International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
International Conference & workshop on Advanced Computing 2013 (ICWAC 2013) – www.ijais.org

Min-Max Select Bubble Sorting Algorithm

Dhwaneel Trivedi
Student of B.E (Computer),
TCET(Mumbai University),
Mumbai, India.

Prathmesh Trivedi,
B.Sc(I.T),
K.C (Mumbai University),
Mumbai,India.

Suraj Singh
Student of B.E (Computer),
TCET(MumbaiUniversity),
Mumbai, India.

ABSTRACT

The paper is on Sorting Algorithm which uses modified Selection Sort and modified Bubble sort. It contains explanation of procedural concept of algorithm along with implemented Algorithm. It also contains calculation on Time complexity of algorithm and highlights the key benefits of using this sorting algorithm.

General Terms

Algorithm.

Keywords

Sort,Select,Leftmost bound, Rightmost bound,Minimum, Maximum.

1. INTRODUCTION

Sorting Algorithms with reduced time complexity has always been a research topic. Performance of processor gets optimized by implementing one of the best sorting algorithms.

We were inspired by various techniques used for sorting but wanted to design a stable sorting algorithm which could sort “Maximum” Number of elements in every pass [1],[2],[3].

In Min-Max Select Bubble Sort Algorithm, initially “TWO” elements are placed at proper positions in each iteration, and then we use Neighborhood property to swap elements, hence it is advantageous [5].

This sorting algorithm uses concept of searching and selecting the Minimum and the Maximum element of the array. After selecting those elements, array is rearranged such that minimum element can be kept at leftmost bound and maximum element can be kept at rightmost bound. Then Elements are swapped based on Neighborhood’s value. These steps are repeated till the Array gets sorted.

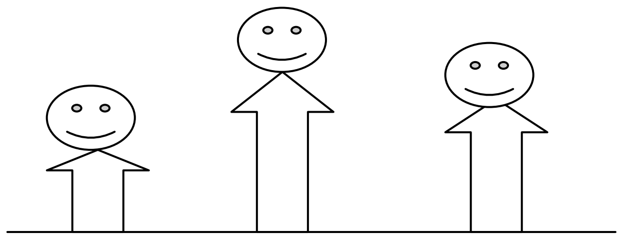
2. Procedural Concept

Initially we start using Modified Selection sort Algorithm which finds minimum and maximum elements of array and places them at respective positions. In the implementation of algorithm initially Leftmost bound is set at 0^{th} position and Rightmost bound is set at last position i.e. Length of array -1.The main loop gets repeated “n/2” times provided “n” is the Length of array. Then Minimum and Maximum elements are searched through the array within specified bound and their respective positions are stored in variables.After every iteration the distance between Leftmost bound and Rightmost bound goes on decreasing. If elements are found at proper positions i.e. Minimum element at Leftmost bound and Maximum element at the Rightmost bound then Leftmost bound is incremented by 1 and Rightmost bound is decremented by 1 and you start with modified bubble sort and then proceed to next iteration. If elements are not found at proper positions then a check is made whether the Minimum element is found at Rightmost position and Maximum element is found at Leftmost position. If condition is true then these element are Swapped and Leftmost bound is incremented by 1 and Rightmost bound is decremented by 1 and you continue to perform Modified Bubble sort. If condition is false then the array has to be rearranged. For rearrangement a comparison is made between the positions of Minimum and Maximum elements. This step is done because there is a chance of getting Maximum element at lower positions of array and Minimum element at higher positions of array. After comparison the array is right shifted from Leftmost bound till the lower position and array is left shifted from Rightmost bound till the higher position. Finally the Minimum element is stored at Leftmost bound and Maximum element is stored at Rightmost bound. Then Leftmost bound is incremented by 1 and Rightmost bound is decremented by 1 and you continue to next iteration. Hence after Maximum “n/2” iterations the array gets sorted. We have presented Worst cases where the array gets sorted in maximum “n/3” passes.

