International Journal of Computing Sciences Research (ISSN print: 2546-0552; ISSN online: 2546-115X)
Vol. 8, pp. 3298-3310
doi: 10.25147/ijcsr.2017.001.1.211
https://stepacademic.net

Short Paper

POT-AVL: A Novel CPU Scheduling Algorithm based on AVL Trees and Postorder Traversal

Don Harl C. Malabanan
College of Information Technology and Computer Science,
University of the Cordilleras, Philippines
dcmalabanan@uc-bcf.edu.ph
(corresponding author)

Mishael M. Valdez
College of Information Technology and Computer Science,
University of the Cordilleras, Philippines
mmvaldez@uc-bcf.edu.ph

Dionisio R. Tandingan Jr.
College of Engineering and Architecture,
University of the Cordilleras, Philippines
drtandingan@uc-bcf.edu.ph

Date received: May 17, 2024
Date received in revised form: July 26, 2024; July 29, 2024
Date accepted: August 17, 2024

Recommended citation:

> Malabanan, D. H., Valdez, M. M., & Tandingan, D. Jr. (2024). POT-AVL: A novel CPU scheduling algorithm based on AVL trees and postorder traversal. *International Journal of Computing Sciences Research*, *8*, 3298-3310. https://doi.org/10.25147/ijcsr.2017.001.1.211

Abstract

Purpose – This study aims to offer a new perspective on the development and optimization of CPU scheduling algorithms in the field of research utilizing the concept of an Adelson-Velsky and Landis (AVL) tree which has not been used before in related studies which signifies a departure from standard practices, seeking to offer fresh insights into scheduling challenges.

Method – A novel scheduling algorithm called POT-AVL encompasses the structure of an

AVL tree while using the postorder traversal to identify and select which processes shall be chosen and executed by the scheduler. The proposed algorithm was tested against the more common FCFS and two optimized RR algorithms, AMRR and MMRRA in terms of their Average Turnaround Time, Average Waiting Time, and Context Switch metrics.

Results –The results show that POT-AVL consistently performs better than the other algorithms in instances when the burst times for the processes are long burst times. POT-AVL performs worse than the FCFS algorithm when there are long gaps between arrival times.

Conclusion – The novel approach of integrating an AVL tree wait queue leads to an improved efficiency in terms of searching and managing processes in the queue which may be useful as a new path in the development and optimization of CPU scheduling algorithms.

Recommendations – The inclusion and other factors such as quantum time, and priority level, among others, can identify the strengths and weaknesses of the proposed algorithms in different scenarios.

Research Implications – This study exhibits more possibilities for amalgamating data structures and CPU scheduling algorithms.

Practical Implications – This study could suggest exploring alternative balancing techniques or adapting AVL trees to leverage hardware features efficiently.

Keywords – CPU Scheduling, novel approach, AVL, Optimization

INTRODUCTION

The use of CPU scheduling in determining the order of processes executed in the CPU is a topic in computer science full of unique and new advancements. The wave of studies investigating advancing systems lends itself to the need to continually improve system throughput and wait times by looking at different problems through a new lens (Pemasinghe & Rajapaksha, 2022). As such, the researchers have aimed to develop a new scheduling algorithm by looking at the problem from a different perspective.

To offer a new perspective on the problem, the researchers utilized the concept of an Adelson-Velsky and Landis (AVL) tree which has not been used before in related studies (Mishra & Ofujeh Ahmed, 2020). This choice signifies a departure from standard practices, seeking to offer fresh insights into scheduling challenges. An AVL tree is a data structure in the form of a binary tree that can rebalance itself if the tree becomes imbalanced. There are three common ways to traverse an AVL tree for a system to process the nodes in the tree: inorder, preorder, and postorder. Each traversal method reads the tree in a different

manner which lends itself to multiple possible uses within the domain of computer science and beyond.

The researchers have observed that one method in optimizing the algorithm was selecting the order in which process to run (Dwibedy & Mohanty, 2023) which supports the decision to recontextualize a data structure concept into CPU scheduling. Throughout this paper, the researchers have aimed to develop a new CPU scheduling Algorithm based on AVL trees and postorder traversal and find out whether the newly developed algorithm was competitive against other modern algorithms.

