LAUTECH Journal of Computing and Informatics (LAUJCI) – ISSN: 2714-4194
Volume 3 Issue 1, June 2023 – www.laujci.lautech.edu.ng

EMPIRICAL ANALYSIS OF FACTORS AFFECTING THE PERFORMANCE OF MINIMUM SPANNING TREE ALGORITHMS USING PRINCIPAL COMPONENT

O. T Odebunmi
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
busayotola@gmail.com

A.O Olabode
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
anthony2olabode@gmail.com

Y.S Jeremiah
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
teejabar5@gmail.com

Adeleke I.A
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
adeleke_israel@yahoo.com

M.O Lawrence
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
morolake.lawrence@bazeuniversity.com

S. O. Olabiyisi
Dept. of Computer science
Ladoke Akintola University of Technology, Ogbomoso, Nigeria
soolabiyisi@lautech.edu.ng

ABSTRACT

The Minimum Spanning Tree (MST) of a graph is the cheapest subset of edges that keeps the graph in one connected component. The significant impact of overall efficiency of a minimum spanning tree is hugely determined by the efficiency of some selected factors such as time taken, memory usage and number of edges visited. However, the level at which each factor affect the performance of the minimum spanning tree is yet to be investigated. The experimentation evaluation was performed on the MST algorithms (Borukva, Kruskal, Prim and Reverse-Delete) by varying the input routes of Lagos State and Federal Capital Territory Road line distances, such that data were generated. Seventy two data samples were obtained from the experiments. The MST algorithms studied were implemented using Java Programming Language. The performance analysis of Borukva, Kruskal, Prim and Reverse-Delete spanning techniques were evaluated based on time taken, memory usage and number of edges visited. Statistical analysis was further performed on the evaluation results using Factor analysis by principal component for the analysis of the generated data. The percentages of variance based on time taken, memory usage and number of edges visited for Borukva were 84.90%, 14.40% and 0.65%, respectively, while the corresponding values for Kruskal were 82.30%, 30.80% and 10.58%, respectively. Also, the percentages of variance based on time taken, memory usage and number of edges visited for Prim were 86.10%, 13.10% and 0.81% respectively, while the corresponding values for Reverse-Delete were 58.70%, 17.50% and 0.13%. The percentage of variance forms the basis for establishing the level of contribution of each factor towards the performance of the MST algorithms. The study revealed that main factor affecting the efficiency of minimum spanning tree algorithm was time taken.

Keywords: Critical factors, Decision variables, Factor Analysis, Minimum Spanning Tree algorithms, Principal Component Analysis (PCA)

1. INTRODUCTION

A spanning tree is a sub-graph of the graph that contains all the vertices. A Minimum Spanning Tree (MST) of a weighted graph is a spanning tree in which the sum of the weights of all its edges is minimum of all such possible spanning trees of the graph. There can be multiple MST of a graph, but all of these MSTs must have unique same total cost [1]. A graph is a collection of vertices and edges, and each edge connects a pair of vertices. MST proves important for several reasons; they can be computed quickly and easily, and create a sparse sub graph that reflects a lot about the original graph. They also provide a way to identify clusters in sets of points. Factor analysis is a branch of dimension reduction method. It seeks to discover if the observed variables can be interpreted in a more compressed form with few numbers of variables called

