Standard Scientific Research and Essays Vol 11(6): 170- 177, June, 2023 (ISSN: 2310-7502)
http://www.standresjournal.org/journals/SSRE

Research Article

Computational time complexity investigation of the various wireless network performance improvement algorithms: A superiority case of the Novel TGI Algorithm

***Nwogu Uchenna Obioma, Nwanwelu Udora Nwabuoku and Nwakamma Mary Nkechinyere**

[1]Department of Electrical and Electronics, Federal Polytechnic, Nekede, Imo State, Nigeria

*Corresponding Author E-mail: ucnwogu@fpno.edu.ng

Accepted 20, June, 2023

Abstract

In the world of data science and digital processing, there is a computational complexity: which could be in space or time; while accessing the algorithms to perform computational tasks. Time complexity is a type of computational complexity that describes the time it takes to execute an algorithm; time it takes to complete a statement in a single algorithm. The real essence of innovation in science and engineering is to make people's lives easier by providing solutions to problems that they face. To perform better, algorithms are written that are time efficient and that also utilize less memory. To this end, time complexities of the existing wireless network performance algorithms are analyzed to x-ray their computational efficiency with respect to time of execution vis a vis the novel TGI wireless protocol scheme. The performance analysis after their various code execution and implementation using the NS3.36 network simulator version shows that the time complexity of the novel TGI wireless optimization algorithm out performs the other wireless protocol algorithms in any wireless network.

Keywords: algorithm, wireless protocol, TGI wireless, computational complexity

INTRODUCTION

An Algorithm, in computer programming, is a finite sequence of well-defined instructions, typically executed in a computer, to solve a class of problems or to perform a defined task. There can be any number of ways, a specific set of instructions can be defined to perform the same task. Also, with options available to choose any one of the available programming languages, the instructions can take any form of syntax along with the performance boundaries of the chosen programming language.

Time complexity is the amount of time taken by an algorithm to run, as a function of the length of the input (Balabaskar, 2020). Here, the length of input indicates the number of operations to be performed by the algorithm. It is not going to examine the total execution time of an algorithm. Rather, it is going to give information about the variation (increase or decrease) in execution time when the number of operations (increase or decrease) in an algorithm. Importantly, the amount of time taken is a function of the length of input only. Furthermore, Time complexity measures the time taken to execute each statement of code in an algorithm. If a statement is set to execute repeatedly then the number of times that statement gets executed is equal to N multiplied by the time required to run that function each time (Balabaskar, 2020).

Time complexity as a QoS Parameter, checks the computational time it takes to run the various RAW Slot adjustment algorithms. A lesser time complexity value indicates less complex execution regime, therefore signals a less time to

execute the algorithm; and in the case of the RAW Slot algorithms: a lower registration time (reduction in the time of the stations to authenticate and associate).

Review of related works

This section reviewed some literatures that are related to time complexity and various impacts, in terms of performance in the optimization processes. Also discussed are the merits and demerits of the individual time complexities. There are different types of time complexities (in the ascending order of complexity) (Adrian, 2020)
1. Constant time – O(1)
2. Linear time – O(n)
3. Logarithmic time – O(log n)
4. Quadratic time – O(n^2)
5. Cubic time – O(n^3)

There are also more complex notations like Exponential time, Polynomial, Quasilinear time, factorial time, etc. which are used based on the type of functions defined.

Constant time – O (1)
An algorithm is said to have constant time with order O (1) when it is not dependent on the input size n. Irrespective of the input size n, the runtime will always be the same.

Linear time – O(n)
An algorithm is said to have a linear time complexity when the running time increases linearly with the length of the input. When the function involves checking all the values in an input data, such function has Time complexity with this order O(n). Thus, the function runs linearly with input size and this comes with order O (n).

Logarithmic time – O (log n)
An algorithm is said to have a logarithmic time complexity when it reduces the size of the input data in each step. This indicates that the number of operations is not the same as the input size. The number of operations gets reduced as the input size increases. Algorithms with Logarithmic time complexity are found in binary trees or binary search functions. This involves the search of a given value in an array by splitting the array into two and starting searching in one split. This ensures the operation is not done on every element of the data.