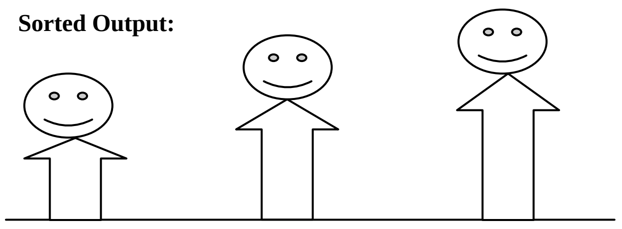
2.1 Modified Bubble sort concept:

Let us assume that Children are standing in a straight line for a competition. They have been told to stand in a line in a sorted order. Each of the children knows about his neighborhood i.e. a friend on left side and a friend on Right side. Hence at a time the middle friend can see two of his friends. They can compare their heights and swap their positions accordingly.

For example: Consider three children of varying heights standing in a line. They observe their neighbors and stand accordingly in a sorted manner.

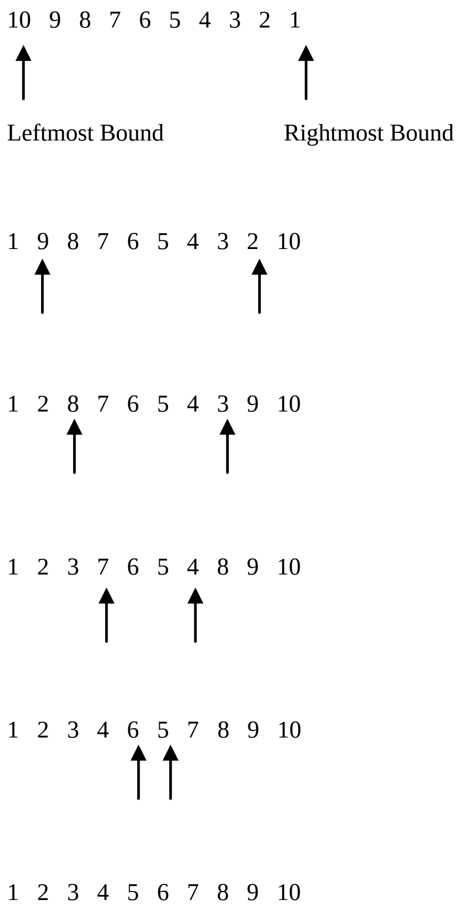


Sorted Output:



3. Explanation of Bound Concept

Our Algorithm implements Bound concept using Leftmost bound and Rightmost bound. The uniqueness of our Algorithm is that Scanning of Array elements is done between the Bounds and thus this scanning space and time goes on decreasing after every pass.[16][17]. After placing elements at proper positions we apply Modified Bubble sort which was explained above. This is explained with an example:



This concept explained above places minimum and maximum element at proper positions at every pass. The motivation behind this concept was aroused from Traditional Select Sorting Algorithm where only one minimum or maximum element is placed at proper position [17].

4. ALGORITHM

This algorithm is written to sort elements in an Ascending order. A java program has been written which implements the Algorithm.

We use Array as data structure for storing elements. ArrayLength specifies length of array.

Step 1: Initialize array, variables. Set leftmost bound to 0 and rightmost bound to ArrayLength-1. Set count to 0.

Step 2: If count is greater than Arraylength/2 or the new modified array is same as array in previous iteration go to step 8 else repeat steps 3 to 7.

Step 3: Find minimum and maximum elements within specified bound. If elements are found at proper positions, go to step 6 else go to step 4.

Step 4: If they are found at opposite positions then swap elements and go to step 6 else go to step 5.

Step 5: Choose minimum and maximum positions from the index of minimum and maximum elements. Perform right shift operation from leftmost bound till minimum position only if the leftmost bound is not equal to minimum position and left shift operation from rightmost bound till maximum position only if rightmost bound is not equal to maximum position. Then store minimum element at leftmost bound and maximum element at rightmost bound and go to step 6.

