PARALLEL ALGORITHMS FOR FINDING THE MAXIMUM AND THE MEDIAN ALMOST SURELY IN CONSTANT-TIME

Nimrod Megiddo*

Abstract. It is shown that the maximum and the median of a set of n elements can be found by n processors in $O(1)$ time with probability approaching 1 as n tends to infinity. The maximum-finding algorithm runs on the WRAM model whereas the median-finding algorithm is in the parallel computation tree model. These models are of course strong but, as we have shown in a previous paper, are very useful for designing good serial algorithms based on parallel ones. Our parallel probabilistic algorithms in this paper yield interesting serial probabilistic algorithms.

*Carnegie-Mellon University, GSIA, Pittsburgh, PA 15213

1. Introduction

We will discuss here probabilistic algorithms that make no errors, i.e., algorithms which involve random moves and yet always terminate with the correct outcome. The time-complexity is measured by the expected time that it takes to solve the worst-case instance. Thus, the expectation is defined relative to the internal randomization of the algorithm. In fact, for the algorithms presented here we can make stronger claims regarding the (random) running time in the worst-case. The format will be the following: The probability $p(n)$ that the algorithm terminates within $t(n)$ time units tends to 1 as n approaches infinity. Moreover, $t(n)$ will be constant!

To our knowledge, there is no good example in serial computation where the option of randomization leads to a significantly better time bound, given that the correct outcome should always be produced. It is thus interesting to find that randomization is very helpful in parallel computation. This is demonstrated in two cases: Finding the maximum and the median. Our main motivation comes from [M1] where we showed how to use good parallel algorithms in the design of serial algorithms. Here we introduce a new feature which lets us produce interesting probabilistic algorithms from parallel probabilistic ones. Applications of the results of this paper are given in a companion paper [M2] where a dynamic computational geometric problem is solved with the aid of our algorithm for the maximum. An application along the same lines was also given by Zemel [Z].

Our first example is of finding the maximum. It has neen proved by L. Valiant [V] that $\log\log n$ comparisons are required for n processors in order to find the maximum in a set of n elements. Valiant also gave an $O(\log\log n)$ algorithm in the strong model of comparisons (later called the parallel computation tree [BH]), which was later extended by Shiloach and

Vishkin [SV] to a weaker model. It is interesting to note that the exact maximum can be found by a randomizing algorithm in expected $O(1)$ time on n processors, in the latter model.

The second example deals with the general selection problem. An open question in parallel computation is whether selection is substantially easier than sorting. The question was raised by Valiant [V] in the context of the parallel computation tree model. It is not known how to deterministically sort a set of n elements on n processors in $O(\log n)$ time. On the other hand an expected number of $\Omega(\log n)$ comparisons is required for sorting on n processors. Surprisingly, it is possible to exactly solve the selection problem in $O(1)$ expected time in the parallel computation tree model. So, this is another case where randomization helps significantly.

2. Finding the Maximum

We first let each of $\sqrt{n}$ processors sample an element of the set. Since this is done in parallel, some of the processors may sample the same elements so this is like "sampling with replacement." The expected number of distinct elements in the sample is $\sqrt{n} - o(\sqrt{n})$.

In the next step we find the maximum of the sample, deterministically in constant-time, by comparing every pair of sample elements. The details of the implementation of this step in the WRAM model (requiring concurrent reads and writes) are in [SV].

Given the sample-maximum, we now compare each element of the set with this maximum and "discard" the ones that are smaller. This is done in constant-time (subject to concurrent read). If the number of the remaining elements is smaller than $\sqrt{n}$ then it takes one step of comparisons to find the maximum.

(See [SV] for the implementation). Otherwise, we again sample $\sqrt{n}$ elements, determine their maximum, and so on.

We will now prove that the probability, that we will be done after two sampling steps, is approaching 1 when n tends to infinity.