Quadratic time – O (n^2)
An algorithm is said to have a non – linear time complexity where the running time increases non-linearly (n^2) with the length of the input. Generally, nested loops come under this time complexity
order where for one loop takes O(n) and if the function involves loop within a loop, then it goes for O(n)*O(n) = O(n^2) order.
The order of growth for all time complexities are indicated in figure 1 below (Adrian, 2020)

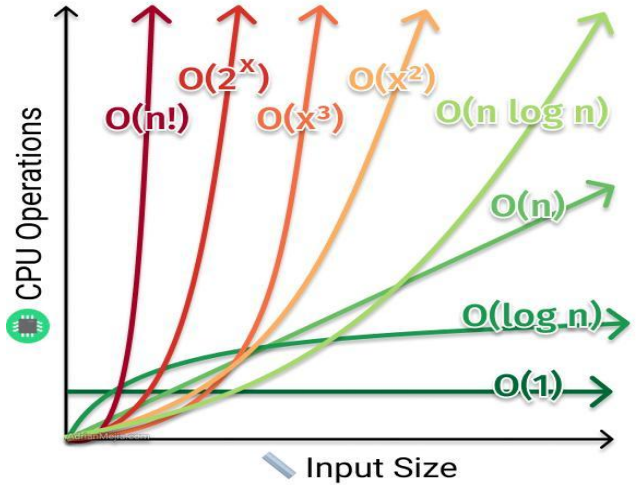


Figure 1. Time Complexity Graph

Thus, the above illustration gives a fair idea on how each function gets the order notation based on the relation between run time against the number of input data size and number of operations performed on them. The lower the time complexity function, the better the performance.

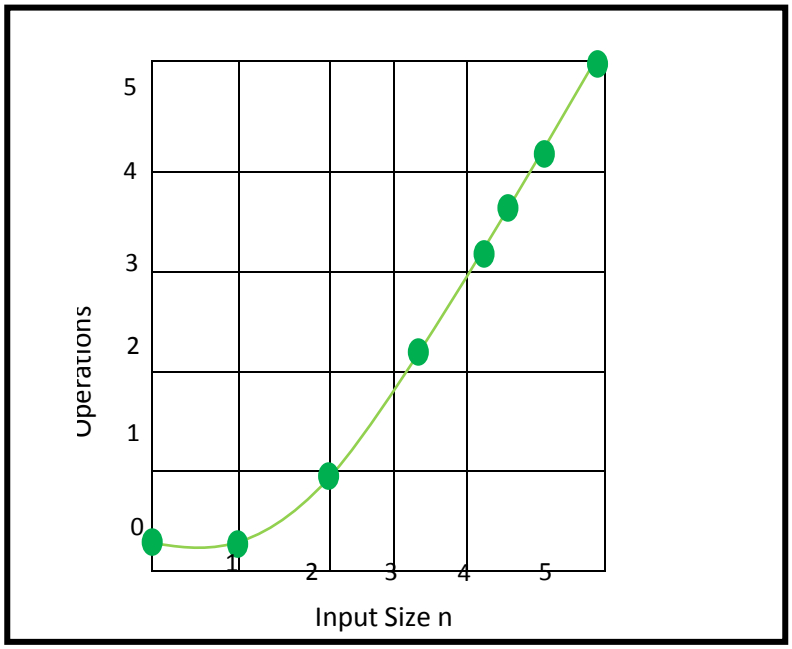
METHODOLOGY

System modeling of the time complexities of the various raw slot adjustment algorithms

The following RAW Slot algorithms are used: *RAd* (Pankaj, 2020), *TSG* (Stackoverflow.com, 2020), *HSCT* (Mike, 2021) *and MRA* (Bhatnagar et al., 2011) to compare with the improved RAW Slot TGI optimization algorithms: CVT-ME (Vassiliades et al., 2017) ETH (Slaoui et al., 2018) and Ada TDMA (Gohar et al., 2002). Firstly, a stability analysis of the algorithm is obtained. This is done by understanding the nature of the input data and parameters set up for the operation. The programming language used to complete this task is Python, while the input operational size ranges from n=1 to n = infinity.

