Approximate and Exact Deterministic Parallel Selection

Shiva Chaudhuri, Torben Hagerup and Rajeev Raman

MPI–I–93–118 May 1993

Approximate and Exact Deterministic Parallel Selection

Shiva Chaudhuri, Torben Hagerup and Rajeev Raman

MPI–I–93–118 May 1993

Approximate and Exact Deterministic Parallel Selection*

Shiva Chaudhuri,[1] Torben Hagerup,[1] and Rajeev Raman[2]

[1] Max-Planck-Institut für Informatik, Im Stadtwald, W–6600 Germany
[2] UMIACS, University of Maryland, College Park, MD 20742

Abstract. The selection problem of size n is, given a set of n elements drawn from an ordered universe and an integer r with $1 \le r \le n$, to identify the rth smallest element in the set. We study approximate and exact selection on deterministic concurrent-read concurrent-write parallel RAMs, where approximate selection with relative accuracy $\lambda > 0$ asks for any element whose true rank differs from r by at most λn. Our main results are: (1) For all $t \ge (\log\log n)^4$, approximate selection problems of size n can be solved in $O(t)$ time with optimal speedup with relative accuracy $2^{-t/(\log\log n)^4}$; no deterministic PRAM algorithm for approximate selection with a running time below $\Theta(\log n/\log\log n)$ was previously known. (2) Exact selection problems of size n can be solved in $O(\log n/\log\log n)$ time with $O(n\log\log n/\log n)$ processors. This running time is the best possible (using only a polynomial number of processors), and the number of processors is optimal for the given running time (optimal speedup); the best previous algorithm achieves optimal speedup with a running time of $O(\log n \log^* n/\log\log n)$.

1 Introduction

Selecting the element of prescribed rank from an ordered (but not sorted) set is an important and well-studied problem. Blum *et al.* [6] showed that selection from a set of size n can be performed in linear time $O(n)$ sequentially. Considerable research has gone into determining the parallel complexity of selection. Valiant [20] introduced the parallel comparison-tree (PCT) model and showed that any deterministic algorithm in the PCT model for finding the maximum of n elements using p processors requires $\Omega(n/p+\log(\log n/\log(1+p/n)))$ time. Since finding the maximum is a special case of selection, this lower bound applies to selection in general as well. Azar and Pippenger [4], building on the work of Ajtai *et al.* [1], gave a matching upper bound for the PCT model. Reischuk [19] showed that in the randomized PCT model, selection can be done in constant expected time using a linear number of processors, which is clearly optimal. The problem of selection in the PCT model has therefore been completely solved. The PCT model counts only comparisons, however, while processing of any other kind is considered free. For this reason, lower bounds obtained for the PCT model apply to all parallel comparison-based algorithms, but upper bounds do not carry over to other, more realistic, models of parallel computation.

In the PRAM model of computation, the upper bounds for the PCT model demonstrably cannot be matched. It follows as a corollary to the lower bound of

* Supported by the ESPRIT Basic Research Actions Program of the EC under contract No. 7141 (project ALCOM II). Authors' email addresses: shiva@mpi-sb.mpg.de, torben@mpi-sb.mpg.de, raman@umiacs.umd.edu.

Beame and Håstad [5] that any (randomized) algorithm that selects the rth smallest among n elements on a p-processor CRCW PRAM has an (expected) running time of $\Omega(\log r/\log\log p)$. In particular, finding the median requires $\Omega(\log n/\log\log n)$ time using any polynomial number of processors. It is not difficult to solve selection problems of size n in (the best possible) time $\Theta(\log n/\log\log n)$ on the CRCW PRAM, but straightforward algorithms for this task use a large number of processors. An important design goal is to get by with as few *operations* as possible, where, as usual, the number of operations executed by a parallel algorithm is defined as the product of the number of processors used and the number of time steps needed by the algorithm. The obvious sequential simulation of any parallel computation shows that the number of operations executed by a parallel algorithm for a given problem is always $\Omega(T)$, where T is the sequential complexity of the problem. A parallel algorithm that uses only $O(T)$ operations is said to have *optimal speedup* or to be *work-optimal*, because it employs the available processors in the most efficient manner possible (up to a constant factor). The development of work-optimal algorithms is one of the most important goals of current research in parallel computation.

In the special case of selection, the result of Blum *et al.* [6] implies that a work-optimal algorithm is one that executes $O(n)$ operations. Cole [9] gave a work-optimal deterministic CRCW PRAM algorithm for selection with a running time of $O(\log n \log^* n/\log\log n)$, and Dietz and Raman [11] recently described a work-optimal algorithm that selects the rth smallest among n elements in $O(\log\log n + \log r/\log\log n)$ time if $1 \le r \le n^{1/3}$, which is the fastest possible for this range of r. The problem of discovering a deterministic CRCW PRAM algorithm for general selection that combines work-optimality with the fastest running time of $O(\log n/\log\log n)$ has remained unsolved, and no progress was made on this front since the publication of Cole's paper.