Let $X_1,\ldots,X_s$ be independent random variables uniformly distributed over $[0,1]$, where $s = \lfloor\sqrt{n}\rfloor$. Let $Y_i = \lceil nX_i \rceil$ $(i = 1,\ldots,s)$ so that Y_i represents the rank of the element sampled by X_i. Let $X_{(1)} \leq \ldots \leq X_{(s)}$ denote the order statistics of $\{X_1,\ldots,X_s\}$ and let $Y_{(1)} \leq \ldots \leq Y_{(s)}$ be the order statistics of $\{Y_1,\ldots,Y_s\}$. It is well-known (see [W, p. 236]) that $X_{(s)}$ has the density function $f(x) = sx^{s-1}$ so that its mean is $\frac{s}{s+1}$ and its variance is $\frac{s}{(s+1)^2(s+2)}$. Asymptotically, the mean of $Y_{(s)}$ is hence $n - \sqrt{n}$ and its standard deviation is $n \cdot \sqrt{\frac{\sqrt{n}}{(\sqrt{n}+1)^2(\sqrt{n}+2)}} \sim \sqrt{n}$. Thus, the probability that $Y_{(s)} > n-n^{1/2+\varepsilon}$ approaches 1 for every $\varepsilon > 0$. Equivalently, the number of remaining elements after the first step is $o(n^{1/2+\varepsilon})$, with probability approaching 1, for every $\varepsilon > 0$. Now, consider the second step. Using the same notation, let now $Y_i = \lceil n^{1/2+\varepsilon} \cdot X_i \rceil$. Thus Y_i is the rank of the element sampled by X_i (during the second step), relative to the remaining set, assuming this set has $n^{1/2+\varepsilon}$ elements. Now the mean of $Y_{(s)}$ is asymptotically equal to

$$n^{1/2+\varepsilon} - \frac{n^{1/2+\varepsilon}}{\sqrt{n}+1}.$$

Thus the mean of the number of elements remaining after the second step is asymptotically n^{ε}. Obviously, the actual number is less than $n^{1/2}$, with probability approaching 1.

3. Finding the Median

The algorithm for the median is in the parallel computation tree model, i.e., we count only comparisons as time consuming operations. The algorithm is related to that of [FR] for serial selection.

Like in the case of the maximum, we start by sampling $\sqrt{n}$ elements. Next, we select from the sample two elements, namely, the ones whose ranks relative to the sample are $\frac{1}{2}\sqrt{n} \pm n^{\varepsilon}$, where $\varepsilon > 0$ is a number to be determined later.

Now, given the two elements, it takes two rounds of comparisons to find their ranks relative to the input set, merely by comparing them with every element. Then, it can be decided which elements may be discarded and what is the rank of the median relative to the remainder of the set. The remainder consists of the elements that lie between the two we have selected, with probability approaching 1.

Assume that we are left with m elements, and we need to find our median, which has a rank of r relative to the remainder. If $m < \sqrt{n}$ then we can terminate in one step of comparisons. Otherwise, we again sample $\sqrt{n}$ elements from the remainder and select the ones whose ranks, relative to the sample, are $\frac{r}{m} \cdot \sqrt{n} \pm n^{\varepsilon}$. The process continues in this way until we are left with less than $\sqrt{n}$ elements.

We will now prove that the probability, that we will be done after three sampling steps, is approaching 1 when n tends to infinity, provided $\frac{1}{4} < \varepsilon < \frac{1}{3}$.

Let X_i, Y_i, $X_{(i)}$ and $Y_{(i)}$ be as above. The variable $X_{(k)}$ has the beta distribution $\beta(k, s-k+1)$ [W, p. 236]. As such, its mean equals $\frac{k}{s+1}$ while its variance is equal to $\frac{k(s-k+1)}{(s+1)^2(s+2)}$ (see [W, p. 174]). If $s = \sqrt{n}$

and $k = \alpha\sqrt{n} - n^{\varepsilon}$, then the mean is $\frac{\alpha\sqrt{n}-n^{\varepsilon}}{\sqrt{n}+1} = \alpha - n^{\varepsilon-1/2} + O(\frac{1}{\sqrt{n}})$. This implies that the expected rank of the element sampled by $X_{(k)}$ (i.e., the mean of $Y_{(k)}$) is $\alpha n - n^{1/2+\varepsilon} + O(\sqrt{n})$. Similarly, the variance of $X_{(k)}$ is equal to

$$\frac{(\alpha\sqrt{n}-n^{\varepsilon})(\sqrt{n} - \alpha\sqrt{n}+n^{\varepsilon}+1)}{(\sqrt{n}+1)^2(\sqrt{n}+2)} \sim \frac{\alpha(1-\alpha)}{\sqrt{n}} .$$

Thus, the standard deviation of $X_{(k)}$ is asymptotically equal to $\sqrt{\alpha(1 - \alpha)}\cdot n^{-1/4}$, so that the standard deviation of $Y_{(k)}$ is asymptotically $\sqrt{\alpha(1 - \alpha)}\cdot n^{3/4}$. It thus follows that if $\varepsilon > \frac{1}{4}$, then the probability, that $Y_{(k)}$ is less than αn, is approaching 1. Analogously, for $\ell = \alpha\sqrt{n} + n^{\varepsilon}$, the probability that $Y_{(\ell)}$ is greater than αn is approaching 1. It thus follows that, with probability approaching 1, the (αn)-th order statistic of the set is between the elements sampled by $X_{(k)}$ and $X_{(\ell)}$.