Traffic station grouping TSG technique: greedy based algorithm

Greedy algorithms define a set of algorithms that solve a large number of problems using a similar strategy with a variety of time complexities (Stackoverflow.com, 2020). Therefore, the running time of it is consist of: Sorting the *n requests* in order, which yields O(nlogn). Constructing the array containing sorted requests, which yields O(n) and Iterating through the intervals in the array, which equals O(n). However, the second and the third step can often be merged into one step. Thus, the running time of the algorithm is O(nlogn + n) which is *O(nlogn)*.



Fig**ure** 2. TSG Time Complexity Graph

Raw Aid Backoff technique (RAd) Welsh Powell Algorithm

Welsh Powell Algorithm tends to execute operations on the pseudocode as follows:
It defines the degree of each vertex point. It lists the vertices in order of descending valence i.e. valence degree(v(i)) >= degree(v(i+1)) (Pankaj, 2020). It properly identifies the first vertex in the list. Then it gets down to the sorted list and identifies every vertex not connected to the vertex points. And crosses out those not connected/identified on the list. It repeats the process on the unmarked vertices with a new identifier and always working in descending order of degree until all vertices are identified. The time complexity of the algorithm is *O(N^2)* (Pankaj, 2020).

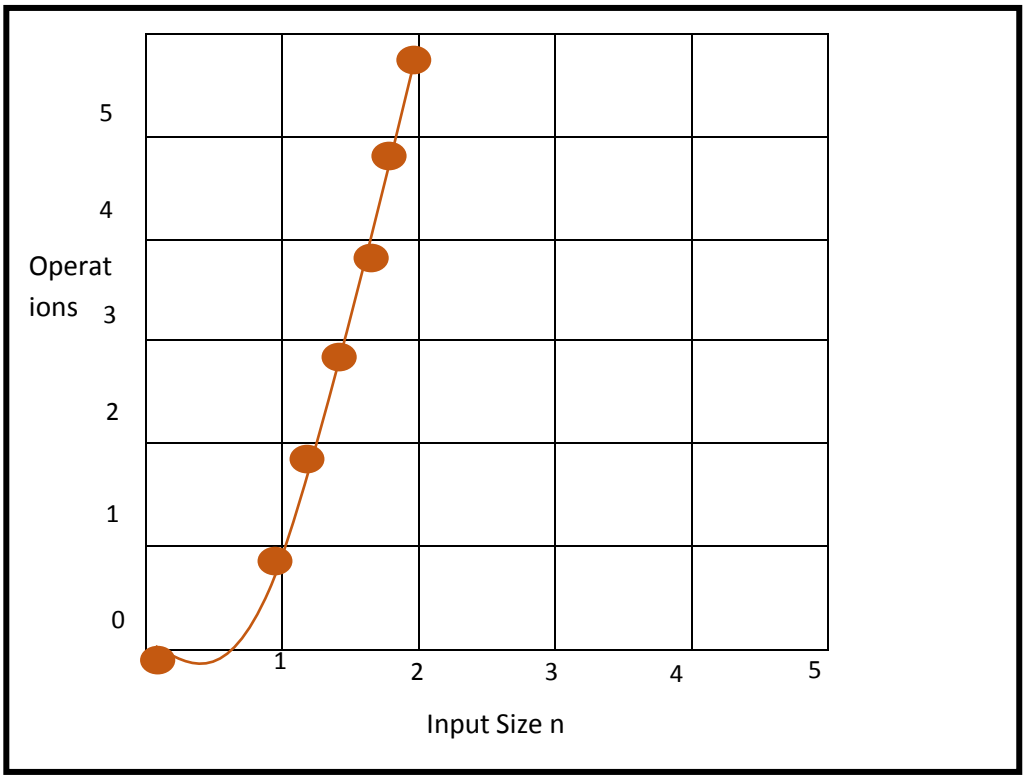


Figure 3. RAd Time Complexity Graph

Hybrid slotted CSMA/TDMA (HSTC): Up-down algorithm

This is a CAC based algorithm used in dynamic programming approach. It is used for impromptu mobile network associations (Mike, 2021). The time complexity of this (recursion + memorization, ie top-down) algorithm can be ascertained, by looking at the dp solution from a Fibonacci approach. Fibonacci for example can be calculated from Bottom Up: f[i] = f[i - 1] + f[i - 2]. Here, it is easy to see the complexity would be *O(n).* Or Top Down: f(n) = f(n - 1) + f(n - 2). Recursively, the complexity is *O(φn)* which will be reduced to O(n)with memorization (by knowing the *distinct states n*) (Mike, 2021). So, there is always a relation between complexity of top-down approach and the distinct states/sub problems.