Attempts have been made to circumvent the lower bounds mentioned above by replacing exact selection by approximate selection. Here, in addition to a target rank $r \in \{1,\dots,n\}$, an *accuracy parameter* $\lambda > 0$ is specified, and the task is to select an element whose rank is guaranteed to lie between $r-\lambda n$ and $r+\lambda n$, which we call *λ-selection for rank r*. Upper and lower bounds for the complexity of approximate selection in the PCT model were given by Alon and Azar [3]. In the PRAM setting, the lower bound of Beame and Håstad does not apply directly to approximate selection, although it can be used to place bounds on the accuracy obtainable with a given amount of resources (i.e., processors and time). Hagerup [13], extending a result of Goodrich [12], showed that approximate selection problems of size n can be solved in constant time with high probability on an n-processor CRCW PRAM for $\lambda = 1/(\log n)^{O(1)}$, which is the best possible accuracy for the stated time and processor bounds. On the other hand, no deterministic PRAM algorithms for approximate selection were previously known.

In this paper we describe deterministic CRCW PRAM algorithms for the problems of approximate and exact selection. Our main result (Corollary 12) is that for all $t \ge (\log\log n)^4$, approximate selection problems of size n can be solved with optimal speedup with relative accuracy $2^{-t/(\log\log n)^4}$. The minimum running time is hence $(\log\log n)^4$, but allowing more time yields a better accuracy, a tradeoff that has been observed before [15, 14, 16]. As an easy corollary of the main result, we derive a work-optimal algorithm for exact selection with a running time of $O(\log n/\log\log n)$ (Theorem 13), thereby solving the open problem left by Cole's paper.

2 Brute-Force Approximate Selection

Without loss of generality we can always assume that the input elements presented to a selection algorithm are distinct. Given an ordered set A, i.e., a subset of an ordered universe U, and an element $x \in U$, the *rank* of x in A, $rank_A(x)$, is defined as $|\{y \in A : y \le x\}|$, and if A is nonempty, the *relative rank* of x in A, $\rho_A(x)$, is defined as $rank_A(x)/|A|$.

If a sufficient number of processors is available, a selection problem can be solved simply by independently computing the rank of each element and returning the unique element whose rank has the desired value. This reduces selection to counting (the number of smaller elements). Approximate selection similarly reduces to approximate counting. The following result, a special case of a theorem due to Hagerup [14, Theorem 6.1], deals with the latter problem.

Lemma 1. *Let $n, t \ge 4$ be given integers with $t \ge (\log\log n)^3$ and take $\lambda = 2^{-t\log\log\log n/(\log\log n)^3}$. Then, given n bits $b_1, \ldots, b_n$, an integer r with $\sum_{i=1}^n b_i \le r \le (1+\lambda)\sum_{i=1}^n b_i$ can be computed using $O(t)$ time and $O(n)$ operations.*

Lemma 2. *Let $n, t \ge 4$ be given integers with $t \ge (\log\log n)^3$ and take $\lambda = 2^{-t\log\log\log n/(\log\log n)^3}$. Then, given an ordered set A of size n and an integer r with $1 \le r \le n$, an element $x \in A$ with $(1-\lambda)r \le rank_A(x) \le (1+\lambda)r$ can be computed using $O(t)$ time and $O(n^2)$ operations.*

Proof. Let $A = \{x_1, \ldots, x_n\}$, compare x_i and x_j for all $i, j \in \{1, \ldots, n\}$ and express the result in the form of an $n \times n$ boolean matrix. Then apply the algorithm of Lemma 1 independently to each row of this matrix to compute integers $r_1, \ldots, r_n$ such that $rank_A(x_i) \le r_i \le (1+\lambda)rank_A(x_i)$, for $i = 1, \ldots, n$, and return any x_i with $r \le r_i \le (1+\lambda)r$.

The relation $r \le r_i \le (1+\lambda)r$ is satisfied for at least one element x_i, namely the one of rank r. On the other hand, it clearly is not satisfied for any element of rank larger than $(1+\lambda)r$, and since $(1+\lambda)(1-\lambda)r < r$, it cannot be satisfied for any element of rank smaller than $(1-\lambda)r$ either. The algorithm is therefore correct. □

3 Sampling

This section studies certain properties of sampling. Some of the arguments parallel ones made in [16].

We allow an element of a set A to be included more than once in a sample of A, i.e., the sample is a *multisubset* of A. Following [16], we define a *ranking function* on a nonempty multiset B as any function $rank_B$ that maps each element x outside of B to the number of elements $y \in B$ with $y \le x$, and that maps B itself bijectively to $\{1, \ldots, |B|\}$ in an order-preserving fashion, i.e., for all $x, y \in B$, if $x < y$, then $rank_B(x) < rank_B(y)$ (in other words, a total order is imposed on B). We also define a *relative ranking function* on B as any function of the form $rank_B/|B|$, where $rank_B$ is a ranking function on B.

