# A Survey of Randomness and Parallelism in Comparison Problems

Danny Krizanc *

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

**Abstract.** A survey of results for the problems of selection, merging and sorting in the Randomized Parallel Comparison Tree (RPCT) model is given. The results indicate that while randomization "helps" in the case of selection, it does not provide any advantage for the cases of merging and sorting, in this model.

## 1 Introduction

Valiant [27] introduced the Parallel Comparison Tree (PCT) model for studying parallelism in the classical comparison problems of selection, 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 $p$ processors, $p$ comparisons may be performed simultaneously in one step. Depending on which of the $2^p$ possible results is attained, the next set of $p$ 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 worst-case *deterministic* 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. We note that in this model we do not consider any of the overheads, such as processor communication, memory accesses, etc., associated with performing the comparisons and making the appropriate deductions from the results of the comparisons. However, in cases where the cost of comparisons dominates the computation, upper bounds in this model can often be translated into upper bounds in more restricted models and in all cases where algorithms base their decisions solely upon comparisons, lower bounds in this model translate to lower bounds in these other models.

The model is easily extended to allow random computations. In the Randomized PCT (RPCT) model, at each step we introduce a probability distribution over the choice of which $p$ comparisons are to be performed. In this case, the worst-case *randomized* complexity of a RPCT is the expected number of steps

* Research was supported in part by the Natural Sciences and Engineering Research Council of Canada.

required on the worst case input. The *deterministic (randomized) average-case* complexity is the expected number of steps required by a PCT (RPCT) on the uniform distribution of the inputs.

It is easy to see that the deterministic complexity is greater than or equal to the randomized complexity which in turn is greater than or equal to the average-case complexity. In particular, this implies that upper bounds on the deterministic complexity imply upper bounds on all three measures and, similarly, lower bounds on the average-case complexity imply lower bounds on all three measures. Since, as was observed by Yao [28], any randomized PCT can be thought of as a probability distribution over deterministic PCTs, the randomized average-case complexity is equal to the deterministic average-case complexity. Therefore it is sufficient to consider only the deterministic average-case complexity below. Also note that since any comparison-based problem can be solved in one step on a PCT with $p \geq \binom{n}{2}$ on inputs of size $n$, we consider only the case where $1 \leq p \leq \binom{n}{2}$.

A parallel algorithm is said to achieve *deterministic (average-case) optimal speed up* if its deterministic (average-case) complexity is proportional to $\frac{Seq(n)}{p}$, where $Seq(n)$ is a lower bound on the worst-case (average-case) deterministic serial running time, $n$ is the size of the problem being considered and $p$ is the number of processors used. We note that if a $p$ processor algorithm achieves (deterministic or average-case) optimal speed up then by a straightforward "slowing down" argument there exists a $p'$ processor optimal speed up algorithm for any $p' < p$. This motivates the following definitions. The *deterministic (average-case) break point* for a comparison-based problem is the minimum deterministic (average-case) complexity achieved by an deterministic (average-case) optimal speed up algorithm for the problem.

Below, we consider each of the problems of selection, merging and sorting in turn. Note that all logarithms are to the base two.

## 2 Selection

The problem of PCT selection has three parameters: the number of processors, $p$, the size of the input set, $n$, and the rank of the element to be selected, $k$. Three cases are of special interest: the minimum ($k = 1$), the maximum ($k = n$) and the median ($k = \lceil \frac{n}{2} \rceil$). By introducing extra $-\infty$ and $+\infty$ elements it is easy to see that the problem of selecting the element of rank $k$ out of $n$ elements can be reduced to that of selecting the median of at most $2n$ elements and therefore we will concentrate on this (two parameter) case in what follows. Let $T(p, n)$ be the worst-case time to select the median of $n$ elements using a $p$ processor PCT and let $\hat{T}(p, n)$ be the worst-case expected time to select the median of $n$ elements using a $p$ processor RPCT.

### 2.1 Deterministic Lower Bound

Valiant [27] showed a deterministic lower bound for selecting the maximum which by the reduction mentioned above and together with a deterministic upper bound shown by Azar and Pippenger [7] (building on the work of Ajtai et al. [2]) implies the following:

**Theorem 1** $T(p,n) = \Theta(n/p + \log(\log n/\log(2+p/n)))$.

### 2.2 Randomized Upper Bound

Meggido [22] and independently Reischuk [26] showed that the above lower bound could be "beaten" using randomization by providing an optimal RPCT algorithm for selection:

**Theorem 2** $\hat{T}(p,n) = \Theta(n/p + 1)$.

The proof of the above result relies on well understood sampling techniques introduced in the context of sequential selection by Floyd and Rivest [13] and subsequently exploited in a variety of parallel models for both selection and sorting. (See for example [11, 16, 17, 23, 24, 25, 26].)