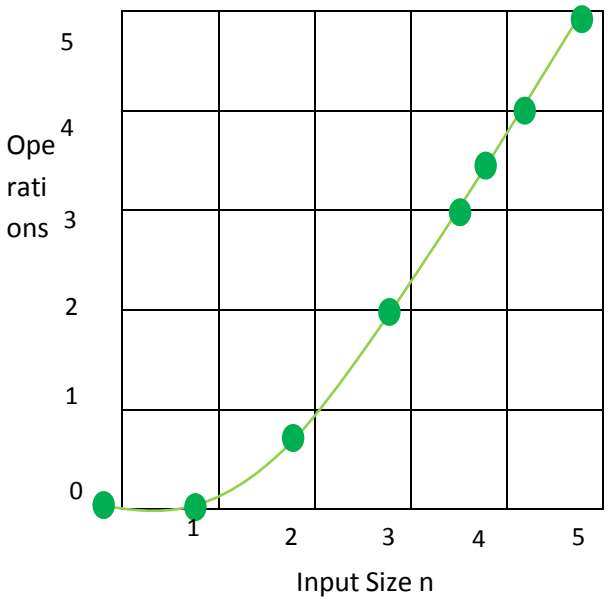


Figure 4. HSTC Time Complexity Graph

Markov chain AND M/G/I Model based raw slot allocation algorithm (MRA)

The complexity boundaries this model is obtained by counting the number of operations, which is a difficult task, and most of it is because it depends on the number of estimated parameters θ, the selected proposal distribution f, and the number of simulations m. Another perspective is to bound the complexity in terms of the chain's mixing time to its stationary distribution. (Bhatnagar et al., 2011) proved that in the established Markov chain the time t is close to stationary, and it is an NP-hard problem. Even so, (Jansen et al., 2012) using diffusion limits, gave a *lower bound O(n)* for a Metropolis-Hastings to converge to the stationary distribution, where d is the dimension of the parameter space. On the other hand, in cases of large data, (Roberts and Rosenthal, 2014) provided an *upper bound* $O(n^2)$, for a MH to converge to its stationary distribution. For MCMC methods with optimal jump adjustment such as MALA, (Jansen et al., 2012) provided a *lower bound* $O(n^{1/3})$ to converge.

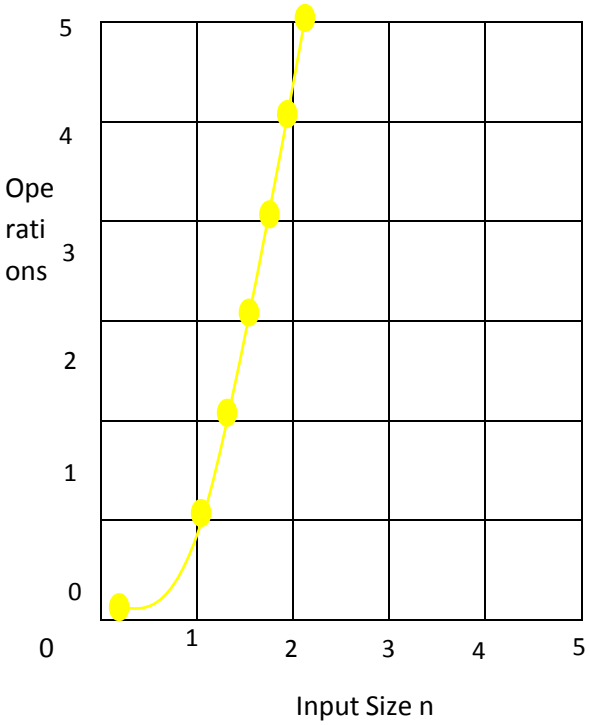


Figure 5. MRA Time complexity Graph

T**GI ALGORITHMS**

CENTROIDAL VORONOI TESSELLATIONS-MAP ELITE (CVT-ME ALGORITHM)