Step 6: Increment count, Increment leftmost bound; decrement rightmost bound.

Step 7: Check for Neighborhood property: This property is applied from the position of leftmost bound till the position of rightmost bound. Comparisons are made between Array[middle], Array[left], Array[right]; and positions are swapped accordingly based on ascending order and go to step 2.

Step 8: Stop and display the sorted array.

5. Comparisons of Sortingalgorithms on basis of Examples

In this section we compare results of Min-Max Select Bubble sort, Bubble sort, and Min-Max Selection sort by observing number of passes based on variation of inputs.

5.1 Best Case Example

If the input array is given in Ascending order then the output is only “1 pass” for any length of input Array for above proposed algorithm.

5.1.1 *Output of Min-Max Select Bubble Sort*

```
Enter size:
10
Enter elements:
1 2 3 4 5 6 7 8 9 10
Pass 1: 1 2 3 4 5 6 7 8 9 10
Elements after sorting:
1 2 3 4 5 6 7 8 9 10
Total number of passes: 1
```

5.1.2 *Output of Bubble Sort*

```
Pass 1: 1 2 3 4 5 6 7 8 9 10

Pass 2: 1 2 3 4 5 6 7 8 9 10

Pass 3: 1 2 3 4 5 6 7 8 9 10

Pass 4: 1 2 3 4 5 6 7 8 9 10

Pass 5: 1 2 3 4 5 6 7 8 9 10
Pass 6: 1 2 3 4 5 6 7 8 9 10

Pass 7: 1 2 3 4 5 6 7 8 9 10

Pass 8: 1 2 3 4 5 6 7 8 9 10

Pass 9: 1 2 3 4 5 6 7 8 9 10

Pass 10: 1 2 3 4 5 6 7 8 9 10

Total number of passes: 10
```

Hence the sorting algorithm suggested in this paper takes only "1 pass" instead of "10 passes" of bubble sort.

5.1.3 *Output of Min-Max Selection sort*

```
Pass 1: 1 2 3 4 5 6 7 8 9 10

Total number of passes: 1
```

The output is same as our Algorithm's output.

5.2 Average case Example

Let us take an example of array with even number of elements say 8, so it must take maximum of 4 passes to sort this array. But it takes only "1 pass" to sort the array for our algorithm.

5.2.1 *Output of Min-Max Select Bubble Sort*

```
Enter size:

8

Enter elements:

100 -4 3 2 1 9 18 91

Pass 1:-4 1 2 3 9 18 91 100

Elements after sorting:

-4 1 2 3 9 18 91 100

Total number of passes: 1
```

5.2.2 *Output of Bubble Sort*

```
Pass 1: -4 3 2 1 9 18 91 100

Pass 2: -4 2 1 3 9 18 91 100

Pass 3: -4 1 2 3 9 18 91 100

Pass 4: -4 1 2 3 9 18 91 100

Pass 5: -4 1 2 3 9 18 91 100

Pass 6: -4 1 2 3 9 18 91 100

Pass 7: -4 1 2 3 9 18 91 100

Pass 8: -4 1 2 3 9 18 91 100

Total number of passes: 8
```

Hence the sorting algorithm suggested in this paper takes only "1 pass" instead of "8 passes" of bubble sort.

5.2.3 *Output of Min-Max Selection sort*

```
Pass 1: -4 3 2 1 9 18 91 100

Pass 2: -4 1 3 2 9 18 91 100

Pass 3: -4 1 2 3 9 18 91 100

Total number of passes: 3
```

Hence the sorting algorithm suggested in this paper takes only "1 pass" instead of "3 passes" of Min-Max Selection sort.

5.3 Worst case Example:

Let us give inputs in Descending order. Here the Number of passes are always "n/3" for any input of length "n" for our Algorithm.