### 2.3 Discussion

The results above show that there is a significant gap between the randomized and deterministic parallel complexity of selection in the PCT model. In particular we see that the deterministic break point for selection occurs at $\Theta(\log\log n)$ while the average-case break point occurs at $\Theta(1)$. A partial explanation of this phenomenon is given in [18] where a tight tradeoff between the amount of randomness used by a RPCT for selection and its performance, measured by the time it requires to complete its computation with a given failure probability, is shown. While the asymptotic complexity of finding the maximum is the same as that of finding the median it is interesting to note that when considering two round deterministic PCTs, Häggkvist and Hell [15] have shown that $O(n^{4/3})$ processors are sufficient to find the maximum of $n$ elements while Alon and Azar [3] have proven that $\Omega(n^{4/3}\log^{2/3} n)$ processors are necessary to find the median. We also note that an unresolved gap exists between the (deterministic or randomized) complexity of finding the maximum and finding the median on the mesh (see [11, 19, 20]).

## 3 Merging

The problem of merging has three parameters: the number of processors, $p$, and the sizes of the two sorted lists to merged, $n$ and $m$. Below we concentrate on the canonical (two parameter) special case where $m = n$. Let $M(p,n)$ be the worst-case time to merge two $n$ element lists using a $p$ processor PCT and let $\bar{M}(p,n)$ be the average-case time for the same problem.

### 3.1 Deterministic Upper Bound

The following deterministic upper bound for merging follows from results of Valiant [27] and Kruskal [21]. A lower bound matching their upper bound was shown by Borodin and Hopcroft [10].

**Theorem 3** $M(p, n) = \Theta(n/p + \log(\log n/\log(2 + p/n)))$.

### 3.2 Average-Case Lower Bound

The deterministic lower bound of Borodin and Hopcroft [10] was extended to the average-case by Geréb-Graus and Krizanc [14]:

**Theorem 4** $\bar{M}(p, n) = \Theta(n/p + \log(\log n/\log(2 + p/n)))$.

### 3.3 Discussion

The results above show that the deterministic break point and average-case break point for merging both occur at $\Theta(\log\log n)$. That is, unlike the case of selection, randomization does not help. The results for merging can be extended (with similar conclusions) to the case where $m < n$ [14] and to the case of merging more than two ordered lists [6]. Borodin and Hopcroft [10] were able to show that the upper bound of Valiant [27] can be implemented in the more restrictive PRAM model.

## 4 Sorting

The problem of sorting has two parameters: the number of processors, $p$, and the size of the input set to be sorted, $n$. Let $S(p, n)$ be the worst-case time to sort an $n$ element set using a $p$ processor PCT and let $\bar{S}(p, n)$ be the average-case time for the same problem.

### 4.1 Deterministic Upper Bound

The upper bound for the case $p \leq n$ was established by Ajtai et al. [1]. This was extended by Alon, Azar and Vishkin [5, 8] who showed the following tight result:

**Theorem 5** $S(p, n) = \Theta(\log n/\log(1 + p/n))$.

### 4.2 Average-Case Lower Bound

The matching average-case lower bound for sorting was first shown by Alon and Azar [4]. A significantly simpler proof of the same result was provided by Boppana [9].

**Theorem 6** $\bar{S}(p, n) = \Theta(\log n/\log(1 + p/n))$.

### 4.3 Discussion

The results above show that sorting is similar to merging in that the deterministic and average-case break points are equal, in this case occurring at $\Theta(\log n)$. Again we must conclude that randomization does not provide any improvement in the parallel complexity of sorting in the PCT model.

## 5 Conclusions

The main conclusion of the above discussion is that while randomization makes a provable difference in the parallel complexity of the problem of selection, it provably does not effect the asymptotic parallel complexity of either merging or sorting, in the PCT model. An explanation of why this distinction, between selection on the one hand and merging and sorting on the other, occurs, would be of great interest. There are other models in which a gap between the parallel and randomized complexity of a comparison-based problem exists. Examples include sorting on the hypercube [12, 25] and selection on the mesh [11, 20]. But in these cases the gap in complexity represents a gap in the current state of our knowledge of the problem, not a provable difference in the deterministic and randomized complexities of the problem.