Sectorizing an IoT network in an effective way reduces the hidden node problem drastically. To sectorize the network area, we utilize the CVT-ME algorithm which sectorizes the AP coverage area based on the RAP location. The reason behind selecting this algorithm is that it provides high scalability in a dense network compared to the other sectorization algorithms. It splits the network area into the different voronoi sectors. The proposed CVT-ME algorithm initially identifies the centroid to construct the voronoi regions. We consider the RAP location as the centroid point to generate the voronoi sectors in the network. This proposed CVT-ME algorithm performs faster and provides better performance for a high scale network. The time complexity of this algorithm is *O(n)* where n represents the number of iteration. Hence, our work doesn't introduce any complex processing in sectorization of the network (Vassiliades et al ., 2017).

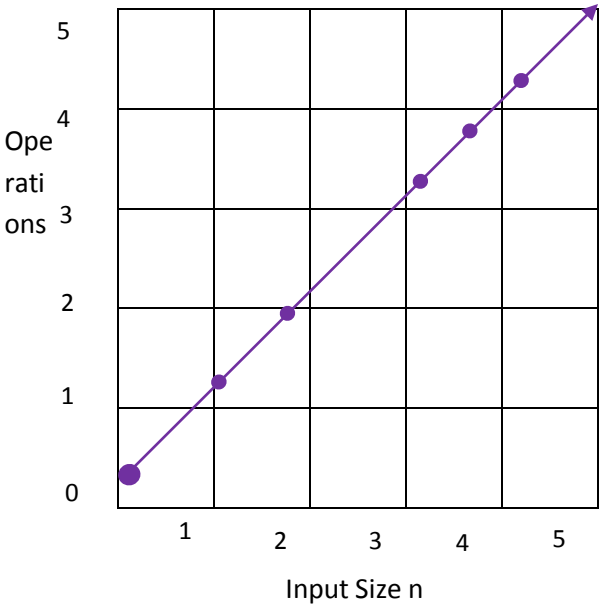


Figure 6. CVT Time Complexity Graph

ADAPTIVE TIME DIVISION MULTI ACCESS (AdaTDMA) ALGORITHM

We allocate RAW slots to each RAP through AP in an adaptive manner using the Ada-TDMA algorithm.
Our novelty is present in adjusting RAW slot sizes where we utilize VM-HMM based state prediction method. None of the techniques have performed VS-HMM (Variable Size Hidden Markov Model) based RAW slot adjustment procedures [6]. Ada-TDMA algorithm consumes less parameter to reduce the tedious processing in the existing state prediction algorithm. This algorithm estimates the state of each station in the group and how it relates with other stations in the sector in, an IEEE 802.11ah based IoT network. The time complexity of the VS-HMM: Ada-TDMA algorithm is *O(nT)* where n represents the number of states and T represents the sequence length [6]. The time complexity of the proposed markov model is less than the other existing markov models.