LITERATURE REVIEW

The development of scheduling algorithms would typically focus on optimizing the way jobs are added to the ready queue (Jeyaprakash & M, 2021). Multiple approaches could be taken to optimize this process such as the Adjustable Time Slice (ATS) algorithm which was made as an improvement to the round-robin algorithm by creating multiple queues based on multiple metrics and then assigning a different time quantum (TQ) for each queue for a more efficient and adaptive approach (Mostafa et al., 2022). Another approach named the VRRP developed an adaptive priority algorithm that made sure to give priority to new processes while dynamically changing the priority of older processes based on a computed ratio of waiting time and remaining burst time to minimize the waiting time (Singh et al., 2015).

Multiple unique approaches attempt to solve the ever-present challenge of discovering the most efficient approach for CPU scheduling problems. The researchers' approach of using an AVL tree specifically as the scheduling logic of the algorithm is novel to the point that no similar studies could be found by the researchers. The development of the AVL self-balancing algorithm first introduced in a seminal paper by Adelson-Velsky & Landis (1962) shows promise in the potential benefits of improving the process of CPU scheduling as it has been used to improve various processes in other domains such as IoT (Canli & Toklu, 2021), computer vision (Chan IV, 2021), and in embedded systems (Lázaro et al., 2021).

The evaluation of CPU scheduling algorithms often hinges on key performance metrics like efficiency, throughput, turnaround time (TAT), waiting time (WT), response time, and fairness. Various similar studies (Al-Khatib et al, 2023) have benchmarked common using these criteria. In optimizing these algorithms, the selection of performance metrics—primarily TAT and WT—is crucial, as these indicate the efficiency of task execution and the delay before tasks begin execution, respectively. Additionally, the number of Context Switches (CS) is a vital metric for assessing CPU utilization, underscoring the importance of comprehensive performance evaluation (Al-Safar, 2021).

The researchers have evaluated various scheduling algorithms against new ones,

highlighting First Come First Served (FCFS) for its prominence in optimizing legacy systems. Despite its challenges, such as potential system slowdowns under heavy workloads leading to increased process idle times and CPU execution delays, FCFS remains a focal point of the study (Vayadande et al., 2023). Notably, some studies (Panda et al., 2023) have observed FCFS's advantages in mitigating such issues in specific contexts like cloud computing.

The Round Robin (RR) algorithm's variable performance has also spurred research into its optimization. Efforts range from modifying the static quantum time formula (Banerjee et al., 2012; Mora et al, 2020; Sakshi et al., 2022) to incorporating features like multiple queues and dynamic quantum times (Biswas et al, 2023; Manuel et al., 2019; Niranjan & Thenmozhi, 2023). Notably, the Average Max Round Robin (AMRR) and Modified Median Round Robin Algorithm (MMRRA) enhance RR's stability, positioning them as competitive alternatives. AMRR dynamically adjusts the Time Quantum (TQ) based on average and maximum burst times in the queue, while MMRRA's TQ varies according to the median and highest burst times, calculated using a specific formula.

METHODOLOGY

The following section introduces how the POT-AVL algorithm works using a sample test case. The step-by-step algorithm with its corresponding flowchart outputs the calculated ATAT and AWT of the processes as well as the Gantt chart showcasing how each process is executed as per their AT and BT. The comparison to the other scheduling algorithms will be done using 5 different test cases which have different inputs with varying BTs and ATs.

A. POT-AVL Algorithm and Flowchart

Let Q be the ready queue. Let A be the AVL queue.