## References

1. M. Ajtai, J. Komlós and E. Szemerédi, *An $O(N \log N)$ Sorting Network,* Proc. of 15th ACM Symp. on Theory of Computing, 1983, pp. 1-9.
2. M. Ajtai, J. Komlós, W. L. Steiger, and E. Szemerédi, *Optimal Parallel Selection Has Complexity $O(\log\log N)$* , J. of Computer and System Sciences, **38** (1989), pp. 125-133.
3. N. Alon and Y. Azar, *Sorting, Approximate Sorting and Searching in Rounds,* SIAM J. of Discrete Mathematics, **1** (1988), pp. 269-280.
4. N. Alon and Y. Azar, *The Average Complexity of Deterministic and Randomized Parallel Comparison Sorting Algorithms,* SIAM J. of Computing, **17** (1988), pp. 1178-1192.
5. N. Alon, Y. Azar and U. Vishkin, *Tight Complexity Bounds for Parallel Comparison Sorting,* Proc. of 29th IEEE Symp. on Foundations of Computer Science, 1986, pp. 502-510.
6. Y. Azar, *Parallel Comparison Merging of Many-Ordered Lists,* Theoretical Computer Science, **83** (1991), pp. 275-285.
7. Y. Azar and N. Pippenger, *Parallel Selection,* Discrete Applied Mathematics, **27** (1990), pp. 49-58.
8. Y. Azar and U. Vishkin, *Tight Comparison Bounds on the Complexity of Parallel Sorting,* SIAM J. of Computing, **16** (1987), pp. 458-464.
9. R. Boppana, *The Average-Case Parallel Complexity of Sorting,* Information Processing Letters, **33** (1989), pp. 145-146.
10. A. Borodin and J. E. Hopcroft, *Routing, Merging and Sorting on Parallel Models of Computation,* J. Computer and System Sciences, **30** (1985), pp. 130-145.

11. A. Condon and L. Narayanan, *Upper and Lower Bounds for Selection on the Mesh,* Proc. of Symp. on Parallel and Distributed Processing, 1994, pp. 497-504.
12. R.Cypher and G. Plaxton, *Deterministic Sorting in Nearly Logarithmic Time on the Hypercube and Related Computers,* Proc. of 22nd ACM Symp. on Theory of Computing, 1990, pp. 193-203.
13. R. Floyd and R. Rivest, *Expected Time Bounds for Selection,* Communications of the ACM, **18** (1975), pp. 165-172.
14. M. Geréb-Graus and D. Krizanc, *The Average Complexity of Parallel Comparison Merging,* SIAM J. of Computing, **21** (1992), pp. 43-47.
15. R. Häggkvist and P. Hell, *Graphs and Parallel Comparison Algorithms,* Congr. Numer., **29** (1980), pp. 497-509.
16. C. Kaklamanis and D. Krizanc, *Optimal Sorting on Mesh-Connected Processor Arrays,* Proc. of the 4th ACM Symp. on Parallel Algorithms and Architectures, 1992, pp. 50-59.
17. C. Kaklamanis, D. Krizanc, L. Narayanan and T. Tsantilas, *Randomized Sorting and Selection on Mesh-Connected Processor Arrays,* Proc. of the 3rd ACM Symp. on Parallel Algorithms and Architectures, 1991, pp. 17-28.
18. D. Krizanc, *Time-Randomness Tradeoffs in Parallel Computation,* Journal of Algorithms, **20** (1996), pp. 1-19.
19. D. Krizanc and L. Narayanan, *Optimal Algorithms for Selection on a Mesh-Connected Processor Array,* Proc. of IEEE Symp. on Parallel and Distributed Processing, 1992, pp. 70-76.
20. D. Krizanc, L. Narayanan and R. Raman, *Fast Deterministic Selection on a Mesh-Connected Processor Array,* Algorithmica, **15** (1996), pp. 319-332.
21. C. P. Kruskal, *Searching, Merging and Sorting in Parallel Computation,* IEEE Trans. on Computers, **C-32** (1983), pp. 942-946.
22. N. Meggido, *Parallel Algorithms for Finding the Maximum and the Median Almost Surely in Constant-time,* Carnegie-Mellon University, Oct. 1982.
23. S. Rajasekaran, *Randomized Parallel Selection,* Proc. of Foundations of Software Technology and Theoretical Computer Science Conf., 1990, pp. 215-224.
24. S. Rajasekaran, *Sorting and Selection on Interconnection Networks,* DIMACS Series on Discrete Mathematics and Theoretical Computer Science, **21** (1995), pp. 275-296.
25. J. Reif and L. Valiant, *A Logarithmic Time Sort for Linear Size Networks,* J. of the ACM, **34** (1987), pp. 60-76.
26. R. Reischuk, *Probabilistic Parallel Algorithms for Sorting and Selection,* SIAM J. of Computing, **14** (1985), pp. 396-409.
27. L. G. Valiant, *Parallelism in Comparison Problems,* SIAM J. of Computing, **4** (1975), pp. 348-355.
28. A. C-C. Yao, *Probabilistic Computations: Towards a Unified Measure of Complexity,* Proc. of 18th Symp. on Foundations of Computer Science, 1977, pp. 222-227.