factors [2]. The significant impact of overall efficiency of a program is hugely determined by the approach used in the evaluation of factors influencing the performance of the program. Spanning is an important process that determines the efficiency of many computing tasks and procedures. This paper determined the level of impact that time taken to span, memory usage and number of edges visited has on the efficiency of some selected spanning tree techniques; Borukva, Kruskal,Prim and Reverse-Delete algorithm.
Minimum Spanning Tree (MST) has long been of interest to mathematicians because of their many applications. Most commonly, cable and communications companies can represent the task of connecting every house in a network in the least expensive way possible as an MST problem. In this case, the cost of fixing meters between houses corresponds to the weights of the edges. There are analogous applications to transportation networks, such as determining the least expensive method of connecting a number of islands or bodies of land [3].There are various classical algorithms available such as Kruskal, Prim and Borukva algorithm. Kruskal, Prim and Borukva are greedy algorithms which is use to find a minimum spanning tree for a connected weighted undirected graph. This means that, when the total weight of all the edges is minimized in the tree, at that time it finds a subset of the edges which forms a tree which includes every vertex[4]. Another classical algorithm is Reverse – Delete.
Borukva algorithm is an algorithm for finding a minimum spanning tree in a graph for which all edge weights are distinct. It was first published in 1926 by Otakar Borukva as a method of constructing an efficient electricity network for Moravia [5]. The algorithm was rediscovered by Choquet (1938) again by Florek, Łukasiewicz, Perkal, Steinhaus, and Zubrzycki in 1951; and again by Sollin in 1965. Because Sollin was the only computer scientist in this list living in an English speaking country, this algorithm is frequently called Sollin algorithm, especially in the parallel computing literature. The algorithm begins by first examining each vertex and adding the cheapest edge from that vertex to another in the graph, without regard to already added edges, and continues joining these groupings in a like manner until a tree spanning all vertices is completed. Borukva algorithm can be shown to take ***O(log V)*** iterations of the outer loop until it terminates, and therefore to run in time ***O(E log V),*** where ***E*** is the number of edges, and V is the number of vertices in ***G***.
The Prim algorithm is also known as the DJP (Dijkstra-Jarnik Problem) algorithm, the Jarnik algorithm, or the Prim–Jarnik algorithm. Using a simple binary Prim data structure complexity is ***O(|E| log |V|),*** where ***|E|*** is the number of edges and ***|V|*** is the number of vertices. Using Fibonacci Prims in dense graph complexity is ***O(|E| + |V| log |V|),*** which is asymptotically faster [6]. Prim’s algorithm also looks for one edge at a time. The difference, however, is that it starts with a single node and intent to grow the discovered tree by adding one new vertex who is the most closest node to the tree.
Kruskal algorithm is a greedy algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component) [7]. Kruskal algorithm is easy to understand and give good result for large number of vertices and edges.
The reverse-delete algorithm is an algorithm in graph theory used to obtain a minimum spanning tree from a given connected, edge-weighted graph. If the graph is disconnected, this algorithm will find a minimum spanning tree for each disconnected part of the graph. The set of these minimum spanning trees is called a minimum spanning forest, which contains every vertex in the graph.
This algorithm is a greedy algorithm, choosing the best choice given any situation. It is the reverse of Kruskal algorithm, which is another greedy algorithm to find a minimum spanning tree. Kruskal algorithm starts with an empty graph and adds edges while the Reverse-Delete algorithm starts with the original graph and deletes edges from it. Algorithm 4 presents the algorithm for Reverse delete algorithm. The three factors (time taken, number of edges visited and memory usage) were considered in evaluating the four minimum spanning tree algorithms. This paper aims at determining the most critical of the three factors.

2. LITERATURE REVIEWS

Ardhendu, Jayanta and Pal [8] in their work on a new algorithm for finding MST with undirected neutrosophic graphs, they investigated the MST problem whose edges weights are represented by neutrosophic numbers. The main contribution of their study is to provide an algorithmic approach to find the minimum spanning tree in uncertain environment using neutrosophic numbers as arc lengths. They incorporated the concept of uncertainty in Kruskal algorithm alone without considering other algorithms.

Shivaani, Nithish and Joshua [10] in the paper titled Performance analysis of minimum spanning tree algorithm. Their work compares the efficiency of modern MST algorithms like Least Cost MST with the classical algorithms such as Prim's, Kruskal's and Borukva's algorithms. The research observed that for applications demanding less elapsed time, Borukva's and Kruskal's algorithms can be used. It was also observed that, for applications demanding less memory consumption, Prim's algorithm is the most suitable one. Their work only considered time and memory consumption without considering number of edges visited.

3. MATERIALS AND METHOD

This research investigates the level at which some selected factors affect the performance of minimum spanning tree. The performance of the four algorithms were tested on Lagos State and FCT Road map distance as case studies to produce different results for the considered factors such as time taken, memory usage and number of edges. PCA was used as a factor analysis technique to determine the most critical factor. The basic stages involved in the approach are;

i. Critical factors or decision variables selected were three and determined through the extensive review of literatures and comprehensive survey of journals and articles. The performance of the spanning algorithm in one factor affects its performance in another factor.
ii. The minimum spanning algorithms were implemented in Java programming language and the performance of each of the algorithms were tested on Nigeria Road map distance to generate the experimental data used for this research. The efficiency of the algorithms were evaluated in terms of time taken, memory usage and number of edges. The Java program compiled time taken in nanoseconds, memory usage in bits.
iii. The results obtained from the implementation of the algorithms were presented to SPSS23.0 using PCA as factor analysis technique.