Definition 3. Let $\lambda \ge 0$. A nonempty multisubset B of an ordered set A is a λ-*sample* of A if for some relative ranking function ρ_B on B (and therefore for all such functions), $|\rho_A(x) - \rho_B(x)| \le \lambda$, for all $x \in B$.

Lemma 4. *Let n and m be positive integers with m dividing n, let A be an ordered set of size n and let $\lambda \geq 0$. Suppose that x_l is obtained by λ-selecting from A for rank $l \cdot n/m$, for $l = 1, \ldots, m$. Then the multiset $\{x_1, \ldots, x_m\}$ is a λ-sample of A.*

Proof. By construction, $|\rho_A(x_l) - \frac{l}{m}| \leq \lambda$, for $l = 1, \ldots, m$. Let $rank_B$ be an arbitrary ranking function on $B = \{x_1, \ldots, x_m\}$ and fix $i \in \{1, \ldots, m\}$. Taking $j = rank_B(x_i)$, our goal is to show that $|\rho_A(x_i) - \frac{j}{m}| \leq \lambda$. We distinguish between three cases.

Case 1: $i = j$. The claim follows immediately.

Case 2: $i > j$. Since $rank_B$ is a bijection from B to $\{1, \ldots, m\}$, there must be some $l \in \{1, \ldots, j\}$ with $rank_B(x_l) > j$. But then $x_l \geq x_i$ and hence $\rho_A(x_i) - \frac{j}{m} \leq \rho_A(x_l) - \frac{l}{m} \leq \lambda$. On the other hand, $\rho_A(x_i) - \frac{j}{m} \geq \rho_A(x_i) - \frac{i}{m} \geq -\lambda$. The claim follows.

Case 3: $i < j$. Symmetrical to Case 2. □

Lemma 5. *Let $\lambda \geq 0$ and let m be a positive integer. Furthermore let $A_1, \ldots, A_m$ be disjoint sets of a common size, and let $B_1, \ldots, B_m$ be λ-samples of a common size of $A_1, \ldots, A_m$, respectively. Then $B = \bigcup_{i=1}^m B_i$ is a $(\lambda + 1/|B_1|)$-sample of $A = \bigcup_{i=1}^m A_i$.*

Proof. Let $rank_B$ be an arbitrary ranking function on B, and let $rank_{B_i}$ be a ranking function on B_i consistent with $rank_B$, for $i = 1, \ldots, m$, i.e., for all $x, y \in B_i$, $rank_{B_i}(x) < rank_{B_i}(y)$ if and only if $rank_B(x) < rank_B(y)$.

Let $x \in \bigcup_{i=1}^m B_i$, fix $i \in \{1, \ldots, m\}$ and let $r = rank_{B_i}(x)$. If $r < |B_i|$, the rank of x in A_i is no larger than the rank in A_i of the element in B_i of rank $r+1$ in B_i, i.e., no larger than

$$(r+1)\frac{|A_i|}{|B_i|} + \lambda|A_i| = (r+1)\frac{|A|}{|B|} + \lambda|A_i|.$$

This relation obviously also holds if $r = |B_i|$. Summing it for $i = 1, \ldots, m$, we obtain

$$rank_A(x) \leq rank_B(x)\frac{|A|}{|B|} + m\frac{|A|}{|B|} + \lambda|A| = rank_B(x)\frac{|A|}{|B|} + \left(\lambda + \frac{1}{|B_1|}\right)|A|.$$

Arguing similarly, one easily shows that $rank_A(x) \geq rank_B(x)\frac{|A|}{|B|} - \lambda|A|$. □

Lemma 6. *Let $n, k, t \geq 4$ be given integers with $t \geq (\log\log n)^3$ such that $(8k)^2$ divides n and let $\lambda = 2^{-t\log\log\log n/(\log\log n)^3}$. Then, given an ordered set A of size n, a $(\lambda/(12\log\log n) + 1/(8k))$-sample of A of size $n/(8k)$ can be computed using $O(t)$ time and $O(k^3 n)$ operations.*

Proof. Partition A into $n/(8k)^2$ groups of $(8k)^2$ elements each and use the algorithm of Lemma 2 in parallel for each group and for each $i \in \{1, \ldots, 8k\}$ to $(\lambda/(12\log\log n))$-select for rank $i \cdot 8k$. By Lemma 4, this produces a $(\lambda/(12\log\log n))$-sample of each group. Return as the final sample the union of the group samples, which, by Lemma 5, is a $(\lambda/(12\log\log n) + 1/(8k))$-sample of A. □

