Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 10s (2025)

Optimization of Round-Robin Arbitration for Low Power

[1]Jeetu Debbarma, [2]Rita Banik
[1]PhD student, ECE Deptt., ICFAI University, Tripura (W), Associate Professor, ECE, Tripura Institute of Technology, Narshingarh, Tripura (W)
[2]Associate Professor, EE Deptt., ICFAI University, Tripura (W)

Article History:

Received: 12-01-2025

Revised: 15-02-2025

Accepted: 01-03-2025

Abstract:

This paper proposes an energy-efficient task scheduling strategy by optimizing the Round-Robin (RR) arbitration mechanism using a hybrid Artificial Bee Colony with Whale Optimization Algorithm (ABC & WOA) in conjunction with Dynamic Voltage Scaling (DVS)**.** While RR scheduling is widely used for its simplicity and fairness, it typically lacks energy awareness—an increasingly critical concern in energy-constrained and battery-powered embedded systems. This study addresses this gap by incorporating DVS and optimized idle-time sleep states to significantly reduce both total energy consumption and average energy per task. The proposed hybrid ABC-WOA algorithm demonstrates superior performance over its individual counterparts (ABC and WOA), offering improvements in energy efficiency**,** convergence speed**,** task completion rate**,** and execution time. The experimental results validate that the integrated DVS and ABC-WOA approach is a robust and scalable solution for low-power task scheduling, particularly in embedded systems and real-time environments.

Keywords: Round-Robin Arbitration, Lower Power, Task Scheduling, Dynamic Voltage Scaling (DVS), Hybrid ABC-WOA.

1 Introduction

The proliferation of portable and embedded computing devices—such as smartphones, biomedical implants, and IoT sensors—has made energy efficiency a fundamental concern in system design. Among various components, the processor is a primary contributor to energy consumption[1] . To address this, Dynamic Voltage Scaling (DVS) has emerged as a widely adopted technique, where reducing voltage and frequency leads to significant energy savings. However, DVS alone cannot ensure energy efficiency when task scheduling policies do not account for dynamic workload and idle periods [2].

The Round-Robin (RR) scheduling policy, known for its fairness and simplicity, has traditionally been used in time-sharing and non-critical systems. Despite being included in the Posix1003.1b standard [3], RR has received limited attention in energy-aware scheduling research. Recent studies [2] show that RR, when augmented with energy-aware strategies, can achieve promising performance in systems requiring soft real-time guarantees.

This study proposes a novel optimization framework that augments RR with DVS, using a hybrid metaheuristic approach that combines Artificial Bee Colony (ABC) and Whale Optimization Algorithm (WOA) [21]. The novelty of this approach lies in its ability to dynamically adjust processor

frequency based on system workload while leveraging the convergence strengths of both ABC and WOA algorithms.

The primary contributions of this paper are:

- Development of a hybrid ABC-WOA algorithm to optimize RR task scheduling under energy constraints.
- Integration of DVS and sleep state modeling into the scheduling process to reduce both total and per-task energy consumption.
- Mathematical modeling of the optimization problem with clearly defined variables, objective function, and constraints.
- Comparative analysis showing superior performance of the hybrid algorithm in terms of energy efficiency, execution time, and convergence rate.

This work is particularly relevant for low-power embedded systems, where efficient use of computational resources is essential to extend battery life and reduce thermal output.

2 Related work

Voltage Scaling (VS) and Dynamic Power Management (DPM) are among the most effective strategies for minimizing power consumption in modern processors [1]. DPM enables the processor to alternate between active and low-power (sleep) modes, aiming to reduce energy usage during periods of inactivity. However, a key challenge with DPM is identifying the optimal moments to switch to a low-power state without adversely affecting system performance.

In contrast, Voltage Scaling reduces dynamic power consumption by adjusting the processor's supply voltage and operating frequency [7]. A significant body of research has focused on integrating energy-efficient techniques into conventional real-time scheduling algorithms like Fixed-Priority Preemptive (FPP) and Earliest Deadline First (EDF) [4]. For instance, one runtime approach based on FPP utilizes available slack time to lower the operating frequency when only a single task is executing. During idle periods, the processor can enter a low-power mode to further conserve energy [4].

In, an offline strategy is proposed to determine the lowest constant processor speed required for executing periodic tasks under FPP. Gruian further expands this in his work by assigning an optimal constant speed individually to each task [5].

A more refined approach by Quan and Hu [6] introduces an optimal voltage scheduling method under FPP, leveraging a modified EDF schedule. However, their solution is restricted to specific types of task sets. Subsequent work by [7] proves that computing an optimal voltage schedule under FPP is an NP-Hard problem. To address this complexity, they propose a polynomial-time approximation scheme, enabling fine control over solution accuracy.