Mathematical model approach was developed for the evaluation of the decision variables or critical factors according to [1] as presented in equation (1). The decision variables are declared in terms of time taken, memory usage and number of edges visited are interrelated to one another. Such that performance of algorithms in one factor could affect its performance in another

$$y_i = \sum_{k}^{n} a_{i,k} X_k \ldots\ldots\ldots\ldots\ldots. \quad I = 1,2,3 \ldots\ldots\ldots\ldots\ldots\ldots., m \tag{1}$$

Where y_i represents the i^{th} assessor's observation of decision variable X_k; $a_{i,k}$ represents the assessment of k^{th} decision variable by i^{th} assessor.

For a sample population of minimum spanning tree algorithms, system of linear equations is obtained expresses as:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ Y_m \end{pmatrix} = \begin{pmatrix} a_{1.1}\, X_1 \; + \cdots + \; a_{1,3} \quad X_3 \\ \cdot \\ \cdot \\ a_{m,n} \quad X_1 + \cdots + \; a_{m,3} \; X_3 \end{pmatrix} \tag{2}$$

The factor analysis by principal component is adopted for the evaluation of the impact of time taken, memory usage and number of edges visited on the performance of the four minimum spanning tree algorithms. The following statistics were generated and used for the purpose of achieving the stated goal of determining the most critical factor.

a. Descriptive statistics
b. Correlation matrix
c. Kaiser-Meyer Olkin (KMO) and Bartlett's test
d. Communalities
e. Rotated factor loadings
f. Eigenvalues and percentage of variance
g. Component matrix

The descriptive statistics gives the mean and standard deviation of the raw score of each performance indices given by the sample assessors. The correlation matrix presents the degree of relationships between paired decision variables [11]. The barlett's test of sphericity is used to test the adequacy (true representation) of the sample from the population. Another measure of the adequacy of a sample is Kaiser-Meyer Olkin (KMO).

In factor analysis, there is a set of factors which is generally referred to as “common factors” each of which loads on some performance indices and another set of factors which are extraneous to each of the performance indices [11]. The proportion of a variance of a performance indices explained by the common factor is called “communality” of the performance indices. The communality of the performance index ranges between 0and 1, where 0 indicates that the common factors explain none of the variance and 1 indicates that all variance is explained by the common factors.
According to [11], the component matrix presents the initial factor loading. The factor loading associated with a specific variable is simply the correlation between the factor and variable’s standard scores. Each factor represents an area of generalization that is qualitatively distinct from that represented by any other factor. The degree of generalization found between each variable and each factor is referred to as “factor loading”. The farther a factor loading is from zero in the positive direction, the more one can conclude the contribution of a variable to a factor. The component matrix can be rotated by varimax, promax, equamax or quartimax for the purpose of evaluating high correlation between indices and factors. The factor score coefficient matrix can be used to evaluate the assessment of each assessor is generated. The eigenvalues and percentage variance of the factors extracted are generated for the purpose of evaluating the contributions of each factor to the efficiency of the minimum spanning tree algorithms [11, 12].
The generated component score coefficient matrices are used to estimate the assessment of each assessor of the impact of time taken, number of edges visited and memory usage on the efficiency of minimum spanning trees.
This can be achieved by formulating a linear equation of the form:

$$C_{i,j} - \sum_{k=1}^{3} b_{k,j}\, S_{i,k} \qquad i = 1,2 \ldots\ldots\ldots n;\ \ j - 1 \tag{3}$$