Lemma 7. *Let $\lambda, \lambda' \geq 0$ and let A be an ordered set. Suppose that B is a λ-sample of A and that C is a λ'-sample of B, where a total order is imposed on B by means of a relative ranking function ρ_B on B (otherwise it makes no sense to speak of a λ'-sample of B). Then C is a $(\lambda + \lambda')$-sample of A.*

Proof. Let ρ_C be a relative ranking function on C consistent with the order on B imposed by ρ_B (i.e., for all $x, y \in C$, if $\rho_B(x) < \rho_B(y)$, then $\rho_C(x) < \rho_C(y)$). Then for all $x \in C$, $|\rho_A(x) - \rho_C(x)| \leq |\rho_A(x) - \rho_B(x)| + |\rho_B(x) - \rho_C(x)| \leq \lambda + \lambda'$. □

4 Work-Optimal Approximate Selection

Our algorithm has the same top-level structure as that of Cole [9]; the details differ. The algorithm consists of two loops, one nested within the other. The outer loop makes progress by eliminating more and more elements from consideration. Since the remaining elements are always those between a lower and an upper limit, similarly as in the case of binary search, the outer loop can be thought of as narrowing down the "search interval". When the search interval has become so small that an arbitrary element in the search interval is an acceptable answer, the algorithm returns such an element and stops.

Each iteration of the outer loop is called a *stage*. A stage works by computing a sample B of the input set A of the current stage (i.e., of the elements in the current search interval) with the property that, on the one hand, B is sufficiently small to make brute-force selection from B according to Lemma 2 feasible and, on the other hand, B represents A faithfully in the sense that the relative rank of an element in B is a good approximation of its relative rank in A. The goal of the algorithm at this point is to select an element from A whose rank in A is within m of r, for some integers m and r. Approximate selection from B is then used to obtain two elements x_L and x_H of A with $x_L \le x_H$ whose ranks are as close as possible, but guaranteed to be on either side of r. An approximation of the number of elements smaller than x_L is then computed and subtracted from r, after which the next stage operates on the elements between x_L and x_H with an absolute accuracy slightly smaller than m (to counteract the inaccuracy incurred in counting the number of elements smaller than x_L).

Computing the sample B is the task of the inner loop, whose single iterations are called *rounds*. The inner loop works by gradually thinning out a sample, which initially is the full input set A and at the end is the final sample B returned to the outer loop. The size of B always lies between $|A|^{1/4}$ and $|A|^{1/2}$, while the quality of B as a sample of A will depend on the processor advantage (number of processors per element) initially available for the computation of B. The processor advantage derives from the progress of the outer loop: When the search interval has narrowed down to the point where it contains only n/k elements, for some $k \ge 1$, the processor advantage available to the inner loop will be essentially k, which allows B to be computed as a $1/k$-sample of A. This in turn allows the search interval to be narrowed down by a factor of roughly k, so that the initial processor advantage available to the inner loop of the next stage will be about k^2. The processor advantage therefore increases doubly-exponentially. If λ is the relative accuracy of the top-level selection, the narrowing process stops when the number of elements in the search interval has dropped to about λn; by the above, this happens after $O(\log\log(1/\lambda))$ stages.

The gradual thinning-out in each stage mentioned above is done by means of repeated subsampling in $O(\log\log n)$ rounds and is characterized in the following lemma.

Lemma 8. *Let n and k be given powers of 2 with $2^6 \le k \le n^{1/8}$, let $t \ge (\log\log n)^4$ be a given integer and take $\lambda = 2^{-t\log\log\log n/(\log\log n)^4}$. Then, given an ordered set A of size n, a $\max\{\lambda, 1/k\}$-sample B of A with $n^{1/4} \le |B| \le n^{1/2}$ can be computed using $O(t)$ time and $O(k^3 n)$ operations.*

Proof. Take $k_1 = k$ and $B_0 = A$ and execute a number of *rounds*. In Round i, for $i = 1, 2, \ldots$, do the following: Spending $O(t/\log\log n)$ time, use the algorithm of Lemma 6 to compute a $(\lambda/(12\log\log n) + 1/(8k_i))$-sample of B_{i-1} of size $|B_{i-1}|/(8k_i)$, call this sample B_i and let k_{i+1} be the largest power of 2 no larger than $\min\{k_i^{4/3}, \sqrt{|B_i|}/8\}$. Take B as the first sample of size $\le n^{1/2}$ encountered in this process.