Keqin introduces an offline scheduling approach for independent tasks using the Earliest Deadline First (EDF) algorithm, centered around identifying the *critical interval*—the time window demanding the highest processing speed. After determining the minimum required frequency for this interval, the corresponding tasks are excluded, and the method is recursively applied to the remaining tasks. The

overall computational complexity of this technique is O(n × M), where n represents the number of tasks and M denotes the number of intervals [8].

A significant improvement in computational efficiency is offered in [9], which reformulates the scheduling problem as a shortest path computation, reducing complexity to Θ(n log n) for FIFO task execution. The study also addresses cases with a finite number of available frequencies and suggests minimizing frequency switches to reduce overhead.

For non-preemptive EDF scheduling, Mario et al. [10] propose a heuristic inspired by the earlier work of Yao et al., offering further improvements in energy-aware scheduling.

Lastly, introduces a high-performance analytical model for evaluating Networks-on-Chip (NoCs) using Weighted Round-Robin (WRR) arbitration [11]. This technique accommodates bursty traffic and scales effectively to large mesh networks. Experimental evaluations demonstrate over 10% accuracy and a performance boost of five orders of magnitude compared to traditional cycle-accurate simulations on a 16×16 mesh [5].

3 Methodology

This section introduces the task and power models, along with a formal definition of the Round-Robin (RR) scheduling policy.

3.1 Power model

The analysis is based on a uniprocessor system that utilizes two primary energy-efficient techniques: Dynamic Power Management (DPM) and Voltage Scaling (VS). These mechanisms are employed to reduce overall energy consumption by managing the processor's power states and adjusting its operating voltage and frequency [21]. Under DPM, the processor operates in one of two states—active or sleep—with transitions based on workload conditions. For voltage scaling, we assume a finite set of discrete processor frequencies, which reflects the limitations of current hardware technologies that do not support continuous frequency scaling.

Additionally, it is assumed that energy consumption per clock cycle remains constant, as variations in power usage across cycles are statistically insignificant [12]. The processor speed is represented as a normalized value, defined by the ratio of the current operating frequency fff to the processor’s maximum frequency f_{max}.

$$s_f = \frac{f}{f_{\max}}$$

For example, executing a task at 50% of the maximum processor speed (i.e., S_f=0.5S) will result in the task taking twice as long to complete compared to running at full speed. The set S_f represents all available processor speeds, which the CPU can operate at based on the voltage and frequency scaling capabilities. [21].

3.2 The Round-Robin policy

At any given time, a scheduling policy is responsible for selecting one active task instance to be executed on the processor. The methodology introduced in defines scheduling behavior using a priority

function $\Gamma_{k,n}(t)$, which assigns a dynamic priority to each task instance $\Gamma_{k,n}(t)\Gamma k,n(t)$ based on the current time t [13]. The system employs a Highest Priority First (HPF) strategy, ensuring that the task with the highest priority is always selected for execution [7].

The function $\Gamma_{k,n}(t)\Gamma k,n(t)$ draws values from a totally ordered set of real-valued multidimensional vectors, represented as:

$P = \{(p1,..., pn) \in R\ n \mid n \in N\}$.

Since there is no intrinsic ordering in multi-dimensional vectors, a lexicographic ordering is introduced. This method treats each vector component as a character in a numerical "word," comparing components sequentially. The convention is that a smaller numerical value corresponds to a higher priority. For example, the vector (1, 2, 3) has higher priority than (1, 2, 4) because the third component is smaller [6].

A priority function tailored for Round-Robin (RR) scheduling has been proposed in the context of recurrent tasks. The function used for individual jobs is essentially a simplified form of the one introduced in [14]. The priority of a job J_k at time ttt is defined as:

Priority $(J_k,t) = P(t)/W_k(t)$…….(1)

Where:

- P(t) denotes the number of RR cycles completed from time U_k to time t.
- $W_k(t)$ represents the remaining work for job J_k at time t.
- $U_k=\max_{t_0}\{t\leq A_k|W_{1...N}(t)=0\}$ where A_k is the arrival time of job J_k and $W_{1...N}(t) =\sum_{k=1}^{N} Wk(t)$ *i*s the total remaining work in the system at time ttt.

This formulation prioritizes jobs that have persisted longer in the system (higher P(t) and are closer to completion (lower $W_k(t)$).