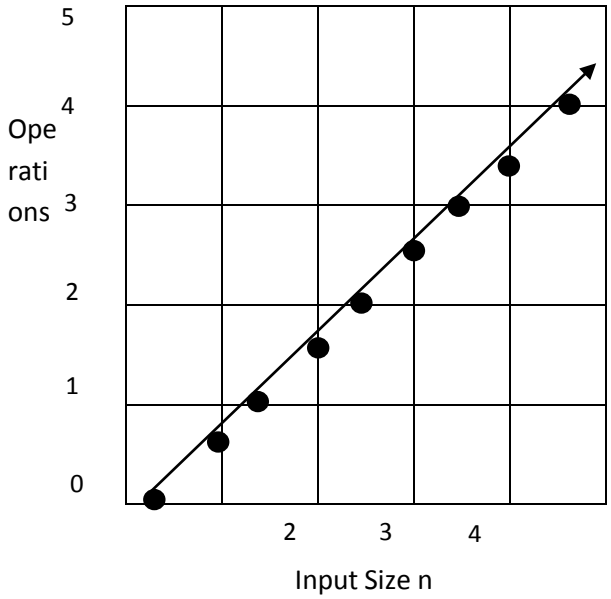


Figure 7. AdaTDMA Time Complexity Graph

ENHANCED TRANSITIVE HEURISTIC (ETH) ALGORITHM

With the use of these three metrics: delay, traffic demand and signal to noise ratio, the proposed ETH algorithm groups the stations within the sectors. It estimates the co-efficient between stations in order to group it accordingly (Slaoui et al.,2018). By analyzing this equation, the ETH algorithm identifies the similarity between the stations in the network. It estimates the performance of the co-efficient between two stations in a group. The stations that are having high co-efficient with each other are clustered into one group. Based on the estimated co-efficient value, it groups the stations that are in the same context with one other. This way of grouping the stations in the network evades the contention overhead and also improves the throughput in the network. The time complexity of the proposed ETH based grouping is *O(Zn)* Slaoui et al.,2018), where Z represents the number of clusters and n represents the number of iterations. Also, the proposed ETH algorithm is flexible in high dense network and also exhibit better performance Slaoui et al., 2018).

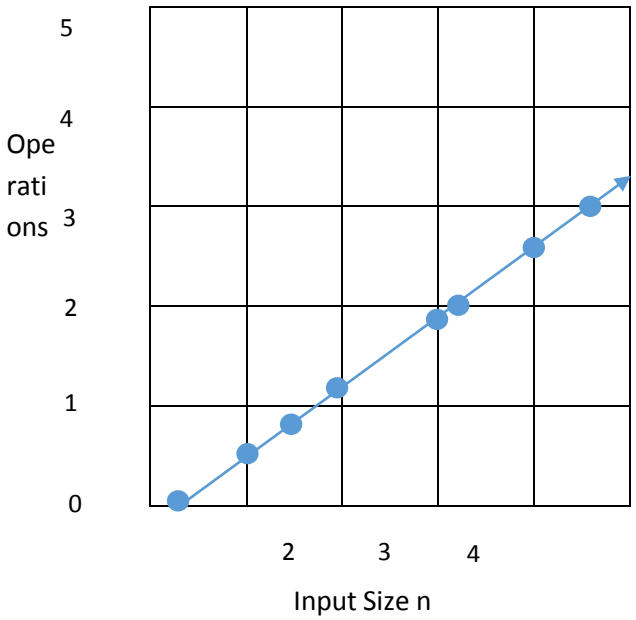


Figure 8. ETH Time Complexity Graph

RESULTS DISCUSSIONS AND CONCLUSIONS

In this section, we compare results of the time complexities of the existing RAW slot adjustment schemes such as RAd (Pankaj, 2020) TSG (Stackoverflow.com, 2020), HSCT (Mike, 2022) and MRA (Bhatnagar et al., 2011) with the proposed TGI optimization algorithms: CVT-ME [13], ETH [5] and Ada TDMA (Vassiliades, 2017)

A good algorithm executes quickly and saves space in the process. This can be shown from the shape of the graph as indicated in each algorithm. Figure 9 shows the performance graphs of the algorithms presently used for RAW Slot adjustment in IEEE 802,11ah networks.

Fig 2 shows the chart for the TSG (Greedy Algorithm). It involves a binary search operation, which takes a longer time, thus giving the linearithmic shape. It is still better than the quadratic complexity.

RAd in fig 3 uses the welsh powell algorithm and the MRA algorithms for Markov Chain operations and they have a growth rate of n2. So if the input size is 2, the operation will be 4 as indicated. For a big n (nesting of the loops), the time it takes to run or solve problems increases drastically.

HSTC in figure 4 uses the up and down algorithm which is linear as it visits every element from the input. It takes a proportionally longer time to compute as the input grows. From the plot of n input and the running time, a linear function graph is obtained.

The TGI algorithms: CVTE-ME, Ada TDMA and ETH are all linear algorithms like the HSTC. It is less complex to execute as the time of execution is linearly proportional to the number of input. The TGI algorithms would performed well despite the length n of the data inputted. On the other hand, RAd, MRA and TSG will perform poorly, significantly increasing computing time.