1) When processes arrive, add them to A using a First-Come-First-Served algorithm.
2) If multiple processes arrive simultaneously, add them in the order of their Process IDs (PID), with the lower PID being added first.
3) After insertion, allow the AVL tree to self-balance
4) Use Postorder traversal on A. The first process encountered during this traversal is the one selected for execution. This process is then moved to Q.
5) Execute the full burst time of the current process at Q
6) During execution, if new processes arrive, repeat Step 2 to add and manage these new arrivals in
7) After a process is executed, check if there are any remaining processes to be executed
8) If all processes are executed, then EXIT
9) else, go to step 2

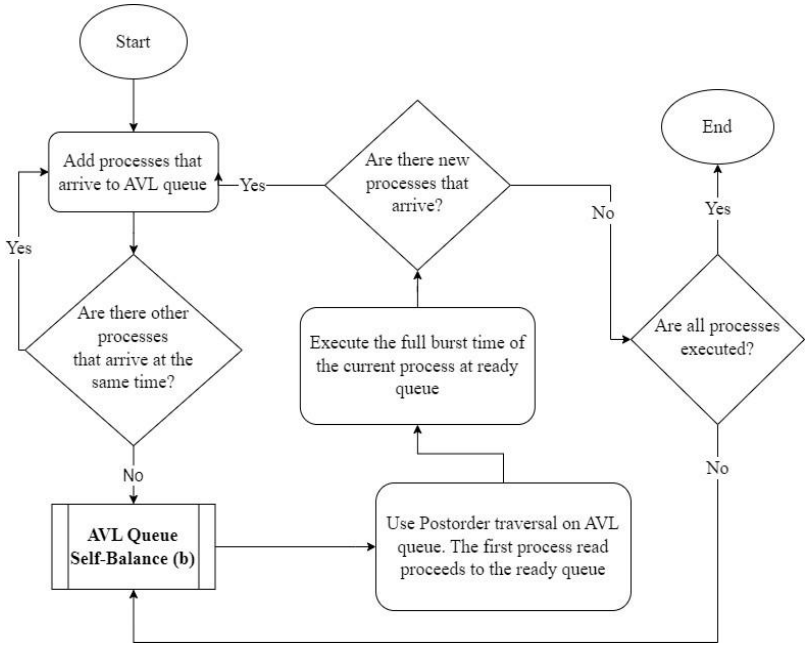

Figure 1. POT-AVL Flowchart.

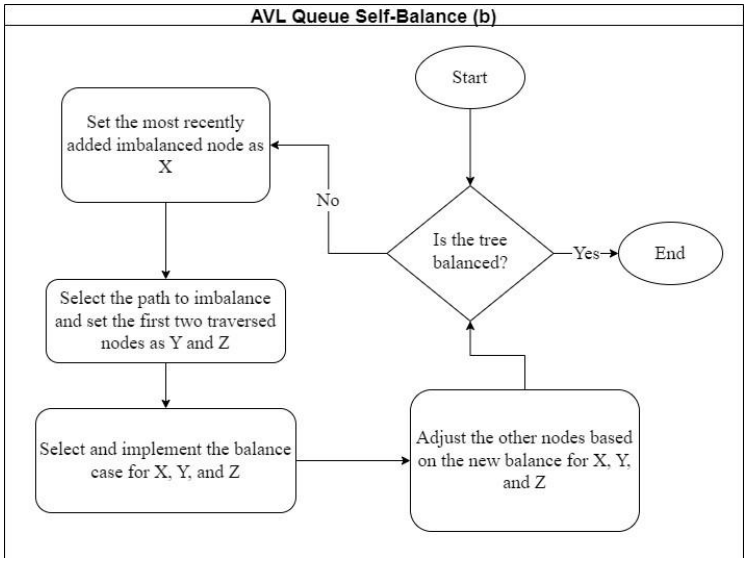

Figure 2. AVL Queue Self-Balance Flowchart.

Table 1. Sample Processes

PID	BT	AT	*TAT*	WT
Process 1	8	0	8	0
Process 2	12	2	56	44
Process 3	6	4	10	4
Process 4	10	7	24	14
Process 5	15	10	36	21
Process 6	7	13	8	1
Average			23.67	14

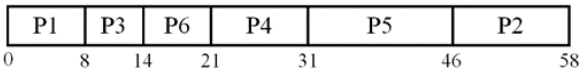

Figure 3. Gantt Chart for Sample Test Case.

The POT-AVL algorithm proceeds as follows based step-by-step procedure shown in Figure 1 and Figure 2 using the sample processes provided in Table 1. Process P1 arrived first with an arrival time of 0. Since no other processes arrived at the same arrival time, P1 goes to the AVL queue and proceeds with the Postorder traversal. P1 then goes to the ready queue and executes its burst time of 8 units. During the non-preemptive execution of P1, the processes P2, P3, and P4 arrive by the 2nd, 4th, and 7th-time units respectively. Processes P2, P3, and P4 proceed to the existing AVL queue (which is currently empty) and proceed to self-balance its nodes.