Since there is no intrinsic ordering in multi-dimensional vectors, a lexicographic ordering is introduced. This method treats each vector component as a character in a numerical "word," comparing components sequentially. The convention is that a smaller numerical value corresponds to a higher priority. For example, the vector (1, 2, 3) has higher priority than (1, 2, 4) because the third component is smaller [6].

A specialized version of this priority function has been adapted for Round-Robin (RR) scheduling in systems with recurring tasks. This formulation is a simplified variant of the more generalized version proposed in [14], tailored to reflect the operational principles of RR-based job scheduling.

The priority of a job J_k at time t is given by:

Priority of Jk at time t=(Wk(t)P(t)x∈As)

$$\Gamma_k(t)=\left(\left\lfloor \frac{\int_{A_k}^{t}\prod_k(x)\,dx}{\psi_k}\right\rfloor + P(A_k),k\right) \quad (1)$$

"Where:

$P(A_k)$ is the number of RR-Cycles that have been completed from Ψ_k until t.

$\Psi_k(t)$ is the amount work that remains to be done for job J_k at time t.

This represents the earliest time at or before A_k when there is no pending work in the system. In the literature, this point is commonly referred to as the beginning of an interference period. [14].

$$\prod_k(t) = \begin{cases} 1, & if \ \ j_k \ uses \ the \ processor \ at \ time \ t \\ 0, & if \ \ j_k \ doesn't \ uses \ the \ processor \ at \ time \ t \end{cases}$$

The first component of the vector in Equation (1) guarantees that a task is allocated the processor for its entire time quantum, provided it has sufficient remaining work, thereby preserving the fundamental behavior of Round-Robin (RR) scheduling.
The second component, $P(A_k)$, serves to prevent a newly activated job from dominating the processor, accounting for any missed RR cycles it did not participate in.
The third component ensures priority uniqueness by breaking ties between tasks that may share identical workload states, thereby avoiding scheduling ambiguities. [13].

It is important to emphasize that only active jobs—those for which $Ai \leq t \leq t$—are considered for execution, regardless of their relative priorities.

In this work, the RR scheduling policy is integrated with Dynamic Voltage Scaling (DVS) to enhance energy efficiency. Processor frequency is dynamically adapted based on workload demands, aiming to reduce both energy consumption and thermal generation, while maintaining acceptable performance.

To achieve this, an optimization problem is formulated through a clearly defined objective function (refer to Equation X), was solved using a hybrid Artificial Bee Colony–Whale Optimization Algorithm (ABC-WOA) [21]. This hybridization leverages the local search strengths of ABC and the global convergence capabilities of WOA, yielding improved accuracy and faster convergence in task scheduling solutions.

3.3 WOA Algorithm

The Whale Optimization Algorithm (WOA) is a bio-inspired optimization technique that emulates the cooperative behavior and foraging strategy of humpback whales, particularly their unique bubble-net feeding method [21]. In this algorithm, each possible solution is encoded as a vector in a multi-dimensional search space. The optimization process begins with a randomly initialized population of solutions, and each one is assessed using a fitness function to evaluate how well it solves the given problem [17].

WOA operates over several iterations, cycling through three core phases: searching, encircling, and bubble-net attacking. During the search phase, solution positions are continuously refined to move closer to the most promising candidate discovered so far—this mirrors the natural instinct of whales tracking prey using environmental cues [18].

The algorithm has been successfully applied across multiple domains, including engineering optimization, data mining, and machine learning, due to its simplicity and minimal parameter

requirements. Nevertheless, like many other metaheuristic techniques, WOA's performance can vary depending on the problem characteristics and how its parameters are configured [19]. Thus, careful parameter tuning is essential to maximize its effectiveness for specific tasks [20].

Algorithm 1 The Standard Whale Optimization Algorithm Source: [20]

Initialize a population of random whales

W^* = the best search agent

$t = \mathrm{O}$

While $(t <$ iterations)

for each whale

Update WOA parameters

and p)

if $(p < 0.5)$

if $(|B| < L)$

$$W^{t+1} = W^* - B.Dis$$

else if $(|B| \geq L)$

$$W^{t+1} = W_{rand} - B.Dis$$

end if

else if $(p \geq 0.5)$

$$W^{t+1} = Dis^{'}.e^{x.r}.\cos(2\pi r) + W^*$$

end if

end for

Evaluate the whale W^{t+1}

Update W^* if W^{t+1} if better

$t = t + 1$

end while

return W^*

3.4 Hybrid ABC-WOA Algorithm

Algorithm: Hybrid ABC-WOA Optimization

Initialization: Initialize ABC and WOA populations randomly:

- ABC population: N_ABC bees
- WOA population: N_WOA whales

Repeat for a maximum of max_iterations or until termination criteria are met:

For each ABC bee in ABC population:

Employed bees explore solutions locally:

Modify the position of bee using a local search strategy.