If for some i we have $|B_i| > n^{1/2}$, but $k_i^{4/3} \ge \sqrt{|B_i|}/8$, we will have $k_{i+1} \ge \sqrt{|B_i|}/16$ and hence $|B_{i+1}| \le 2\sqrt{|B_i|} \le n^{1/2}$, i.e., the $(i+1)$st round will be the last. Let T be the number of rounds executed. By the observation just made, $k_{i+1} \ge k_i^{8/6}/2 \ge k_i^{7/6}$, for $i = 1, \ldots, T-2$ (recall that $k \ge 2^6$), so that $k_i \ge k^{(7/6)^{i-1}}$, for $i = 1, \ldots, T-1$. In particular, $k^{(7/6)^{T-2}} \le n$, which can be shown to imply that $T \le 6\log\log n$.

Since $k_{i+1}^3|B_i| \le k_i^3|B_{i-1}|/8$, for $i = 1, \ldots, T-1$, it is easy to see that the total number of operations executed is indeed $O(k^3 n)$. We have $k_{i+1} \ge 2k_i$, for $i = 1, \ldots, T-2$, and either $k_T \ge 2k_{T-1}$ as well, or $k_T \ge \sqrt{|B_{T-1}|}/16 \ge n^{1/4}/16 \ge k$. In either case $\sum_{i=1}^{T}(1/k_i) \le 3/k$. By Lemma 7, B is a λ'-sample of A, where $\lambda' = \sum_{i=1}^{T}(\lambda/(12\log\log n) + 1/(8k_i)) \le \lambda/2 + 1/(2k) \le \max\{\lambda, 1/k\}$. Finally observe that $|B| = |B_T| \ge \sqrt{|B_{T-1}|} \ge n^{1/4}$. □

The following two lemmas describe the reduction of the "search interval" performed in a single stage of the outer iteration, whereas the outer iteration as a whole is treated in the proof of Theorem 11. To select for rank r with *absolute* accuracy m is to select an element whose rank differs from r by at most m.

Lemma 9. *Let $n, m, t, q \ge 4$ be given integers with $t \ge (\log\log n)^3$, take $\lambda = 2^{-q}$ and $\mu = 2^{-t\log\log\log n/(\log\log n)^3}$ and suppose that $m \ge \mu n$. Then, given an ordered set A of size n and a λ-sample B of A with $n^{1/4} \le |B| \le n^{1/2}$, $O(t)$ time and $O(n)$ operations suffice to reduce the problem of selection from A with absolute accuracy m to that of selection from a set of size at most $10\lambda n$ with absolute accuracy $m - \mu n$.*

Proof. Let $\mu' = 2^{-\lceil t\lceil\log\log\log n\rceil/\lfloor\log\log n\rfloor^3\rceil}$. We can assume without loss of generality that $\mu' n \ge 9(2n^{3/4} + 1)$, since otherwise $t = \Omega(\log n)$, in which case the problem can be solved using Cole's selection algorithm [9].

Assume that the task is to select from A for rank r with absolute accuracy m. Let $\lambda' = \max\{\lambda, \mu'/9\}$ and $r_L = \lfloor(r/n - 2\lambda')|B|\rfloor$. If $r_L < 1$, compute x_L as $\min A$. Otherwise use Lemma 2 to λ'-select from B for rank r_L and let x_L be the resulting element.

We will show that the rank of x_L in A is at most r. Since this is obvious if $x_L = \min A$, assume that this is not the case. Then the rank of x_L in B is at most $r_L + \lambda'|B| \le (r/n - \lambda')|B|$. Since B is a λ'-sample of A, the rank of x_L in A therefore is at most $(r/n - \lambda' + \lambda')|A| = r$.

Similarly, let $r_H = \lceil(r/n + 2\lambda')|B|\rceil$ and obtain x_H by λ'-selecting from B for rank r_H, unless $r_H > |B|$, in which case we take $x_H = \max A$. The rank of x_H in A is at least r.

Next partition A into the sets $A_L = \{x \in A : x < x_L\}$, $A_M = \{x \in A : x_L \le x \le x_H\}$ and $A_H = \{x \in A : x > x_H\}$, i.e., mark each element in A with the set to which it belongs. The rank of x_L in B is at least $\lfloor(r/n - 2\lambda')|B|\rfloor - \lambda'|B| \ge (r/n - 3\lambda' - 1/|B|)|B|$, and the rank of x_L in A therefore is at least $(r/n - 4\lambda' - 1/|B|)|A| \ge$

$r - 4\lambda' n - n^{3/4}$. Similarly, the rank of x_H in A is at most $r + 4\lambda' n + n^{3/4}$, and hence $|A_M| \le 8\lambda' n + 2n^{3/4} + 1 \le 9\lambda' n$.

If $\lambda' = \lambda$, $|A_M| \le 9\lambda n$, and the approximate compaction algorithm of [14] can be used to store the elements of A_M in an array of size at most $10\lambda n$, unused cells of which are filled with dummy elements with a value of ∞. We will show that selecting from A with absolute accuracy m reduces to selecting from A_M with absolute accuracy $m - \mu n$. First the algorithm of Lemma 1 can be used to compute an integer s with $|A_L| \le s \le (1+\mu)|A_L|$. It is now easy to see that if the rank of an element in A_M is within $m - \mu n$ of $r' = r - s$, then its rank in A is within m of r, as desired (in particular, the dummy elements do not change the rank of any real element).