We will now consider the number of remaining elements, given that we are left with the middle range.

Consider the difference $X_{(\ell)} - X_{(k)}$. This random variable also has the beta distribution (see [W, p. 238])

$$\beta(\ell - k, s - \ell + k + 1).$$

Its mean equals $\frac{\ell-k}{s+1}$, which in our case is $\frac{2n^{\varepsilon}}{\sqrt{n}+1} \sim 2n^{\varepsilon-1/2}$. The corresponding number of input elements is asymptotically $2n^{1/2+\varepsilon}$. The variance of $X_{(\ell)} - X_{(k)}$ is equal to

$$\frac{(\ell-k)(s-\ell+k+1)}{(s+1)^2(s+2)} = \frac{2n^{\varepsilon}(\sqrt{n}-2n^{\varepsilon}+1)}{(\sqrt{n}+1)^2(\sqrt{n}+2)} \sim 2n^{\varepsilon-1} .$$

The standard deviation of $X_{(\ell)} - X_{(n)}$ is asymptotically $\sqrt{2}\cdot n^{-1/2+\varepsilon/2}$ so that the standard deviation of the corresponding number of elements is

asymptotically $\sqrt{2}\ n^{1/2+\epsilon/2}$. Thus, with probability approaching 1, the number of remaining elements after the first sampling step is $O(n^{1/2+\epsilon})$. The second sampling step will leave us (with probability approaching 1) with only $2n^{\epsilon-1/2} \cdot O(n^{1/2+\epsilon}) = O(n^{2\epsilon})$ elements. Recall that this holds for $\epsilon > \frac{1}{4}$. Now, a third sampling stop will leave us (with probability approaching 1) with $2n^{\epsilon-1/2} \cdot O(n^{2\epsilon}) = O(n^{3\epsilon-1/2})$ elements. Thus, if $\frac{1}{4} < \epsilon < \frac{1}{3}$, the third step will leave us with less than $\sqrt{n}$ elements, with probability approaching 1. In other words, we will almost surely not need more than three sampling steps.

4. Conclusion

The main difference between the two algorithms is due to the fact that the maximum is of course always not smaller than the sample-maximum whereas the position of the median relative to the sample-median is not known in advance, and requires counting the number of "winners" in a round of comparisons. This is essentially why we cannot run our median-finding algorithm in $O(1)$ time in other models. However, the power of randomization in parallel computation is surprising and the potential with respect to probabilistic serial algorithms is very promising.

Acknowledgments

Discussions with W. F. Eddy, L. Rudolph and U. Vishkin are gratefully acknowledged.

References

[BH] A. Borodin and J. E. Hopcroft, "Routing, Merging and Sorting on Parallel Models of Computation," Proceedings of the 14th Annual ACM Symposium on Theory of Computing (1982) 338-344.

[FR] R. W. Floyd and R. L. Rivest, "Expected Time Bounds for Selection," Communications of the ACM 18 (1975) 165-172.

[M1] N. Megiddo, "Applying Parallel Computation Algorithms in the Design of Serial Algorithms," Proceedings of the 22nd IEEE Symposium on Foundations of Computer Science (1981), 399-408.

[M2] N. Megiddo, "Poly-log Parallel Algorithms for LP with an Application to Exploding Flying Objects," preliminary report, November 1982.

[SV] Y. Shiloach and U. Vishkin, "Finding the Maximum, Merging and Sorting in a Parallel Computation Model," J. Algorithms 2 (1981) 88-102.

[V] L. G. Valiant, "Parallelism in Comparison Problems," SIAM J. Comput. 4 (1975) 348-355.

[W] S. M. Wilks, Mathematical Statistics, John Wiley & Sons, New York, 1962.

[Z] E. Zemel, "Randomized Parallel Algorithms for Selection with Applications," Working paper, November 1982.