# Research Article <br> The Middle Pivot Element Algorithm 

Anchala Kumari ${ }^{1}$ and Soubhik Chakraborty ${ }^{2}$<br>${ }^{1}$ Department of Statistics, Patna University, Patna 800005, India<br>${ }^{2}$ Department of Applied Mathematics, Birla Institute of Technology (BIT Mesra), Ranchi 835215, India<br>Correspondence should be addressed to Soubhik Chakraborty, soubhikc@yahoo.co.in

Received 7 November 2012; Accepted 29 November 2012
Academic Editors: P. Amodio, R. López-Ruiz, and Q.-W. Wang
Copyright © 2012 A. Kumari and S. Chakraborty. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper is an improvement over the previous work on New Sorting Algorithm first proposed by Sundararajan and Chakraborty (2007). Here we have taken the pivot element as the middle element of the array. We call this improved version Middle Pivot Element Algorithm (MPA) and it is found that MPA is much faster than the two algorithms RPA (Random Pivot element Algorithm) and FPA (First Pivot element Algorithm) in which the pivot element was selected either randomly or as the first element, respectively.

## 1. Introduction

One of the computational problems that an algorithm encounters is due to the multiple input parameters that has a direct effect on the execution time of the algorithm and we call this problem as one of parameterized complexity of the algorithm. In the recent past much of the work has been done to simulate the parameterized complexity of an algorithm under different situations. The present work is an improvement on the previous works by Sundararajan and Chakraborty [1] in which the New Sorting Algorithm was introduced. In order to make this paper self-contained, we are providing it again.

Step 1. Initialize the first element of the array as a pivot element.

Step 2. Starting from the second element, compare it to the pivot element.

Substep 1. If pivot < element then place the element in the last unfilled position of the temporary array (of same size as the original one).

Substep 2. If pivot $\geq$ element then place the element in the first unfilled position of the temporary array.

Step 3. Repeat Step 2 till the last element of the array.
Step 4. Finally place the pivot element in the blank position of the temporary array. (Remark: the blank position is created because one element of the original array was taken out as pivot).

Step 5. Split the array into two, based on the pivot element's position.

Step 6. Repeat Steps 1-5 till the array is sorted completely.
Prashant et al. [2], Anchala and Chakraborty [3, 4], and a recent unpublished paper by the present authors entitled "Parameterized Complexity: A Statistical Approach Combining Factorial Experiments with Principal Component Analysis." are some of the related works. This last paper compares the parameterized complexity of two algorithms called the RPA (Random Pivot element Algorithm) and FPA (First Pivot element algorithm) in respect to the new sorting technique developed by Sundararajan and Chakraborty [1]. In this paper we have taken the pivot element as the middle term of the array in new sorting technique and call this algorithm MPA (Middle Pivot element algorithm). Thus Step 1 in MPA should be read as follows.

Table 1: Factors and their levels.

| Factor | level |  |  |
| :--- | :---: | :---: | :---: |
| $n$ | 10000 | 15000 | 20000 |
| $m$ | 100 | 150 | 200 |
| $P$ | 0.2 | 0.5 | 0.8 |



Figure 1

Step 1. Initialize the middle element of the array as a pivot element.

The rest of the steps are the same as above.
The paper reveals two interesting features: (i) MPA is much faster than RPA and FPA and (ii) MPA is more efficient in sorting a large number of elements than what can be sorted by RPA and FPA. See Mahmoud [5] for a comprehensive literature on sorting with special emphasis on distribution theory.

## 2. Statistical Analysis of the MPA Algorithm

A 3-cube factorial experiment is employed to examine the singular and nonsingular effects of binomial inputs on the complexity. The three factors are (i) $n$, the numbers to be sorted, and (ii) $m$, and (iii) $P$, the two parameters of the binomial $(m, P)$ distribution. The three factors and their levels are given in Table 1.

Five runs for each of the 27 treatment combinations were made and execution time was obtained using Visual C++ code.

A 3-cube factorial experiment was conducted and analyzed using MINITAB 16 and the results are given below in Table 2. As there are five runs, so there will be $r=5$ replicates with $5-1=4$ degrees of freedom. The significant points of the variance ratio $F$ statistic are compared in Table 3.

The interesting and important points to be noted from Table 4 are that irrespective of any value of $n, m$, and $P$ the sorting time is much smaller for MPA as compared to the other two algorithms RPA and FPA. In other words the New


Figure 2

Regression plot
$p$-time $=0.857262+0.469262 n$
$+0.479762 P^{* *} 2$


Figure 3

Sorting technique with pivot element as the middle element is faster and should be preferred.