5.3.1 *Output of Min-Max Select Bubble Sort*

```
Enter size:

10

Enter elements:

10 9 8 7 6 5 4 3 2 1

Pass 1: 1 7 6 5 4 3 2 8 9 10

Pass 2: 1 2 5 4 3 6 7 8 9 10

Pass 3: 1 2 3 4 5 6 7 8 9 10

Elements after sorting:

1 2 3 4 5 6 7 8 9 10

Total number of passes: 3
```

5.3.2 *Output of Bubble Sort*

```
Pass 1: 9 8 7 6 5 4 3 2 1 10

Pass 2: 8 7 6 5 4 3 2 1 9 10

Pass 3: 7 6 5 4 3 2 1 8 9 10

Pass 4: 6 5 4 3 2 1 7 8 9 10

Pass 5: 5 4 3 2 1 6 7 8 9 10

Pass 6: 4 3 2 1 5 6 7 8 9 10

Pass 7: 3 2 1 4 5 6 7 8 9 10

Pass 8: 2 1 3 4 5 6 7 8 9 10

Pass 9: 1 2 3 4 5 6 7 8 9 10

Pass 10: 1 2 3 4 5 6 7 8 9 10

Total number of passes: 10
```

Hence the sorting algorithm suggested in this paper takes only "3 passes" instead of "10 passes" of bubble sort.

5.3.3 *Output of Min-Max Selection sort*

```
Pass 1: 1 9 8 7 6 5 4 3 2 10

Pass 2: 1 2 8 7 6 5 4 3 9 10

Pass 3: 1 2 3 7 6 5 4 8 9 10

Pass 4: 1 2 3 4 6 5 7 8 9 10

Pass 5: 1 2 3 4 5 6 7 8 9 10
```

Hence the sorting algorithm suggested in this paper takes only "3 passes" instead of "5 passes" of Min-Max Selection sort.

6. Effect of Variation of Inputs

Here we consider variation of inputs and compare the three algorithms. The number of passes depends upon the different values of inputs. In this scenario also our Min-Max Select Bubble Sort provides the most efficient output.

6.1 Redundant Inputs

Min-Max Select Bubble Sort is a stable sort which gives correct sorted output for redundant inputs.

6.1.1 *Output of Min-Max Select Bubble Sort*

```
Enter size:
10
Enter elements:
2 2 3 3 1 1 0 0 -10 -10
Pass 1:-10 2 1 1 0 0 -10 2 3
Pass 2:-10 -10 1 0 0 2 2 2 3
Pass 3:-10 -10 0 1 1 2 2 2 3
Elements after sorting:
-10 -10 0 1 1 2 2 2 3 3
Total number of passes: 3
```

6.1.2 *Output of Bubble Sort*

```
Pass 1:2 2 3 1 1 0 0 -10 -10 3
Pass 2:2 2 1 1 0 0 -10 -10 3 3
Pass 3:2 1 1 0 0 -10 -10 2 3 3
Pass 4:1 1 0 0 -10 -10 2 2 3 3
Pass 5:1 0 0 -10 -10 1 2 2 3 3
Pass 6:0 0 -10 -10 1 1 2 2 3 3
Pass 7:0 -10 -10 0 1 1 2 2 3 3
Pass 8:-10 -10 0 0 1 1 2 2 3 3
Pass 9:-10 -10 0 0 1 1 2 2 3 3
Pass 10:-10 -10 0 0 1 1 2 2 3 3
Total number of passes: 10
```

Hence the sorting algorithm suggested in this paper takes only "3 passes" instead of "10 passes" of bubble sort.

6.1.3 *Output of Min-Max Selection sort*

```
Pass 1: -10 2 2 3 1 1 0 0 -10 3
Pass 2: -10 -10 2 2 1 1 0 0 3 3
Pass 3: -10 -10 0 2 1 1 0 2 3 3
Pass 4: -10 -10 0 0 1 1 2 2 3 3
Total number of passes: 4
```

Hence the sorting algorithm suggested in this paper takes only "3 passes" instead of "4 passes" of Min-Max Selection sort.