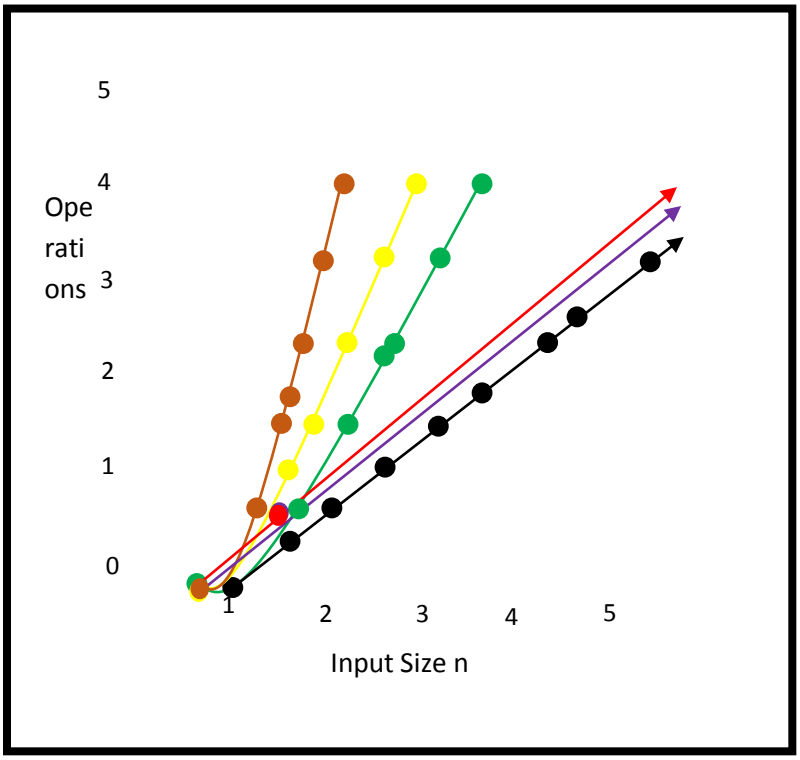


Figure 9. Combination of Raw Slot allocation algorithms time complexities

CONCLUSION

From this, we conclude that the time complexity of our proposed TGI algorithm is less compared to other grouping or RAW Slot Adjustment algorithms like MRA, HSTC, TSG and RAd. The TGI RAW grouping techniques is effective and thus optimizes the RAW Slot Adjustment Procedures in the IEEE 802.11ah network due to its less time complexity in the code execution. It also injects scalability in the administration of the algorithm even in high dense network environments.

Appreciation

Special gratitude to TETFUND and The Research/Development Unit of Federal Polytechnic Nekede Owerri for making this study a success.

References

Adrian M(2021). “Data Structures and Algorithms DSA” URL http: // www.adrianmeijia.com.
Balabaskar B(2020). “Data Science and Time Complexity “. URL http: // www.mygreatlearming.com
Belloni A, Chernozhukoy V(2009). “On the Computational Complexity of MCMC- Based Estimators in Large Samples. “The annals of statistics,37(4), 2011-2055. ISSN 00905364.
Bhatnagar N, Bogdanov A, Mossel E (2011). “The Computational Complexity of Estimating MCMC Convergence Time.http: //www.link.springer.com
Gohar A, Kyong HK, Ki-II K(2002). Adaptive TDMA Scheduling for Real-Time Flows in Cluster-Based Wireless Sensor Networks. Computer Science and Information System 13(2):475-492
Izhar AM(2020). “An introduction to computational complexity in markov chain methods, Hand book of markov chain monte carlo”. CRC press.
Jansen K, Ravi R, Rolim JDP(2012). Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Pp. 424–435. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-22935-0.
Mike M(2021). “smart up-smart down algorithms”. URL http: // www.codeforces.com
Pankaj s(2020). “welsh algorithms: Graph colony approach”. URL http: //www.iq.open genius.org
Roberts GO, Rosenthal JS(2014). “Complexity Bounds for MCMC via Diffusion Limits.”1411.0712 URL http://www.jstor.org/stable/30243694
Slaoui SC, Dafir Z, Lamari Y(2018). E-Transitive: an enhanced version of the Transitive heuristic for clustering categorical data. Procedia Computer Science. 127:26–34.
Vassiliades V, Chatzilygeroudis K, Mouret JB(2017).Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm. IEEE Transactions on Evolutionary Computation, Pp. 1–1.
www. Stackoverflow.com; Greedy Based Algorithms. 2020