If $\lambda' \ne \lambda$, we have $|A_M| \le 9\lambda' n = \mu' n \le m$, so that the rank in A of any element in A_M is within m of r; we can hence return an arbitrary element of A_M as the answer. □

Lemma 10. *Let n and k be given powers of 2 with $2^6 \le k \le n^{1/8}$ and let t and m be given integers with $t \ge \lceil \log k / \log\log\log n \rceil (\log\log n)^4$ and $m \ge \mu n$, where $\mu = 2^{-t \log\log\log n/(\log\log n)^3}$. Then $O(t)$ time and $O(k^3 n)$ operations suffice to reduce the problem of selection from A with absolute accuracy m to that of selection from a set of size at most $(10/k)n$ with absolute accuracy $m - \mu n$.*

Proof. Let $\lambda = 2^{-t\log\log\log n/(\log\log n)^4}$ and note that $\lambda \le 1/k$. The algorithm of Lemma 8 can be used to compute a $(1/k)$-sample B of A with $n^{1/4} \le |B| \le n^{1/2}$, and the claim can be seen to follow from Lemma 9, used with $q = \log k$. □

Theorem 11. *For all given integers $n \ge 16$ and $t \ge (\log\log n)^4$, approximate selection problems of size n can be solved with relative accuracy λ using $O(t)$ time, $O(n)$ operations and $O(n)$ space, where*

$$\lambda = \begin{cases} 2^{-2^{t/(\log\log n)^4}}, & \text{if } t < (\log\log n)^4 \log^{(4)} n; \\ 2^{-t\log\log\log n/(\log\log n)^4}, & \text{if } t \ge (\log\log n)^4 \log^{(4)} n. \end{cases}$$

Proof. Without loss of generality we can assume that $\lambda \ge 1/n$, since otherwise $t = \Omega(\log n)$, so that the problem can be solved using Cole's algorithm [9], and that n is a power of 2 larger than 160^{32}. Let $m = \lambda n$, so that the problem is to select from a set of size n with absolute accuracy m. As mentioned earlier, the main part of our algorithm consists of T *stages*, for some integer $T \ge 0$. For $i = 1, \ldots, T$, Stage i transforms a problem of selection from a set A_i of size n_i with absolute accuracy m_i to one of selection from a set A_{i+1} of size n_{i+1} with absolute accuracy m_{i+1}, where n_{i+1} is significantly smaller than n_i, while m_{i+1} is slightly smaller than m_i, and both n_i and n_{i+1} are powers of 2. In particular, $n_1 = n$ and $m_1 = m$. T is the smallest nonnegative integer such that $n_{T+1} \le m_{T+1}$ or $n_{T+1}^2 \le n$.

We now describe Stage i, for $i \in \{1, \ldots, T\}$. Let k_i be the smallest power of 2 larger than $40\lceil (n/n_i)^{1/16} \rceil$ and take $t_i = \lceil \log k_i / \lfloor \log\log\log n \rfloor \rceil \lceil \log\log n \rceil^4$. For $i = 1, \ldots, T$, $n_i > n^{1/2}$ and hence $k_i \le 160 n^{1/32} \le n^{1/16} \le n_i^{1/8}$. We will show below that $m_i \ge \mu n_i$, where $\mu = 2^{-t\log\log\log n/(\log\log n)^4 - 2}$. Spending $O(t_i + t/\log\log n)$ time, we can therefore use the algorithm of Lemma 10 to reduce the problem at hand to one of selection from a set A_{i+1} of size at most n_{i+1} with absolute accuracy $m_{i+1} = m_i - \mu n_i$, where n_{i+1} is the smallest power of 2 larger than $(10/k_i)n_i$. We increase the size of

A_{i+1} to exactly n_{i+1} by adding a suitable number of dummy elements with a value of ∞ and note that $n_{i+1} \le (20/k_i)n_i$.

For $i = 1, \dots, T-1$,

$$\frac{n}{n_{i+1}} \ge \frac{n}{(20/k_i)n_i} \ge \frac{2n(n/n_i)^{1/16}}{n_i} = 2\left(\frac{n}{n_i}\right)^{17/16}.$$

In particular, $n_{i+1} \le n_i/2$, for $i = 1, \dots, T-1$, so that $\sum_{i=1}^{T} n_i \le 2n$. Noting that $\mu \le \lambda/4$, we will now prove that $m_i \ge \mu n_i$, for $i = 1, \dots, T$, as claimed above. Since $m = \lambda n \ge 4\mu n$, it suffices to show that $m_T \ge m/2$. But $m - m_T = \mu \sum_{i=1}^{T-1} n_i \le 2\mu n \le m/2$, as required.