Where $C_{i,j}$ represents the contribution of i^{th} assessor to j^{th} factor; $b_{k,j}$ represents the component score coefficient of k^{th} decision variables for j^{th} factor; $S_{i,k}$ represents the standard score of i^{th} assessor for k^{th} decision variable and n represents the number of sampled assessors.
$S_{i,k}$ is estimated by:

$$S_{i,x} = A + \left(\frac{x_i + y_i}{d_i}\right) \tag{4}$$

Where A represents the allowable minimum raw score for decision variable; in this instance, it is *I* ;x_i represents the raw score of the i^{th} decision variable; y_i represents the mean of the raw scores of the i^{th} decision variable; d_i represents the standard deviation of the raw scores of i^{th} decision variable. For each sampled Assessor, the system of linear equations for the single extracted factor can be represented as follows:

$$b_{1,1}, S_{i,1} + b_{2,1}S_{1,2} + \cdots b_{4,1}S_{i,4} = C_{i,1} \tag{5}$$

In an attempt to evaluate the percentage contribution of each factor to the efficiency of the minimum spanning tree, the eigen value of each factor is generated. This is presented in Table 5 through Table 9. The eigen value of j^{th} factor denoted by ‘E_j’ is calculated by:

$$E_j = \sum_{k=1}^{3} X_{i,j}^2$$

$$i = 1,2,3; \qquad j = 1 \tag{6}$$

Where $X_{i,j}$ represents the number of decision variables considered in this study. Table 5 to 8 present the eigen values, percentage contribution and cumulative percentage contribution of the three considered factors for each of the four minimum spanning tree algorithm according to [10].

4. RESULTS AND DISCUSSION

Experiments were conducted on the four aforementioned algorithms by varying the input route lines of Lagos and Federal Capital Territory road map distance to generate data that were used for the analysis. The performance of the algorithms was tested for each of the experiment by varying the input routes to produce different results for the time taken, memory usage and number of edges visited. The numbers of data generated were seventy-two (72) and each minimum spanning algorithm has records for the time taken, number of edges visited and memory usage.

Descriptive statistics show the mean and standard deviation of the rating of the impact of the time taken, number of edges visited and memory used on the efficiency of minimum spanning techniques by the experimental results generated. For instance, the mean and standard deviation for the mean and standard deviation for Borukva algorithm on rating of time taken, memory usage and number of edges were (1391214.670,2129201.780), (581261.330,184036.741) and (28.170, 27.477), respectively as illustrated in Tables 1-4. Thereafter, the final data were subjected to factor analysis by principal component for statistical analysis using SPSS package. Principal component analysis was used for the extraction method and the rotation method by promax with Kaiser Normalization. According to the computed analysis, the analyzed results show that each factors show high correlation in terms of their loading on the four minimum spanning techniques. Borukva spanning technique for instance, the correlation Borukva algorithm has correlation between time taken and memory usage to be 0.948, time taken and number of edge to be 0.770. The implication is that time taken is not likely to share the same factor with the number of edges visited. On the other hand, number of edges visited is likely to share the same factor with the memory.
The analyzed results for Borukva algorithm show that KMO value is 0.571 and Bartlett's test value is 0.005, the degree of freedom (Df) is 3, since 3 factors (time taken, memory usage and number of edge) are put into analysis. The results obtained from the Bartlett's test and KMO test are good indicators of the suitability of factor analysis.
The communalities of the performance indices generated for the minimum spanning technique based algorithms with principal component analysis as the extraction method are presented in Table 5 through Table 8 for all the four minimum spanning algorithms.

Table 1: Descriptive Statistics for Borukva Algorithm

	Mean	Std Deviation	Analysis N	Missing N
Time taken	1391214.7	2129201.8	6	0
Memory usage	581261.33	184036.7	6	0
Number of edges	28.17	27.5	6	0

Table 2: Descriptive Statistics for Kruskal Algorithm

	Mean	Std Deviation	Analysis N	Missing N
Time taken	1391214.7	2129201	6	0
Memory usage	527130.3	176587.2	6	0
Number of edges	28.3	26.5	6	0

Table 3: Descriptive Statistics for Prim Algorithm

	Mean	Std Deviation	Analysis N	Missing N
Time taken	1005813.0	1736108.2	6	0
Memory usage	472218.8	142809.0	6	0
Number of edges	25.0	21.0	6	0

Table 4: Descriptive Statistics for Reverse-Delete Algorithm

	Mean	Std Deviation	Analysis N	Missing N
Time taken	1222646.0	1973969.6	6	0
Memory usage	541167.5	172976.9	6	0
Number of edges	28.67	28.8	6	0

Table 5: Component Score Coefficient Matrix for Borukva

	Component
	1
Time taken(nanoseconds)	0.388
Number of edges visited	0.364
Memory used (bits)	0.331