A systematic pattern is observed for changing the binomial parameters [2]. For fixed $n$ and fixed $P$, the sorting time shows a decreasing trend in response to an increase in $m$. While for fixed $n$ and $m$, the sorting time is minimum at $P=0.5$. At $P=0.2$, the sorting time is less than or equal to the same at $P=0.8$; while in the other two algorithms, for fixed $n$, the sorting time responds in a very erratic way, for example, when we change $P$ keeping $m$ fixed or change $m$ keeping $P$ fixed in RPA. On the other hand, in FPA for fixed $n$ and fixed $m$, the sorting time shows almost decreasing trend while the probability points are increased. However, for fixed $n$ and $P$, no systematic pattern is found in the time value corresponding to the change in the value of $m$.

Now we study the behavior of the three factors separately on sorting time, in seconds, by observing $n$-time, $m$-time, and $p$-time plots.

Table 2: Results of 3-cube factorial experiment.

| Source | DF | Seq SS | Adj SS | Adj MS | $F$ | $p$-value |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| $r$ | 4 | 0.00039 | 0.00039 | 0.00010 | 1.15 | 0.336 |
| $n$ | 2 | 20.89061 | 20.89061 | 10.44531 | $1.2 E+05$ | 0.000 |
| $m$ | 2 | 0.15624 | 0.15624 | 0.07812 | 927.46 | 0.000 |
| $P$ | 2 | 0.02359 | 0.02359 | 0.01179 | 140.01 | 0.000 |
| $n * m$ | 4 | 0.02721 | 0.02721 | 0.00680 | 80.76 | 0.000 |
| $n * P$ | 4 | 0.00355 | 0.00355 | 0.00089 | 10.53 | 0.000 |
| $m * P$ | 4 | 0.00062 | 0.00062 | 0.00016 | 1.84 | 0.127 |
| $n * m * P$ | 8 | 0.00223 | 0.00223 | 0.00028 | 3.31 | 0.002 |
| Error | 104 | 0.00876 |  |  |  |  |
| Total | 134 | 21.11319 |  |  |  |  |

Table 3: Significant points of variance ratio $(F)$.

| Factors | RPA | FPA | MPA |
| :--- | :---: | :---: | :---: |
| $n$ | $8.6 E+04$ | $3.2 E+04$ | $1.2 E+05$ |
| $m$ | 233.21 | 180.35 | 927.46 |
| $P$ | 35.20 | 128.57 | 140.61 |
| $n P$ | 435.52 | 13.82 | 10.53 |
| $n m$ | 9.54 | 42.03 | 80.76 |
| $m P$ | 263.83 | 34.19 | 1.84 |



Figure 4
(1) n-Time Plot. $n$-time plots show that average time complexity can well be explained by a second degree polynomial with a value of $R^{2}=99.9$. We may conclude that experimentally the complexity supports an $\mathrm{O}\left(n^{2}\right)$ complexity (See Figure 1).
(2) m-Time Plot. From Figure 2 it is clear that complexity is of $\mathrm{O}\left(m^{2}\right)$.
(3) $p$-Time Plot. From Figure 3 sorting time is of $\mathrm{O}\left(P^{2}\right)$.

We may conclude that as far as the singular inputs are considered, the experimental complexity can be well


Figure 5


Figure 6
explained by second-degree polynomial in $n$ for each of the binomial inputs.

To make further investigations a second-degree composite response surface design was fitted using MINITAB statistical package and the results are given in Table 5.

Table 4: Execution time for different algorithms.

| RPA |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $P=0.2$ |  |  | $P=0.5$ |  |  | $P=0.8$ |  |  |
| $n$ | m |  |  | m |  |  | m |  |  |
|  | 50 | 100 | 150 | 50 | 100 | 150 | 50 | 100 | 150 |
| 10000 | . 5094 | . 4970 | . 4248 | . 4904 | . 5186 | . 4236 | . 4716 | . 4750 | . 4524 |
| 15000 | . 9516 | 1.056 | . 9188 | 1.408 | 1.1658 | . 9812 | 1.1374 | 1.066 | 1.159 |
| 20000 | 1.872 | 1.866 | 1.873 | 1.903 | 1.878 | 1.606 | 1.611 | 1.656 | 1.753 |
| FPA |  |  |  |  |  |  |  |  |  |
| $n$ |  | $P=0.2$ |  |  | $P=0.5$ |  |  | $P=0.8$ |  |
|  |  | $m$ |  |  | $m$ |  |  | $m$ |  |
|  | 50 | 100 | 150 | 50 | 100 | 150 | 50 | 100 | 150 |
| 5000 | . 1750 | . 1628 | . 1596 | . 1842 | . 1658 | . 1586 | . 1906 | . 1816 | . 1720 |
| 10000 | . 4752 | . 4560 | . 3972 | . 4816 | . 4376 | . 4438 | . 5218 | . 5252 | . 4748 |
| 15000 | . 9876 | . 9592 | . 8278 | . 9718 | . 9404 | . 9170 | 1.0874 | . 8778 | 1.006 |
| MPA |  |  |  |  |  |  |  |  |  |
| $n$ | $P=0.2$ |  |  | $P=0.5$ |  |  | $P=0.8$ |  |  |
|  | $m$ |  |  | m |  |  | $m$ |  |  |
|  | 50 | 100 | 150 | 50 | 100 | 150 | 50 | 100 | 150 |
| 10000 | . 406 | . 3810 | . 3576 | . 3908 | . 3690 | . 3436 | . 4062 | . 3998 | . 3876 |
| 15000 | . 8282 | . 7750 | . 7440 | . 8096 | . 7470 | . 7188 | . 8314 | . 7690 | . 7436 |
| 20000 | 1.422 | 1.344 | 1.3000 | 1.3782 | 1.2934 | 1.272 | 1.4220 | 1.3392 | 1.2968 |