6.2 Random Inputs

Here we observe outputs over Randomized inputs.

6.2.1 *Output of Min-Max Select Bubble Sort*

```
Enter size:
10
Enter elements:
10 2 -1 4 -3 -4 -5 5 7 6
Pass 1:-5 -1 -3 -4 2 4 5 6 7 10
Pass 2:-5 -4 -3 -1 2 4 5 6 7 10
Elements after sorting:
-5 -4 -3 -1 2 4 5 6 7 10
Total number of passes: 2
```

6.2.2 *Output of Bubble Sort*

```
Pass 1:2 -1 4 -3 -4 -5 5 7 6 10
Pass 2:-1 2 -3 -4 -5 4 5 6 7 10
Pass 3:-1 -3 -4 -5 2 4 5 6 7 10
Pass 4:-3 -4 -5 -1 2 4 5 6 7 10
Pass 5:-4 -5 -3 -1 2 4 5 6 7 10
Pass 6:-5 -4 -3 -1 2 4 5 6 7 10
Pass 7:-5 -4 -3 -1 2 4 5 6 7 10
Pass 8:-5 -4 -3 -1 2 4 5 6 7 10
Pass 9:-5 -4 -3 -1 2 4 5 6 7 10
Pass 10:-5 -4 -3 -1 2 4 5 6 7 10
Total number of passes: 10
```

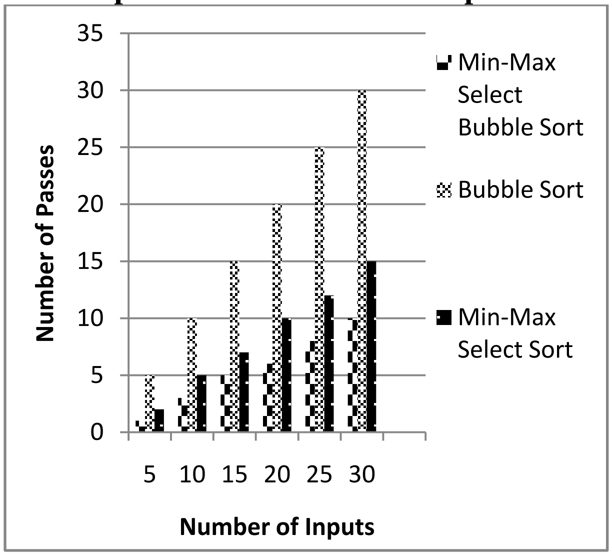
Hence the sorting algorithm suggested in this paper takes only "2 passes" instead of "10 passes" of bubble sort.

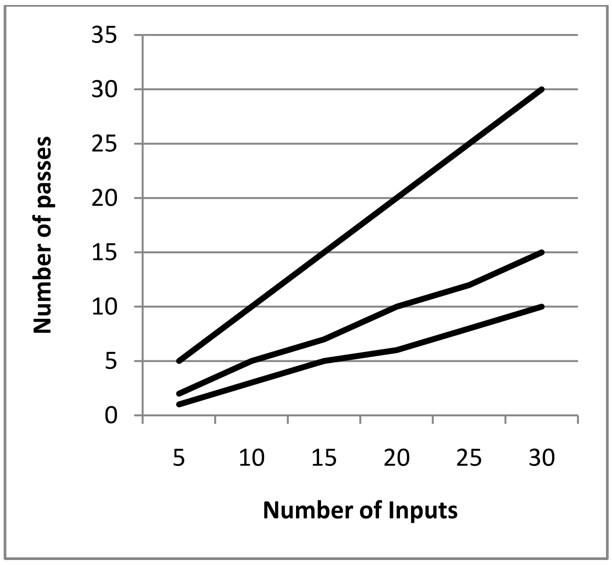
6.2.3 *Output of Min-Max Selection sort*

```
Pass 1: -5 2 -1 4 -3 -4 5 7 6 10
Pass 2: -5 -4 2 -1 4 -3 5 6 7 10
Pass 3: -5 -4 -3 2 -1 4 5 6 7 10
Pass 4: -5 -4 -3 -1 2 4 5 6 7 10
Total number of passes: 4
```

Hence the sorting algorithm suggested in this paper takes only "2 passes" instead of "4 passes" of Min-Max Selection sort.