The Postorder traversal outputs P3 as the first process to be read, therefore, P3 proceeds to the ready queue. P3 executes for the burst time of 6 units, during which, P5 and P6 arrive at the 10th and 13th time unit respectively. P5 and P6 proceed to the existing AVL queue and it self-balances the nodes. The postorder traversal outputs P6 as the next process to proceed to the ready queue which executes for 7 units. The AVL queue then self-balances itself and keeps outputting the next processes which are P4, P5, and P2 respectively. The CT, TAT, and WT were then calculated which also gives the output of 23.67 units for the Average TAT (ATAT) and 14 units for the Average WT (AWT). The final Gantt chart of the processes can be seen in Figure 3.

The POT-AVL algorithm was benchmarked against FCFS and two optimized RR variations, AMRR and MMRRA, due to FCFS's prevalence in scheduling algorithm research and the latter's innovative TQ computations. This comparison ensures that POT-AVL is evaluated alongside both traditional and contemporary counterparts.

RESULTS

The performance metrics chosen to identify the viability of well-performing scheduling were the ATAT and the AWT of all the processes executed to observe the scheduling algorithms in terms of efficiency of task execution and reduction of wasted time respectively. The identified performance metrics were used as the basis for the viability of the test cases.

All experimental test cases performed in this study follow the assumptions listed in the scope and delimitations. The test cases are grouped by their corresponding length of burst times and the length of gaps in the arrival times. The performance metrics, namely average turnaround time and average waiting time (Manuel et al., 2019) of the FCFS algorithm, the AMRR algorithm, and the MMRRA (Sakshi et al., 2022) algorithm were compared to the POT-AVL algorithm (Table 2).

1) Test Case 1: The researchers assumed 5 processes with short burst times and short arrival times.
2) Test Case 2: The researchers assumed 5 different processes with long burst times and short arrival times.
3) Test Case 3: The researchers assumed 5 processes with short burst times and long arrival times.
4) Test Case 4: The researchers assumed 5 different processes with long burst times and long arrival times.
5) Test Case 5: The researchers assumed 5 different processes with randomized long burst times and arrival times.

Table 2. POT-AVL Test Cases 1-5

Test Case	PID	BT	AT	*TAT*	WT
Test Case 1	Process 1	4	3	10	6
	Process 2	6	5	14	8
	Process 3	2	1	8	6
	Process 4	7	0	7	0
	Process 5	5	4	20	15
	Average			11.8	7
Test Case 2	Process 1	24	2	24	0
	Process 2	42	4	111	69
	Process 3	19	8	37	18
	Process 4	32	3	144	112
	Process 5	28	7	66	38
	Average			76.4	47.4
Test Case 3	Process 1	12	5	12	0
	Process 2	4	11	10	6
	Process 3	7	41	7	0
	Process 4	9	27	9	0
	Process 5	5	19	7	2
	Average			9	1.6
Test Case 4	Process 1	44	0	44	0
	Process 2	27	65	38	11
	Process 3	51	82	72	21
	Process 4	32	30	46	14
	Process 5	48	17	185	137
	Average			77	36.6
Test Case 5	Process 1	18	7	62	44
	Process 2	33	0	33	0
	Process 3	17	17	69	52
	Process 4	13	23	23	10
	Process 5	5	10	41	36
	Average			45.6	45.6

For test case 1, the POT-AVL algorithm performed better than AMRR and the MMRRA with a positive difference of 2.4 units for the ATAT and AWT metrics. However, POT-AVL performed slightly worse compared to FCFS with only a negative difference of -0.2 units. The context switches are relatively similar with POT-AVL and FCFS both having 4 and AMRR and MMRRA having 5. For test case 2, when having long burst times with short arrival times, the POT-AVL algorithm performed more efficiently than all other scheduling systems for comparison with both the ATAT and AWT metrics being significantly lower.

