A TIME-RANDOMNESS TRADEOFF FOR SELECTION IN PARALLEL

Danny Krizanc

TR-218, FEBRUARY 1993

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

A Time-Randomness Tradeoff for Selection in Parallel

Danny Krizanc
School of Computer Science
Carleton University
Ottawa, Ontario K1S 5B6

Abstract

In this paper we study the problem finding the median on a parallel comparison tree (PCT). Results due to Valiant [Va], Meggido [Me] and Reishcuk [Re] show a provable gap between the randomized and deterministic parallel complexity of selection. We prove a tight tradeoff between the amount of randomness used by an algorithm for this problem and its performance, measured by the time it requires to complete its computation with a given failure probability. The tradeoff provides a smooth bridge between the deterministic and randomized complexity of the problem.

1 Introduction

The use of randomness in computation is well established (see Rabin [Rab]). It has been applied in sequential, distributed and parallel computation with great success, in many cases providing efficient solutions where no deterministic solutions are known and in some cases where comparably efficient deterministic solutions are provably impossible. To help explain this phenomena we propose to consider the randomness an algorithm uses as a resource and study its effect on the efficiency of the algorithm. We prove a tight tradeoff between the amount of randomness used by an algorithm and its performance, measured by the time it requires to complete its computation

with a given failure probability. Such a tradeoff was first shown by Krizanc et al. [KPU, PU] for the problem of oblivious routing on hypercubic networks. A number of other researchers have also studied the effect of limiting randomness in computation [Mu, KR, Ran].

In this paper we consider the problem of selection in parallel. Valiant [Va] showed a $\Omega(\log\log n)$ deterministic lower bound for the problem of finding the median of n elements on an n processor parallel computation tree (PCT). Reischuk [Re], and independently Meggido [Me], proved the existence of a gap between the randomized and deterministic complexities of the problem by providing a n processor randomized PCT which selects the median of n elements in $O(1)$ time with high probability.

In section 3, we demonstrate how to convert Valiant's lower bound into a tradeoff between the amount of randomness required by a PCT to find the median, its run-time and its probability of failure. In section 4 we extend Reischuk's algorithm (using results due to Pippenger [Pi] to show the existence of randomized PCT strategies matching the bounds given in Section 3. The techniques used for both the lower and upper bounds are analogous to those used by Krizanc et al. and may be applicable to other problems.

2 Preliminaries

Valiant [Va] introduced the parallel computation tree (PCT) model for studying parallelism in the classical comparison problems of maximum, median, merging and sorting. The input for each problem is a set of n elements on which a linear ordering is defined. The basic operation available to processors is the comparison of two elements. With k processors, k comparisons may be performed simultaneously in one step. Depending on which of the 2^k possible results is attained, the next set of k comparisons is chosen. The computation ends when sufficient information is discovered about the relationships of the elements to specify the solution to the given problem. The *deterministic time complexity* of a problem in this model is the number of steps required for the worst case input or the minimum depth of a tree solving the problem, as a function of the size of the input set and the number of processors used.

The model is easily extended to allow random computations. In the randomized parallel computation tree (RPCT)model, at each step we introduce a probability distribution over the choice of which k comparisons are to be

performed. In this case, the *randomized time complexity* of a RPCT is the expected number of steps required on the worst case input.

We say a RPCT is *uniform* if all its random choices are generated by a fixed number of independent calls to the same random process having a finite number of outcomes (e.g., coin flips). Consider the problem, called *k-selection*, of selecting the kth largest element of an n element set. This includes the problems of finding the maximum ($k = 1$) and median ($k = \frac{n}{2}$). We

Claim: Any p processor, uniform RPCT strategy for k-selection is equivalent to a strategy $A = \langle S, D \rangle$ where

1. *$S = \{s_1, s_2, \ldots\}$, a set of p processor, deterministic PCT strategies for k-selection,*
2. *$D = \{p_1, p_2, \ldots\}$, a probability distribution over S with $Prob(s_i) = p_i$,*
3. *the strategy used on input permutation π is: Choose $s \in S$ with distribution D and solve π using strategy s.*

Furthermore the number of independent calls to the random process required in the worst case is the same for both strategies and therefore the number of independent random bits required in the worst case is also the same.