Table 6: Component Score Coefficient Matrix for Kruskal

	Component
	1
Time taken(nanoseconds)	0.348
Number of edges visited visited	0.518
Memory used (bits)	0.422

Table 7: Component Score Coefficient Matrix for Prim

	Component
	1
Time taken(nanoseconds)	0.384
Number of edges visited visited	0.353
Memory used (bits)	0.339

Table 8: Component Score Coefficient Matrix for Reverse-delete

	Component
	1
Time taken(nanoseconds)	0.395
Number of edges visited visited	0.386
Memory used (bits)	0.315

The total variance explained determines the number of components to be extracted during analysis. The extraction is based on components with eigen values greater than 1. Component with eigen values greater than 1 was retained for further analysis. Tables 9-12 depicts total variance explained of the four algorithms. Borukva algorithms only component 1 was extracted with eigen value of 2.548 and percentage of variance of 89.934 while component 2 and 3 were discarded. Likewise in Kruskal algorithm, only component 1 was extracted with eigen value of 1.760 and percentage of variance of 58.669 while component 2 and 3 were discarded. Similarly, in Prim algorithm, only component 1 was extracted with eigen value of 2.584 and percentage of variance of 86.129 while component 2 and 3 were discarded. Ultimately, in reverse-delete algorithm, only component 1 was extracted with eigen value of 2.470 and percentage of variance of 82.327 while component 2 and 3 were discarded.

Table 9: Eigen values generated for Borukva Algorithm

Components	Initial Eigen values			Extraction sums of Square Loading		
	Total	% of Var.	Cumm.%	Total	% of Var.	Cumm.%
1	2.548	84.934	84.934	2.548	84.934	84.934
2	0.433	14.417	99.351			
3	0.019	0.649	100.000			

Table 10: Eigen values generated for Kruskal Algorithm

Components	Initial Eigen values			Extraction sums of Square Loading		
	Total	% of Var.	Cumm.%	Total	% of Var.	Cumm.%
1	1.760	58.669	58.669	1.760	58.669	58.669
2	0.923	30.756	89.425			
3	0.317	10.575	100.000			

Table 11: Eigen values generated for Prims Algorithm

Components	Initial Eigen values			Extraction sums of Square Loading		
	Total	% of Var.	Cumm.%	Total	% of Var.	Cumm.%
1	2.584	86.129	86.129	2.584	86.129	86.129
2	0.392	13.056	99.187			
3	0.024	0.813	100.000			

Table 12: Eigen values generated for Reverse-delete Algorithm

Components	Initial Eigen values			Extraction sums of Square Loading		
	Total	% of Var.	Cumm.%	Total	% of Var.	Cumm.%
1	2.470	82.327	82.327	2.470	82.327	82.327
2	0.526	17.546	99.872			
3	0.004	0.128	100.000			

The three factors contribute a total of 100% to the efficiency of the four minimum spanning tree algorithms. From the results, 'time taken' contributed 84.934 %, number of edges visited contributed 14.417% and memory usage contributed 0.649 % impact on the efficiency of Borukva algorithm. This can be visualized in Figure 1.

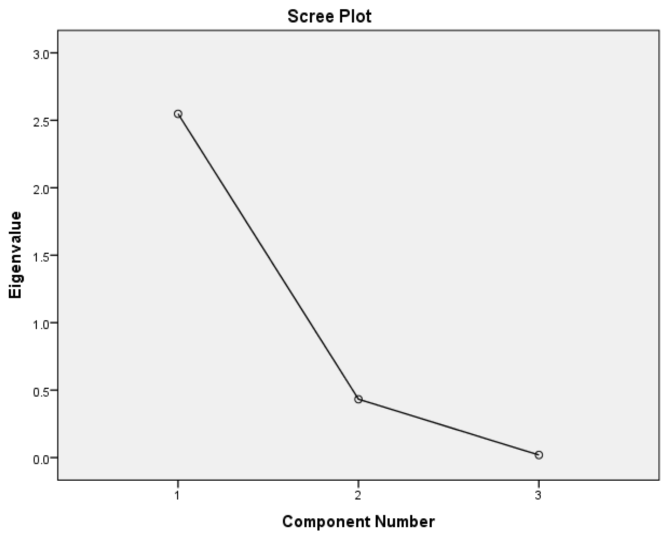

Figure1: Scree test plot of Borukva Algorithm.