Similarly, with test case 1, the context switches are relatively similar with POT-AVL and FCFS both having 4 and AMRR and MMRRA having 5. For test case 3, with the processes having shorter burst times and a longer gap for arrival times, the POT-AVL performed slightly better compared to AMRR and MMRRA with a positive difference of 0.2 units and 0.4 units respectively. However, the algorithm had the same performance as FCFS with both having an ATAT value of 9 units and an AWT value of 1.6 units. The context switches have a significant difference with POT-AVL and FCFS both having 4 and AMRR and MMRRA having 7.

Table 3. Algorithm Comparison for Test Cases 1-5

Test Case	Algorithm	ATAT	ART	CS
Test Case 1	POT-AVL	11.8	7	4
	FCFS	11.6	6.8	4
	AMRR	14.2	9.4	5
	MMRRAA	14.2	9.4	5
Test Case 2	POT-AVL	76.4	47.4	4
	FCFS	87	58	4
	AMRR	94	65	5
	MMRRAA	93.2	64	5
Test Case 3	POT-AVL	9	1.6	4
	FCFS	9	1.6	4
	AMRR	9.4	2	7
	MMRRAA	9.2	1.8	7
Test Case 4	POT-AVL	45.6	28.4	4
	FCFS	48.4	31.2	4
	AMRR	52.6	35.4	6
	MMRRAA	51.8	34.6	6
Test Case 5	POT-AVL	77	36.6	4
	FCFS	83.8	43.4	4
	AMRR	104	63.6	5
	MMRRAA	105	64.2	5

For test case 4, similar to test case 2, having processes with long burst times and arrival times with long gaps results in the POT-AVL algorithm performing more efficiently than all

other scheduling systems for comparison with both the ATAT and AWT metrics being significantly lower. The context switches have a significant difference with POT-AVL and FCFS both having 4 and AMRR and MMRRA having 6. Lastly, test case 5 shows that POT-AVL performed better than the other algorithms during a randomized set of BTs and ATs. The context switches are relatively similar with POT-AVL and FCFS both having 4 and AMRR and MMRRA having 5.

DISCUSSION

The test cases reveal that the newly developed POT-AVL scheduling algorithm can contend against other common and contemporary scheduling algorithms such as FCFS, AMRR, and MMRRA. The POT-AVL algorithm consistently performs better than the AMRR and MMRRA scheduling algorithms in terms of Average Turnaround Time (ATAT) and Average Waiting Time (AWT), especially in scenarios where the processes have long burst times, as seen in test case 2 and test case 4. This indicates that the POT-AVL algorithm is highly effective in environments with heavy computational loads, ensuring minimal delays and efficient processing. In scenarios where the processes have long gaps in their arrival times, the FCFS scheduling algorithm stands a chance to be more efficient compared to the POT-AVL algorithm in terms of ATAT and AWT, as seen in test case 1 and test case 3. This suggests that the POT-AVL algorithm might not handle idle times between processes as effectively as FCFS, leading to increased waiting times in such scenarios.

The strengths of the POT-AVL scheduling algorithm are its efficiency with long burst times, competitive performance against some contemporary algorithms, and its consistency in providing reliable performance across different test cases which indicates stability in differing conditions. This performance aligns with previous studies that have utilized AVL to improve CPU computation efficiency (Lazaro et al., 2021). Its weaknesses lay in its handling of idle times and larger space complexity which leads to lesser efficiency in handling long-gapped scenarios and in managing memory respectively.

CONCLUSIONS AND RECOMMENDATIONS

Testing the novel POT-AVL algorithm against FCFS, AMRR, and MMRRA with the selected performance metrics revealed a few key aspects where the newly developed algorithm excels. POT-AVL consistently performed better than others at tasks with long burst times. FCFS lightly outperforms POT-AVL in cases with shorter burst times. This shows the ability of POT-AVL to be optimized in tasks with heavier loads. POT-AVL also works better in terms of switch cases as it performs with the least possible switch cases for every Gantt chart with the formula no. of switch cases = p_n where p_n is the number of processes.