Proof: Let d be the maximum number of calls the given uniform RPCT makes to its underlying random process on all possible runs. Let $x_1 \cdots x_d$ be a string of d possible outcomes of the random process. Each such string defines a (not necessarily distinct) deterministic PCT strategy, $s(x_1 \cdots x_d)$, where the random choices of the uniform RPCT are governed by the outcomes of the string. Let $p(x_i)$ be the probability of x_i occurring as an outcome of the random process. Let $p(x_1 \cdots x_d) = p(x_1) \cdot \ldots \cdot p(x_d)$ be the probability of a string of d independent outcomes being $x_1 \cdots x_d$. Then the strategy having $s(x_1 \cdots x_d)$ occurring with probability $p(x_1 \cdots x_d)$ is of the required form and exactly simulates the given uniform RPCT. Furthermore the number of independent calls to the random process in the worst case is d for both strategies. □

Let $A = \langle S, D \rangle$ be a uniform RPCT strategy solving the k-selection problem. Let m_π^i be the number of steps required by the deterministic PCT, s_i, on the input π of size n. We say

$$T_q^A \leq t \Leftrightarrow \forall \pi \sum_{\{i | m_\pi^i \geq t\}} p_i \leq q.$$

Thus a strategy, A, has complexity $T_q^A \leq t$, if for all permutations it solves the k-selection problem within time t with probability greater than $1 - q$. We denote by B^A the worst case number of independent random bits the strategy requires. The following fact is easily derived:

Fact 1 $B^A \geq \max_i \{-log\, p_i \mid p_i \neq 0\}$.□

3 A Lower Bound for k-Selection

The main result of this section is a relationship between a time-processor tradeoff for k-selection and its corresponding time-randomness tradeoff. Let $t_{\mathbf{ksel}}(p, n)$ be the minimum over all p processor, deterministic PCT strategies for k-selection of the number of steps required by a PCT on its worst case input of size n. We prove

Theorem 1 *Let $0 < q < 1$ and let $r \geq 1$. Let $A = \langle S, D \rangle$ be a p processor, uniform RPCT strategy for k-selection with inputs of size n. Say $t_{\mathbf{ksel}}(rp, n) \geq t$. If $T_q^A < t$ then $B^A \geq \log r - logq$.*

Proof: From any subset, S', of S of size r, we can form an rp processor, deterministic PCT strategy for k-selection by running in parallel each of the strategies in S' and terminating whenever one of the strategies terminates. Since $t_{\mathbf{ksel}}(rp, n) \geq t$, there exists a permutation taking time greater than t for this strategy. Therefore each of the strategies in S' must take at least time t on this permutation. This implies any subset of S of size r or less must have probability totaling less than q. Therefore there must be at least $\frac{r}{q}$ strategies in S with positive probability. One of these strategies must have probability less than $\frac{q}{r}$. The result now follows from fact 1. □

Corollary 1 *Let $0 < q < 1$, let $r \geq 1$. Let $A = \langle S, D \rangle$ be a p processor, uniform RPCT strategy for finding the kth largest element of n elements. For some $c = \Theta(1)$ we have*

1. *If $1 < pr < n$ then if $T_q^A < \frac{n}{pr} + \log\log pr - c$ then $B^A \geq \log r - \log q$;*
2. *If $4 \leq 2n \leq pr \leq \frac{n(n-1)}{2}$ then if $T_q^A < \log\log n - \log\log \frac{pr}{n} - c$ then $B^A \geq \log r - \log q$.*

Proof: The results follow immediately from the bounds on finding the maximum of n elements with pr processors, $t_{1\text{sel}}(pr, n)$, given in [Va] and the fact that finding the maximum of n elements is reducible to finding the kth of $n + k$ elements. □