5. CONCLUSION AND FUTURE WORKS

The efficiency with which the spanning is carried out often has a significant impact on the overall efficiency of a program. The efficiency of Borukva, Kruskal, Prims and Reverse - Delete spanning techniques in terms of running time, memory usage and number of edges was studied, experiments conducted and results subjected the factor analysis by SPSS.
The results were subjected to factor analysis using SPSS to test the level at which each of the factor affect the spanning techniques. Eigenvalues were used to indicate how well each of the decision variables fits the data from the experimental results. From the results obtained, the main factor affecting the spanning techniques was time taken to edge. It contributed 84.934 %, 58.669 %, 86.129 % and 82.327 % for Borukva, Kruskal, Prim and Reverse - Delete algorithm, respectively. The memory usage came second contributing 14.417 % for Borukva, 30.756 % for Kruskal, 13.058 % for Prim and 17.546 % for Reverse - Delete algorithm. The number of edges visited was the least of the factors contributing negligible percentage for the four spanning techniques; it contributed 0.649 % for Borukva algorithm, 10.575 % for Kruskal algorithm, 0.813 % for Prim algorithm and 0.128 % for Reverse - Delete algorithm.
Time taken to span is the main factor affecting spanning techniques. However, eigenvalues and the scree plot strengthen the fact that the running time and memory usage are the most important factors affecting the efficiency of the spanning techniques and Prim spanning technique is the most efficient spanning technique. Prim algorithm is the most efficient spanning technique among the four techniques used and the most critical factor among the three factors used was time taken. In future work, other techniques of factor analysis could be explored as well. Also, other programming languages apart from Java language should be put into consideration.

6. REFERENCES

[1] Ardhendu, M., Jayanta, D. and Pal S.C. (2012). A New Efficient Technique to Construct a Minimum Spanning Tree .International Journal of Advanced Research in Computer Science and Software Engineering.2: 92-97

[2] Olabiyisi S.O., Aladesae T.S., Oyeyinka F.I., and Oladipo Oluwasegun (2013): Evaluation of Critical Factors Affecting the Efficiency of Searching Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, vol.3 (7), pp.34-40.

[3] Wu, B. Y. and Chao, K.-M. (2004). Spanning trees and optimization problems, Discrete Mathematics and its Applications, Chapman & Hall/CRC, Boca Raton, FL, 2004. MR 2004i.90008 Zbl 1072.90047

[4] Nimesh, P. and Patel, K. M. (2015). A survey on Enhancement of Minimum Spanning Tree. Journal of Engineering Research and Applications. 5: 6 – 10

[5] Nesetril, J., Milkova, E., and Nesetrilova, H. (2001). "Otakar Bor°uvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history", Discrete Math. 233:1-3 , 3–36. MR 2002f:05053

[6] Karger, R. E. Tarjan, P. N., Klein, A., and David, R. (1995). "A randomized linear-time algorithm to find minimum spanning trees", Journal of the ACM-42.2 - , 321-328.

[7] Huang, R.M. (2011). An Improved Kruskal Algorithm—Two Branch Kruskal Algorithm. Chinese Scientific Papers Online, 1-13.

[8] Ardhendu, M., Jayanta, D. and Pal S.C. (2012). A New Efficient Technique to Construct a Minimum Spanning Tree. International Journal of Advanced Research in Computer Science and Software Engineering.2: 92-97

[9] Shivaani, K., Nithish, K. and Joshua, D. (2020). Performance Analysis of Minimum Spanning Tree. International Journal of Computer Engineering and Technology, 5(11):17-28.

[10] Folorunsho, O., Vincent, O., and Salako,O., (2010): An Exploratory Study of Critical Factors Affecting the Efficiency of Sorting Techniques (Shell, Heap and Treap), Anale, Seria informatica, vol. 8(1), pp.163-172.

[11] Akinyokun, O., Angaye, O., and Ubaru (2009): Factor Analysis of the Performance Indices and Communication Technology Projects in the Public Sector of the Nigerian Economy, Journal of Technology Research, Vol.1(5), pp.2-15.

[12] Olabiyisi S. O., Adetunji A. B., Oyeyinka F. I. (2012). Evaluation of the Critical Factors Affecting the Efficiency of Some Sorting Techniques. I.J. Modern Education and Computer Science, Vol 5, No 2, pp 25-33