The researchers recommend the inclusion and other factors such as quantum time, and priority level, among others, can identify the strengths and weaknesses of the proposed algorithms in different scenarios. Other possible recommendations include having different novel approaches as a comparison for test cases to indicate the level of

progression POT-AVL has to offer in terms of CPU utilization.

IMPLICATIONS

The innovative development of the POT-AVL scheduling algorithm impacts multiple fields of computer science in terms of OS management. These include improved efficiency as self-balancing binary search trees provide efficient searching and retrieval operations. Integrating an AVL tree into the wait queue could lead to improved efficiency in terms of searching for and managing processes in the queue. This could result in faster scheduling decisions and overall system performance. Enhanced scalability is also a significant factor for impact as AVL trees are known for maintaining a balanced structure, which helps in maintaining a logarithmic height. This property can enhance the scalability of the scheduling algorithm, making it suitable for systems with many processes. Other impacts include the optimization of the utilization of system resources, ensuring that processes are scheduled in a way that minimizes resource contention and maximizes throughput and more efficient handling of processes in terms of priority or other scheduling criteria which could potentially reduce waiting times for processes, improving the responsiveness of the system. This study exhibits more possibilities for the amalgamation of data structures and CPU scheduling algorithms.

ACKNOWLEDGEMENT

The researchers would like to acknowledge the assistance and funding given by the University of the Cordilleras, Baguio City.

FUNDING

This study is funded by the University of the Cordilleras, Baguio City.

DECLARATIONS

Conflict of Interest

The researchers declare no conflict of interest in this study.

Informed Consent

Not applicable due to the novelty approach.

Ethics Approval

Not applicable due to lack of ethical issues.

REFERENCES

Adelson-Velsky, E. M., & Landis, E. M. (1962). *An algorithm for the organization of information*. Soviet Math.

Al-Khatib, R. M., Al-Khateeb, A., Al-Daom, E., Al-Dagamseh, I. T., Tawalbeh, A., & Abualigah, L. (2023). A new enhanced IGBTQ-based model for CPU scheduling. *Applied and Computational Engineering*, *8*(1), 411–417. doi:10.54254/2755- 2721/8/20230207

Al-Safar, A. (2021). Hybrid CPU scheduling algorithm SJF-RR in static set of processes. *Journal of Al-Rafidain University College for Sciences*, (1), 36–60. https://doi.org/10.55562/jrucs.v29i1.377

Banerjee, P., Banerjee, P., & Dhal, S. S. (2012). Comparative performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using dynamic time quantum with round robin scheduling algorithm using static time quantum. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *1*(3), 56–62.

Biswas, D., Samsuddoha, Md., Al Asif, Md. R., & Ahmed, Md. M. (2023). Optimized round robin scheduling algorithm using dynamic time quantum approach in cloud computing environment. *International Journal of Intelligent Systems and Applications*, *15*(1), 22–34. https://doi.org/10.5815/ijisa.2023.01.03

Canli, H., & Toklu, S. (2022). AVL-based settlement algorithm and reservation system for smart parking systems in IoT-based smart cities. *The International Arab Journal of Information Technology*, *19*(5), 793-801. https://doi.org/10.34028/iajit/19/5/11

Chan, C. VI (2021). *Computer vision: Object tracking using self-balancing tree for optimized data processing* (unpublished manuscript). California State University, Fullerton, USA. https://doi.org/10.5281/zenodo.5765023

Dwibedy, D., & Mohanty, R. (2023). A note on hardness of multiprocessor scheduling with scheduling solution space tree. *Computer Science*, *24*(1), 53-74. https://doi.org/10.7494/csci.2023.24.1.4656

Jeyaprakash, T., & Sambath, M. (2021). Performance analysis of CPU scheduling algorithms – a problem-solving approach. *International Journal of Science and Management Studies (IJSMS)*, 411–416. https://doi.org/10.51386/25815946/ijsms-v4i4p138

Lázaro, J., Bidarte, U., Muguira, L., Cuadrado, C., & Jiménez, J. (2021). Fast and efficient address search in system-on-a-programmable-chip using binary trees. *Computers & Electrical Engineering, 96,* Article ID 107403. https://doi.org/10.1016/j.compeleceng.2021.107403