7. Comparison of Results on Graphs





As clearly seen from the above graph that the lowest curve represents Min-Max Select Bubble Sort, Middle curve represents Min-Max Select sort, and the Upper curve represents Bubble sort.

8. Distinguishing characteristics of Algorithms

8.1 Best case Complexity:

(a) Min-Max Select Bubble Sort has O(n).
(b) Min-Max Select Sort has O(n).
(c) Bubble Sort has O(n^2).

8.2 Average case Complexity:

(a) Min-Max Select Bubble Sort has Θ(n^2).
(b) Min-Max Select Sort has O(n^2).
(c) Bubble Sort has O(n^2).

8.3 Worst case Complexity:

(a) Min-Max Select Bubble Sort has Θ(n^2).
(b) Min-Max Select Sort has O(n^2).
(c) Bubble Sort has O(n^2).

8.4 Number of Passes:

(a) Min-Max Select Bubble Sort has Passes less than or equal to "n/3".
(b) Min-Max Select Sort has Passes less than or equal to "n/2".
(c) Bubble Sort hasPasses equal to "n".

8.5 Procedure:

(a) Min-Max Select Bubble Sort implements Selection and Exchange.
(b) Min-Max Select Sort implements Selection.
(c) Bubble Sort implements Exchange.

9. Calculation of Complexity for Min-Max Select Sorting Algorithm

Assume n to be Length of Array let k denote kth iteration.

1. **Outside loop** has time complexity of n/2+1 if n is even and n/2 for n is odd.

Inside loop:

2**. Finding maximum and minimum element:**

n+1+(n-2)+1+(n-4)+1+... which is generalized to

(n- (2*k-2))+1,Where k=1,2,3,…

3.**Forshifting operations:** It has linear Time complexity and depends upon positions of elements. Generalized complexity is n-$(2)^k$, Where k=1,2,3,…

4**. For Neighborhood property of Modified Bubble sort:**

It has linear Time complexity where loop starts from leftmost Bound and goes upto rightmost bound.

Complexity=n + (n-1) + (n-2) + (n-3) +…

Generalized complexity is n+1-k where k=1,2,3,…

5. **Total Time ComplexityFrom 1, 2, 3, and 4:**

$= (n/2+1)*((n-(2*k-2)+1)+n-(2)^k+(n+1-k))$

$= (n/2+1)*(3n-(2^k)+2+1-(2^k)+1-k)$

$= (n/2+1)*(3n+4-(2^{k+1})-k)$

$=(3/2)*(n)^2 +(2)*n-(2^k)*n-k*(n/2)+3n+4-(2^{k+1})-k$

Therefore $T(n)\leq(n^2)$.

6. **For Best case scenario when array is already sorted and given as Input:**

The number of passes is always 1 hence a check is made whether the old array is same as sorted array which takes at least computational time of "n".

Therefore T(n)≥(n).

7.**Total complexity of Algorithm from 5 and 6 :**

$(n) \leq T(n) \leq (n^2)$

Hence Time complexity is $\Theta(n^2)$.

This complexity $\Theta(n^2)$ remains same for all cases i.e. Best case when array is already sorted ,Average case ,and Worst case when elements of array are in opposite order.

