1985) 938–949.

[23] P. Ragde, W. Steiger, E. Szemerédi and A. Wigderson, The parallel complexity of element distinctness is $\Omega(\sqrt{\log n})$, *SIAM J. Discrete Math.* 1 (3) (1988) 399–410.

[24] B. Schieber and U. Vishkin, Finding all nearest neighbors for convex polygons in parallel: a new lower bound technique and a matching algorithm, *Discrete Appl. Math.* 29 (1990) 97–111.

[25] Y. Shiloach and U. Vishkin, Finding the maximum, merging and sorting in a parallel computation model, *J. Algorithms* 2 (1981) 88–102.

[26] L.G. Valiant, Parallelism in comparison models, *SIAM J. Comput.* 4 (1975) 348–355.