Manuel, J. I., Baquirin, R. B., Guevara, K. S., & Tandingan, D. R. (2019). Fittest job first dynamic round robin (FJFDRR) scheduling algorithm using dual queue and arrival time factor: A Comparison. *IOP Conference Series: Materials Science and Engineering*, *482*, Article 012046. https://doi.org/10.1088/1757-899x/482/1/012046

Mishra, A., & Ofujeh Ahmed, A. (2020). Simulation of CPU scheduling algorithms using Poisson distribution. *International Journal of Mathematical Sciences and Computing*, *6*(2), 71–78. https://doi.org/10.5815/ijmsc.2020.02.04

Mora, H., Abdullahi, S. E., & Junaidu, S. B. (2017, November). Modified median round-robin algorithm (MMRRA). In *Proceedings of the 13th International Conference on Electronics, Computer and Computation (ICECCO)* (pp. 1-7). IEEE.

https://doi.org/10.1109/icecco.2017.8333325
Mostafa, M. S., Ahmed Idris, S., & Kaur, M. (2022). ATS: A novel time-sharing CPU scheduling algorithm based on features similarities. *Computers, Materials & Continua*, *70*(3), 6271–6288.
Niranjan, P., & Thenmozhi, M. (2023, December). Priority-induced round-robin scheduling algorithm with dynamic time quantum for cloud computing environment. In *Proceedings of the 2023 Second International Conference on Advances in Computational Intelligence and Communication (ICACIC)* (pp. 1–5). IEEE. https://doi.org/10.1109/icacic59454.2023.10435310
Panda, S. K., Dhiman, A., & Bhuriya, P. (2023, July). Efficient real-time task-based scheduling algorithms for IOT-fog-cloud architecture. In *Proceedings of the 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-7). IEEE. https://doi.org/10.1109/icccnt56998.2023.10306689
Pemasinghe, S., & Rajapaksha, S. (2022, September). Comparison of CPU scheduling algorithms: FCFS, SJF, SRTF, round robin, priority-based, and multilevel queuing. In *Proceedings of the 2022 IEEE 10th Region 10 Humanitarian Technology Conference (R10-HTC)* (pp. 318–313). IEEE. https://doi.org/10.1109/r10-htc54060.2022.9929533
Sakshi, Sharma, C., Sharma, S., Kautish, S., A. M. Alsallami, S., Khalil, E. M., & Wagdy Mohamed, A. (2022). A new median-average round robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time. *Alexandria Engineering Journal*, *61*(12), 10527–10538. https://doi.org/10.1016/j.aej.2022.04.006
Singh, P., Pandey, A., & Mekonnen, A. (2015). Varying response ratio priority: A preemptive CPU scheduling algorithm (VRRP). *Journal of Computer and Communications*, *3*(04), 40–51. https://doi.org/10.4236/jcc.2015.34005
Vayadande, K., Sheth, P., Pawal, D., Pathak, A., Paralkar, K., & Patil, S. (2023, January). Simulation of CPU scheduling algorithms for efficient execution of processes. In *Proceedings of the 2023 International Conference for Advancement in Technology (ICONAT)* (pp. 1–6). IEEE. https://doi.org/10.1109/iconat57137.2023.10080113

Author's Biography

Don Harl Malabanan is an instructor at the University of the Cordilleras College of Information Technology and Computer Science. He holds a Bachelor's Degree in Information Technology from the same institution. His academic focus includes Machine Learning, Data Structures, and Programming. Currently, he is pursuing a Master of Science in Computer Science, further enhancing his expertise in these areas.

Mishael M. Valdez is an instructor at the University of the Cordilleras College of Information Technology and Computer Science. He holds a Bachelor of Science in Computer Science from the same institution. His research and academic interests include Machine Learning, Natural Language Processing, and Data Science. As an instructor, he utilizes his knowledge and skills to teach Computer Science and Multimedia Arts courses. Currently, he is pursuing his Master of Science in Computer Science to enhance his knowledge and expertise in this field of academics.

Dionisio Tandingan Jr. is an instructor at the University of the Cordilleras College of Engineering and Architecture. He holds a Bachelor of Science in Computer Engineering, Master of Computer Science, and Master of Arts in Education, Major in Educational Management.