We are particularly interested in the case of PCT's using n processors:

Corollary 2 *Let $0 < q < 1$ and let $\frac{1}{\log n} \leq \epsilon \leq 1$. Let $A = \langle S, D \rangle$ be an n processor, uniform RPCT strategy for finding the kth largest of n elements. For some $c = \Theta(1)$ we have, if $T_q^A < \log(\frac{1}{\epsilon}) - c$ then $B^A \geq \epsilon \log n - \log q$.* □

An examination of the proofs of theorem 1 and the lower bound in [KPU, PU] suggests that they both follow from the same general principle: one can trade randomness for processors. This is especially evident in theorem 1. If we keep the error probability constant, by adding one bit of randomness we get the same time lower bound for half the number of processors.

4 Upper Bounds for k-Selection

In this section, combining results of Pippenger on deterministic selection [Pi] and Reischuk on randomized selection [Re] we prove the existence of uniform RPCTs matching the bounds given by corollary 2.

The following fact is due to Pippenger [Pi]:

Fact 2 *Let $\frac{1}{\log n} \leq \epsilon \leq 1$. There exists a deterministic PCT strategy using $n^{1+\epsilon}$ processors solving the k-selection problem for inputs of size n in time $O(\log(\frac{1}{\epsilon}) + c)$ where $c = \Theta(1)$.* □

Below we present a modified deterministic version of Reischuk's n processor selection algorithm which we call (k, ϵ)-SELECT. Let $X = \{x_1, \ldots, x_n\}$ be an n element set. Let k be the rank of the element of X to be selected and let $\frac{1}{\log n} \leq \epsilon \leq 1$.
Procedure (k, ϵ)-SELECT

1. Let $X' = \{x_1, \ldots, x_{n^{1-\frac{\epsilon}{6}}}\}$;
2. Define

$$t_1 = \max\{0, \lfloor k\frac{n^{1-\frac{\epsilon}{6}} + 1}{n+1} - n^{1-\frac{\epsilon}{3}} \rfloor\},$$

$$t_2 = \min\{n+1, \lceil k\frac{n^{1-\frac{\epsilon}{6}}+1}{n+1} + n^{1-\frac{\epsilon}{3}}\rceil\};$$

3. Select s_1 equal to the element of rank t_1 of X';
4. Select s_2 equal to the element of rank t_2 of X';
5. Compare every element of X with s_1;
6. Compare every element of X with s_2;
7. Define
$$A = \{x \in X | x < s_1\},$$
$$B = \{x \in X | s_1 \leq x \leq s_2\},$$
$$C = \{x \in X | s_2 < x\};$$
8. (a) If $|A| < k$ and $|A| + |B| \geq k$ then let $Y = B$ and $k' = k - |A|$,
   (b) If $|A| \geq k$ then let $Y = A$ and $k' = k$,
   (c) If $|A| + |B| < k$ then let $Y = C$ and $k' = k - |A| - |B|$;
9. Select x equal to the k'th element of Y;
10. Output x;

End Procedure.

The correctness of the procedure is easily verified. Using Reischuk's probabilistic analysis we are able to prove

Lemma 1 *Let* $\frac{1}{\log n} \leq \epsilon \leq 1$. *On an input of size* n, *chosen uniformly at random, the algorithm* (k, ϵ)*-SELECT runs in time* $O(\log(\frac{1}{\epsilon}) + c)$ *with probability greater than* $1 - \exp\{-\frac{1}{4}n^{1-\frac{\epsilon}{2}} + O(\log n)\}$, *where* $c = \Theta(1)$.

Proof: Note that as far as the analysis is concerned, choosing a sample of size $n^{1-\frac{\epsilon}{6}}$ at random is equivalent to choosing the first $n^{1-\frac{\epsilon}{6}}$ elements of a randomly chosen set.