Table 5: Analysis of variance for response surface design.

| Source | DF | Seq SS | Adj SS | Adj MS | $F$ | $p$-value |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| Regression | 9 | 3.97991 | 3.97991 | 0.44221 | 178.57 | 0.000 |
| $\quad$ Linear | 3 | 3.16799 | 3.16799 | 1.05600 | 426.43 | 0.000 |
| $\quad$ Square | 3 | 0.80864 | 0.80864 | 0.26955 | 108.85 | 0.000 |
| Interaction | 3 | 0.00328 | 0.00328 | 0.00109 | 0.44 | 0.730 |
| Residual error | 8 | 0.01981 | 0.01981 | 0.00248 |  | $*$ |
| Lack-of-fit | 5 | 0.01981 | 0.01981 | 0.00396 | $*$ | $*$ |
| Pure error | 3 | 0.00000 | 0.00000 | 0.00000 |  |  |
| Total | 17 | 3.99972 |  |  |  |  |

$\mathrm{R}-\mathrm{Sq}=99.5 \%$.
A high value of $R^{2}$ reveals that a second degree response surface design can be fitted to represent time as functions of input parameters.
An approximate model is given as
time $=.378+.284 n-.032 m+.011 P+.071 n^{2}+.057 m^{2}+.053 P^{2}$.

The Contour plots for different pairs are as given below.
(1) m-P Plot. The minimum response is found in the middle and the highest response is near the upper left corner (See Figure 4).
(2) n-P Plot. Minimum time of .7 seconds occurs near middle and goes up to 22 seconds when $m$ is held at middle point (See Figure 5).
(3) $n$ - $m$ Plot. At $P=.5$, minimum time of .7 seconds increases to 22 seconds which occurs in the lower right corner (See Figure 6).

For a target response of 1 second, the optimal solution is $n=10000, m=50$ and minimum time is $P=.3$; minimum sorting time is .43 seconds.

Remark. Capital letter $P$, indicates the probability of success in a trial for the Binomial variate and should not be confused with the small letter $p$ of $p$-value in in Tables 2 and 5 which gives the smallest value of level of significance at which a test statistic (here the $F$ statistic) becomes significant. Level of significance is the probability of committing type I error, i.e., the probability of rejecting the null hypothesis when it is true. Thus both $P$ and $p$ represent two different probabilities.

Further results obtained are as follows:

Condition number: $4.9015 E+00$;
$V$-optimality (average leverage): 0.5000 ;
Maximum leverage: 0.5946 .

## 3. Conclusion

As the New Sorting method is concerned the pivot element as the middle element is preferable to the pivot element being the first and also the same being randomly selected. Also the new sorting technique with middle element as the pivot element is more efficient in the sense that it can sort a large number of elements. However, when compared with Quick sort for the same number of observations to be sorted, the latter is found to be faster. Our future work is to improve the method by deleting the auxiliary array which may further reduce the sorting time as well as save space complexity.

## References

[1] K. K. Sundararajan and S. Chakraborty, "A new sorting algorithm," Applied Mathematics and Computation, vol. 188, no. 1, pp. 1037-1041, 2007.
[2] P. Kumar, A. Kumari, and S. Chakraborty, "Parameterized complexity on a new sorting algorithm: a study in simulation, annals," Annals Computer Science Series, vol. 8, no. 2, pp. 9-22, 2009.
[3] A. Kumari and S. Chakraborty, "Software complexity: a statistical case study through insertion sort," Applied Mathematics and Computation, vol. 190, no. 1, pp. 40-50, 2007.
[4] A. Kumari and S. Chakarborty, "A simulation study on quick sort parameterized complexity using response surface design," International Journal of Mathematical Modeling, Simulation and Applications, vol. 1, no. 4, pp. 448-458, 2008.
[5] H. Mahmoud, Sorting: A Distribution Theory, John Wiley Sons, 2000.


Advances in
Operations Research $=-$


The Scientific World Journal



Journal of
Applied Mathematics
-
Algebra
$\xlongequal{=}$


Journal of Probability and Statistics
$\qquad$


International Journal of Differential Equations