After Stage T, the computation is finished in one of two ways. If $n_{T+1} \le m_{T+1}$, we simply return an arbitrary element of A_{T+1}, which is obviously correct. Otherwise, $n_{T+1}^2 \le n$, and we use the algorithm of Lemma 2 to select from A_{T+1} with absolute accuracy $m/2$, for which $O(t)$ time amply suffices. Since $m_T \ge m/2$, this is also correct.

It remains to estimate the time and the number of operations needed by the algorithm. We begin by bounding the number of stages. Since $n/n_{i+1} = \Theta((n/n_i)^{17/16})$, we have $k_{i+1} = \Theta((n/n_{i+1})^{1/16}) = \Theta((n/n_i)^{17/16^2}) = \Theta(k_i^{17/16})$, for $i = 1, \dots, T-1$. Now if $T \ge 2$, $(20/k_{T-1})n_{T-1} \ge n_T \ge m/2$ and hence $k_{T-1} \le 40n_{T-1}/m \le 40/\lambda$, which implies that $T = O(\log\log(4/\lambda))$.

The processing after Stage T obviously uses $O(t)$ time and $O(n)$ operations. The total number of operations executed in Stages 1 through T is

$$O\Big(\sum_{i=1}^{T} k_i^3 n_i\Big) = O\Big(\sum_{i=1}^{T} (n/n_i)^{3/16} n_i\Big) = O\Big(n \sum_{i=1}^{T} (n_i/n)^{13/16}\Big) = O(n),$$

where the last relation follows from the rapid growth of the sequence $\{n/n_i\}_{i=1}^{T}$. The time needed by Stages 1 through T, finally, is

$$O\Big(\sum_{i=1}^{T}\Big(t_i + \frac{t}{\log\log n}\Big)\Big) = O\Big(\frac{tT}{\log\log n} + \sum_{i=1}^{T}\Big\lceil \frac{\log k_i}{\log\log\log n}\Big\rceil (\log\log n)^4\Big)$$

$$= O\Big(t + \Big(T + \frac{\sum_{i=1}^{T}\log k_i}{\log\log\log n}\Big)(\log\log n)^4\Big) = O\Big(t + \Big(T + \frac{\log k_T}{\log\log\log n}\Big)(\log\log n)^4\Big)$$

$$= O(t + (\log\log(4/\lambda) + \log(4/\lambda)/\log\log\log n)(\log\log n)^4).$$

A case analysis shows that the latter expression is $O(t)$. $\square$

Corollary 12. *For all given integers $n \ge 4$ and $t \ge (\log\log n)^4$, approximate selection problems of size n can be solved with relative accuracy $2^{-t/(\log\log n)^4}$ using $O(t)$ time and $O(n)$ operations. In particular, for any constant $\lambda > 0$, λ-selection problems of size n can be solved in $O((\log\log n)^4)$ time with optimal speedup.*

5 Work-Optimal Exact Selection

Armed with a reasonably accurate work-optimal algorithm for approximate selection, it is an easy matter to derive a work-optimal algorithm for exact selection. The algorithm is similar to one stage of the algorithm of Theorem 11. We now provide the details.

Theorem 13. *For all integers $n \geq 4$, selection problems of size n can be solved using $O(\log n/\log\log n)$ time, $O(n)$ operations and $O(n)$ space.*

Proof. Suppose that the task is to select the element of rank r from a set A. Applied with $t = \Theta(\log n/\log\log n)$, the algorithm of Corollary 12 allows us to compute two elements $x_L, x_H \in A$ with $rank_A(x_L) \leq r \leq rank_A(x_H)$, but $rank_A(x_H) - rank_A(x_L) = O(n \cdot 2^{-\sqrt{\log n}})$. Next partition A into the sets $A_L = \{x \in A : x < x_L\}$, $A_M = \{x \in A : x_L \leq x \leq x_H\}$ and $A_H = \{x \in A : x > x_H\}$ as in the proof of Lemma 9. Then use exact prefix summation [10, Theorem 2.2.2] to determine $|A_L|$ and to store the elements of A_M in an array of size $|A_M| = O(n \cdot 2^{-\sqrt{\log n}})$. The remaining problem is to select the element of rank $r - |A_L|$ from A_M, which trivially reduces to sorting A_M. The latter task can be accomplished using the algorithm of Cole [8, Theorem 3], which, given the available processor advantage of $2^{\Theta(\sqrt{\log n})}$, runs in $O(\log n/\log\log n)$ time. $\square$

6 Increasing the Accuracy

At the price of increasing the number of operations slightly, we can obtain a better accuracy than that provided by Theorem 11. Specifically, in $O(t)$ time, we achieve a relative accuracy of 2^{-t}, as compared to the approximately $2^{-t\log\log\log\log n/(\log\log n)^4}$ of Theorem 11. Our algorithm is based on the *AKS sorting network* [2] and further improvements and expositions of it [17, 18, 7]. In a manner first used in [1], we simulate the initial stages of the operation of the network in order to focus on an interesting subset of the elements.