Using the algorithm of fact 2, steps 3 and 4 take $O(\log(\frac{1}{\epsilon})+c)$ time. Steps 5 and 6 take constant time with n processors. Following Reischuk's analysis [Re] we have the probability that $|Y| > n^{1-\frac{\epsilon}{6}}$ is less than $\exp\{-\frac{1}{4}n^{1-\frac{\epsilon}{2}} + O(\log n)\}$. Therefore, with probability greater than $1-\exp\{-\frac{1}{4}n^{1-\frac{\epsilon}{2}}+O(\log n)\}$, step 9 requires $O(\log(\frac{1}{\epsilon}) + c)$ time. □

Theorem 2 *Let* $2^{\frac{\log(5\log n)}{\log n}} \leq \epsilon \leq 1$ *and* $\frac{1}{n} \leq q \leq \frac{1}{2}$. *For the problem of selecting the kth of n elements there exists a uniform RPCT strategy,* $A = \langle S, U \rangle$, *with* $B^A = \epsilon \log n - \log q$ *for which* $T_q^A = O(\log(\frac{1}{\epsilon}) + c)$, *where* $c = \Theta(1)$ *and* U *is the uniform distribution assigning equal probability to each strategy in* S.

Proof: For each permutation, π, define an n processor deterministic PCT strategy s_π as follows: Perform permutation π on the input and then run (k, ϵ)-SELECT on the resulting permutation. We show that a randomly chosen set of such strategies works with non-zero probability and therefore a set, S, with the desired property exists.

We say a permutation π_1 *succeeds for* an input π_0 *in time* t if s_{π_1} runs in time t on π_0. Lemma 1 is equivalent to the statement: For a fixed permutation, π_0, a randomly chosen permutation, π_1, succeeds for π_0 in time $O(\log(\frac{1}{\epsilon}) + c)$ with probability greater than $1 - \exp\{-\frac{1}{4}n^{1-\frac{\epsilon}{2}} + O(\log n)\}$.
Claim: *Let* $\Pi = \{\pi_1, \ldots, \pi_{\frac{n^\epsilon}{q}}\}$ *be a random set of* $\frac{n^\epsilon}{q}$ *deterministic PCT strategies. Let* E_Π *be the event, for all permutations* π,

$$|\{s_{\pi_i} \in \Pi | \pi_i \textit{ fails (does not succeed) for } \pi \textit{ in } O(\log(\tfrac{1}{\epsilon}) + c)\}| < q|\Pi|.$$

Then $Prob(E_\Pi) > 0$.
Proof: We bound the probability of the event, $\bar{E}_\Pi$, there exists a permutation π, such that

$$|\{s_{\pi_i} \in \Pi | \pi_i \text{ fails for } \pi \text{ in time } O(\log(\tfrac{1}{\epsilon}) + c)\}| \geq q|\Pi|.$$

For a given permutation π, the probability that more than $q|\Pi|$ of the random permutations in Π fail for π is less than

$$\left(\frac{1}{e^{\frac{1}{4}n^{1-\frac{\epsilon}{2}} - O(\log n)}}\right)^{n^\epsilon} \cdot \binom{\frac{n^\epsilon}{q}}{n^\epsilon} < \frac{2^{\frac{n^\epsilon}{q}}}{e^{\frac{1}{4}n^{1+\frac{\epsilon}{2}} - n^\epsilon O(\log n)}} < \frac{1}{n!} \quad (q \geq n^{-1}, \epsilon \geq 2\frac{\log(5\log n)}{\log n}).$$

Thus the probability there exists some permutation, among the $n!$ possible permutations, for which more than $q|\Pi|$ of the strategies in Π fail in time $O(\log(\frac{1}{\epsilon}) + c)$ is strictly less than

$$n! \cdot \frac{1}{n!} = 1.$$

That is, $\text{Prob}(E_\Pi) = 1 - \text{Prob}(\bar{E}_\Pi) > 0$.□

From the claim we may conclude that there exists a Π for which E_Π holds. This Π may be used to define A satisfying the conditions of the theorem. □

5 Conclusions

Comparing theorem 2 with corollary 2 we see that there exist uniform RPCT strategies using an optimal number of random bits and running in time optimal to within an additive constant. However, the upper bound presented here is nonconstructive. A result of Karloff and Raghavan [KR] implies a constructive scheme for k-selection, A, with $B^A = O(\log n)$ and $T^A_{n^{-c}} = O(1)$. Their result (actually dealing with sorting on a PRAM) uses simple pseudo-random number generators. It does not immediately answer the problem of constructing a uniform RPCT strategy for k-selection matching the bound given in theorem 2.