10. Key Benefits

This Algorithm may appear as modification of Min-Max Selection Sort algorithm [14],[7],[4] as it selects and sorts minimum and maximum elements in every pass, but the process of re-arranging elements is different, and similar to Bubble sort Algorithm [6],[8] ,but length of swapping of elements goes on decreasing in every pass. As this algorithm has Time complexity of $\Theta(n^2)$ it has got a major advantage over Bubble Sort [9],[10] and Traditional Selection sort [11],[12] which have Time complexities of $O(n^2)$ and an advantage of sorting at least 2 elements in every pass. Hence it can be used in areas where maximum and minimum elements are required from set of elements within a given time span [13],[14].Moreover we have seen in many examples it simply sorts array where number of passes are less than or equal to one-third the length of array. It is a Stable sorting algorithm which gives Sorted output even for redundant inputs.

11. Conclusion

Min-Max Select Bubble sort as a stable sorting algorithm [15] has been conceptualized and implemented in java programming language. It provides a challenge for reducing the time complexity from $\Theta(n^2)$ to lesser complexities. The benefits of this algorithm can be used in various computational areas like sorting of contact lists, Web Browsers, etc. One of the advantages is that it is a stable sort which can be used in all applications where similar valued keys are used in databases, same last name but different first name of people, etc. It can also be implemented on parallel processors [18] by setting different rightmost and leftmost bound and get optimum result at right time. Although the number of passes are based on different types of inputs, if the array is already sorted it just take only "one pass" and sorts the array in $\Theta(n)$.Hence it just takes one pass to display the output.

12. References

[1] E.Horowitz,S.Sahni,S.Rajasekaran, " Fundamentals of Computer Algorithms",pp.165-174.

[2] Y.Langsam,M.Augenstein,A.Tenenbaum,"Data Structures Using Java",pp.364-365.

[3] N.Dell,D.Joyce,C.Weems,"Object-oriented data structures using java",pp.678-682.

[4] R.Lorentz, " Recursive algorithms",pp.54.

[5] D.Sharma,V.Thapar,R.A.Ammar,S.Rajasekaran,M.Ahmed, "Efficient sorting algorithms for the cell broadband engine,"Computers and Communications,2008.ISCC 2008.IEEE Symposium.

[6] F.G.Khan,O.U.Khan, B.Montrucchio, P.Giaccone , "Frontiers of Information Technology(FIT)" ,2011.

[7] S.Rajasekaran, S.Sahni, "Sorting, Selection, and Routing on the Arraywith ReconfigurableOptical Buses", IEEE Transactions,VOL 8.NO.11.1997.

[8] S.Khamitkar,P.Bhalchandra,S.Lokhande,N.Deshmukh, "The Folklore of Sorting Algorithms",IJCSI Journal,Vol.4, NO.2,2009.

[9] V.Mansotra,K.Sourabh, "Implementing Bubble Sort Using a New Approach",INDIACom-2011.

[10] N.Arora,S.Kumar,V.Tamta,"A Novel Sorting Algorithm and comparison with Bubble Sort and Insertion Sort",IJCA,2012.

[11] Flores, I. "Analysis of Internal Computer Sorting". J.ACM 7,4 (Oct. 1960), 389- 409.

[12] Knuth, D. "The Art of Computer programming Sorting and Searching" , 2nd edition, vol.3. Addison-Wesley, 1998.

[13] J. L. Bentley and R. Sedgewick. "Fast Algorithms for Sorting and Searching Strings", ACM-SIAM SODA 97, 360-369, 1997.

[14] N.Murthy, " Min-max sort: a simple sorting method" CSC '87 Proceedings of the 15th annual conference on Computer Science Page 365 ACM ,USA.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein."Introduction to Algorithms". MIT Press, Cambridge, MA, 2nd edition, 2001.

[16] J.Alnihoud and R.Mansi, "An Enhancement of Major Sorting Algorithms",The International Arab Journal of Information Technology,Vol 7, No.1, January 2010.

[17] S.Chand,T.Chaudhary,R.Parveen,"Upgraded Selection Sort",IJCSE,Vol.3,No.4,April2011.

[18] A.Grama,A.Gupta,G.Karypis,V.Kumar, "Introduction to Parallel Computing",Second Edition,Addison-Wesley.