For an input of size n, where n is a power of 2, the AKS network can be thought of as moving the elements between the nodes of a complete binary tree with n leaves numbered $1, \ldots, n$ from left to right. Say that an element is *addressed* to a node u in the tree if its rank is the number of a leaf descendant of u, and that it is a *stranger* at u otherwise.

The analysis of the AKS network in [7] can be shown to imply the following: For every integer i with $1 \leq i \leq \log n$, there is a positive integer T with $T = O(i)$, computable from i in constant time, such that after T stages of the operation of the network and for any node u at depth i in the tree, the number of keys at u is nonzero, but the fraction of strangers among these is at most 2^{-7}.

Let $n \geq 4$ be a power of 2 and let q be a positive integer. In order to use the above to λ-select from n elements, where $\lambda = 2^{-q}$, we proceed as follows: Choose the positive integer i minimal such that there is a node u at depth i with the property that every element addressed to u is a valid answer to the selection problem; it is not difficult to see that $i = O(q)$. Then compute T from i, run the AKS network for T stages and finally use the algorithm of Corollary 12 to select an element from the middle half of the elements stored at u. Since the strangers at u occupy the extreme ranks of relative size at most 2^{-7}, this produces an element addressed to u, and hence a correct answer. The time needed is $O(T + (\log\log n)^4) = O(q + (\log\log n)^4)$. We hence have

Theorem 14. *For all given integers $n \geq 4$ and $q \geq 1$, approximate selection problems of size n can be solved with relative accuracy 2^{-q} using $O(q+(\log\log n)^4)$ time, $O(qn)$ operations and $O(n)$ space.*

References

1. M. Ajtai, J. Komlós, W. L. Steiger, and E. Szemerédi. Optimal parallel selection has complexity $O(\log\log N)$. *Journal of Computer and System Sciences*, **38** (1989), pp. 125–133.
2. M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proc. 15th ACM STOC* (1983), pp. 1–9.
3. N. Alon and Y. Azar. Parallel Comparison Algorithms for Approximation Problems. *Combinatorica*, **11** (1991), pp. 97–122.
4. Y. Azar and N. Pippenger. Parallel selection. *Discrete Applied Mathematics*, **27** (1990), pp. 49–58.
5. P. Beame and J. Håstad. Optimal bounds for decision problems on the CRCW PRAM. *Journal of the ACM*, **36** (1989), pp. 643–670.
6. M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, **7** (1973), pp. 448–461.
7. V. Chvatal. Lecture notes on the new AKS sorting network. *DIMACS Technical Report 92–29*, 1992.
8. R. Cole. Parallel merge sort. *SIAM Journal on Computing*, **17** (1988), pp. 770–785.
9. R. Cole. An optimally efficient selection algorithm. *Information Processing Letters*, **26** (1988), pp. 295–299.
10. R. Cole and U. Vishkin. Faster optimal parallel prefix sums and list ranking. *Information and Computation*, **81** (1989), pp. 334–352.
11. P. F. Dietz and R. Raman. Heap construction on the CRCW PRAM. In preparation, 1993.
12. M. T. Goodrich. Using approximation algorithms to design parallel algorithms that may ignore processor allocation. In *Proc. 32nd IEEE FOCS* (1991), pp. 711–722.
13. T. Hagerup. The log-star revolution. In *Proc. 9th STACS* (1992), LNCS 577, pp. 259–278.
14. T. Hagerup. Fast deterministic processor allocation. In *Proc. 4th ACM-SIAM SODA* (1993), pp. 1–10.
15. T. Hagerup and R. Raman. Waste makes haste: Tight bounds for loose parallel sorting. In *Proc. 33rd IEEE FOCS* (1992), pp. 628–637.
16. T. Hagerup and R. Raman. Fast deterministic approximate and exact parallel sorting. In *Proc. 5th ACM SPAA* (1993), to appear.
17. M. S. Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, **5** (1990), pp. 75–92.
18. N. Pippenger. Communication Networks. In *Handbook of Theoretical Computer Science, Vol A, Algorithms and Complexity* (J. van Leeuwen, ed.). Elsevier/The MIT Press (1990), Chapter 15, pp. 805–833.
19. R. Reischuk. Probabilistic parallel algorithms for sorting and selection. *SIAM Journal on Computing*, **14** (1985), pp. 396–409.
20. L. G. Valiant. Parallelism in comparison problems. *SIAM Journal on Computing*, **4** (1975), pp. 348–355.