We commented earlier that the lower bound in this paper and that in [KPU] follow from the same general principle: one can trade processors for randomness. Similarly the (nonconstructive) upper bounds each case take the same form: reduce the amount of randomness by choosing from a subset of all objects rather than the full set. It would be interesting to find other examples of time-randomness tradeoffs in parallel computation and/or find general conditions under which such tradeoffs exist.

Acknowledgements. The author would like to thank Les Valiant, Eli Upfal, David Peleg, Allan Borodin, Mihaly Gereb and Thanasis Tsantilas for helpful discussions.

References

[KR] H. J. KARLOFF AND P. RAGHAVAN, *Randomized Algorithms and Pseudorandom Numbers,* Proc. of 20th ACM Symp. on Theory of Computing, 1988, pp. 310-321.

[KPU] D. KRIZANC, D. PELEG AND E. UPFAL, *A Time-Randomness Tradeoff for Oblivious Routing,* Proc. of 20th ACM Symp. on Theory of Computing, 1988, pp. 93-102.

[Me] N. MEGGIDO, *Parallel Algorithms for Finding the Maximum and the Median Almost Surely in Constant-time,* Carnegie-Mellon University, Oct. 1982.

[Mu] K. MULMULEY, *Randomized Geometric Algorithms and Pseudo-Random Generators,* Proc. of 33rd Symp. on Foundations of Computer Science, 1992, pp. 90-100.

[PU] D. PELEG AND E. UPFAL, *A Time-Randomness Tradeoff for Oblivious Routing,* SIAM J. of Computing **20**, (1989), pp. 396-409.

[Pi] N. PIPPENGER, *Sorting and Selecting in Rounds,* SIAM J. of Computing **16**, (1987), pp. 1032-1038.

[Rab] M. O. RABIN, *Probabilistic Algorithms,* in Algorithms and Complexity, Recent Results and New Directions, (J. F. Traub, Ed.), Academic Press, New York, (1976), pp. 21-40.

[Ran] A. G. RANADE, *Constrained Randomization for Parallel Communication,* Yale Univeristy Technical Report TR-511, 1987.

[Re] R. REISCHUK, *Probabilistic Parallel Algorithms for Sorting and Selection,* SIAM J. of Computing **14**, (1985), pp. 396-409.

[Va] L. G. VALIANT, *Parallelism in Comparison Problems,* SIAM J. of Computing **4**, (1975), pp. 348-355.

School of Computer Science, Carleton University
Recent Technical Reports

TR-178 **Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric**
R.S. Valiveti and B.J. Oommen, July 1990

TR-179 **Parallel Algorithms for Determining K-width-Connectivity in Binary Images**
Frank Dehne and Susanne E. Hambrusch, September 1990

TR-180 **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

TR-181 **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

TR-182 **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

TR-183 **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

TR-184 **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990

TR-185 **Uniform Generation of Combinatorial Objects in Parallel**
M.D. Atkinson and J.-R. Sack, January 1991

TR-186 **Reduced Constants for Simple Cycle Graph Separation**
Hristo N. Djidjev and Shankar M. Venkatesan, February 1991

TR-187 **Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer**
Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991

TR-188 **Generating and Sorting Jordan Sequences**
Alan Knight and Jörg-Rüdiger Sack, March 1991

TR-189 **Probabilistic Estimation of Damage from Fire Spread**
Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991

TR-190 **Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects**
Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991

TR-191 **Towards Decomposable, Reusable Smalltalk Windows**
Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991

TR-192 **PARASOL: A Simulator for Distributed and/or Parallel Systems**
John E. Neilson, May 1991

TR-193 **Realizing a Spatial Topological Data Model in a Relational Database Management System**
Ekow J. Otoo and M.M. Allam, August 1991

TR-194 **String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations**
B John Oommen, September 1991

TR-195 **The Expressiveness of Silence: Optimal Algorithms for Synchronous Communication of Information**
Una-May O'Reilly and Nicola Santoro, October 1991

TR-196 **Lights, Walls and Bricks**
J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia and J. Zaks, October 1991

TR-197 **A Brief Survey of Art Gallery Problems in Integer Lattice Systems**
Evangelos Kranakis and Michel Pocchiola, November 1991

TR-198 **On Reconfigurability of Systolic Arrays**
Amiya Nayak, Nicola Santoro, and Richard Tan, November 1991

TR-199 **Constrained Tree Editing**
B. John Oommen and William Lee, December 1991

TR-200 **Industry and Academic Links in Local Economic Development: A Tale of Two Cities**
Helen Lawton Smith and Michael Atkinson, January 1992

TR-201 **Computational Geometry on Analog Neural Circuits**
Frank Dehne, Boris Flach, Jörg-Rüdiger Sack, Natana Valiveti, January 1992

TR-202 **Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays**
Amiya Nayak, Linda Pagli, Nicola Santoro, February 1992

TR-203 **Numeric Similarity and Dissimilarity Measures Between Two Trees**
B. John Oommen and William Lee, February 1992

TR-204 **Recognition of Catastrophic Faults in Reconfigurable Arrays with Arbitrary Link Redundancy**
Amiya Nayak, Linda Pagli, Nicola Santoro, March 1992

TR-205 **The Permutational Power of a Priority Queue**
M.D. Atkinson and Murali Thiyagarajah, April 1992

TR-206 **Enumeration Problems Relating to Dirichlet's Theorem**
Evangelos Kranakis and Michel Pocchiola, April 1992

TR-207 **Distributed Computing on Anonymous Hypercubes with Faulty Components**
Evangelos Kranakis and Nicola Santoro, April 1992

TR-208 **Fast Learning Automaton-Based Image Examination and Retrieval**
B. John Oommen and Chris Fothergill, June 1992

TR-209 **On Generating Random Intervals and Hyperrectangles**
Luc Devroye, Peter Epstein and Jörg-Rüdiger Sack, July 1992

TR-210 **Sorting Permutations with Networks of Stacks**
M.D. Atkinson, August 1992

TR-211 **Generating Triangulations at Random**
Peter Epstein and Jörg-Rüdiger Sack, August 1992

TR-212 **Algorithms for Asymptotically Optimal Contained Rectangles and Triangles**
Evangelos Kranakis and Emran Rafique, September 1992

TR-213 **Parallel Algorithms for Rectilinear Link Distance Problems**
Andrzej Lingas, Anil Maheshwari and Jörg-Rüdiger Sack, September 1992

TR-214 **Camera Placement in Integer Lattices**
Evangelos Kranakis and Michel Pocchiola, October 1992

TR-215 **Labeled Versus Unlabeled Distributed Cayley Networks**
Evangelos Kranakis and Danny Krizanc, November 1992

TR-216 **Scalable Parallel Geometric Algorithms for Coarse Grained Multicomputers**
Frank Dehne, Andreas Fabri and Andrew Rau-Chaplin, November 1992

TR-217 **Indexing on Spherical Surfaces Using Semi-Quadcodes**
Ekow J. Otoo and Hongwen Zhu, December 1992

TR-218 **A Time-Randomness Tradeoff for Selection in Parallel**
Danny Krizanc, February 1993

TR-219 **Three Algorithms for Selection on the Reconfigurable Mesh**
Dipak Pravin Doctor and Danny Krizanc, February 1993