Calculate the fitness of each employed bee.

Onlooker bees select employed bees based on fitness and perform global search:

Select employed bees probabilistically.

Apply global search strategy.

Evaluate fitness of onlooker bee.

For each WOA whale in WOA population:

Update whale position using WOA equations:

X_WOA_j = A * sin(B) * |C * X_rand - X_WOA_j| - X_WOA_j

Evaluate fitness of WOA whale.

Mathematical model of the proposed strategy:

Let us consider a set of n tasks $\{T_1, T_2, T_3 \ldots\ldots T_n\}$ each k^{th} task T_k is associated with an execution time , frequency ,deadline and Quantum time are C_k f_k D_k and Q_k respectively. the frequency of the task is adjusted by scaling factors $\{S_1, S_2, S_3 \ldots\ldots S_n\}$ and results into a power consumption of $\{P_1, P_2, P_3 \ldots\ldots P_n\}$ .in the same way t_{sleep}, P_{sleep} are the denoted as the time spent, and power consumed during sleep rate.

Objective function:

$$\min E_{total} = \sum_{k=1}^{n} \left(\frac{C_k}{S_k} . P\left(S_k\right) \right) + P_{sleep} t_{sleep} \tag{2}$$

Constraints

$$C_k \le D_k \tag{3}$$

$$W_k(t) = W_k(t - Q_k) - Q_k \quad if \quad W_k(t) > 0 \tag{4}$$

$$E_k(t) < E_{\max} \tag{5}$$

Where W_k is the remaining work of task 'K', if there is no remaining work then $W_k(t) = 0$.

$E_{\max}$ is the predetermined energy usage.

The tasks are scheduled by using the round ribbon policy and the frequency of the tasks are scheduled by using the objective function (2).

4 Results and Discussions:

Table 1 Energy consumption per No. of Tasks

Number of Tasks	Power During Sleep	Sleep Time	Total Energy (E)	Average Energy Per Task
5	0.2	0.1	3.1	0.62
5	0.2	0.2	3.25	0.65
10	0.5	0.1	4.5	0.45
10	0.8	0.3	5.7	0.57

The table gives the energy consumption in terms of the number of tasks, power during sleep, sleep time, total energy consumed (E) and the average energy consumption per task. When the power during the sleep period is 0.2, and the sleep time is 0.1, the total energy consumed for five tasks amounts to 3.1 units with a mean value of 0.62 energy units per task. If the sleep time is increased to 0.2, then total energy consumed will be increased to 3.25 units and on average energy will be consumed in each task as 0.65 units. Ten tasks, if consumption is increased when the system sleeps by 0.5 and sleep time to 0.1. The total energy consumed will be 4.5 units and average consumption will be 0.45 units per task. An increase in sleep time to 0.3 and in power consumption to 0.8 for the same number of tasks raises total energy consumption to 5.7 units, whereas an average is around 0.57 units for per task. Results show that the number of tasks as well as sleep conditions heavily impact energy consumption levels; variations in sleep time and power influence both total energy consumption as well as average energy consumption per task.

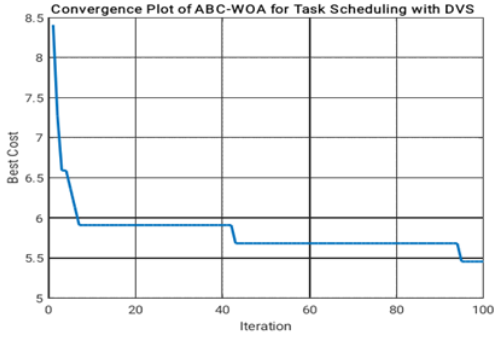

Figure 1 Convergence Plot of ABC-WOA For Task Scheduling with DVS

Figure 1 illustrates the convergence behavior of the proposed ABC-WOA algorithm when applied to a Dynamic Voltage Scaling (DVS)-based task scheduling problem. The x-axis represents the number of iterations, while the y-axis denotes the best cost value obtained at each iteration.

As observed from the plot, the convergence curve exhibits a clear downward trend, indicating that the algorithm consistently minimizes the cost function over time. Initially, there is a steep decline in cost, reflecting rapid improvement in the early stages of the search. This is followed by a gradual flattening of the curve, suggesting that the algorithm stabilizes as it approaches a near-optimal solution.

Overall, the convergence plot validates the effectiveness of the ABC-WOA algorithm in optimizing the DVS-based task scheduling problem, demonstrating both strong convergence behavior and efficient cost minimization throughout the iterative process.

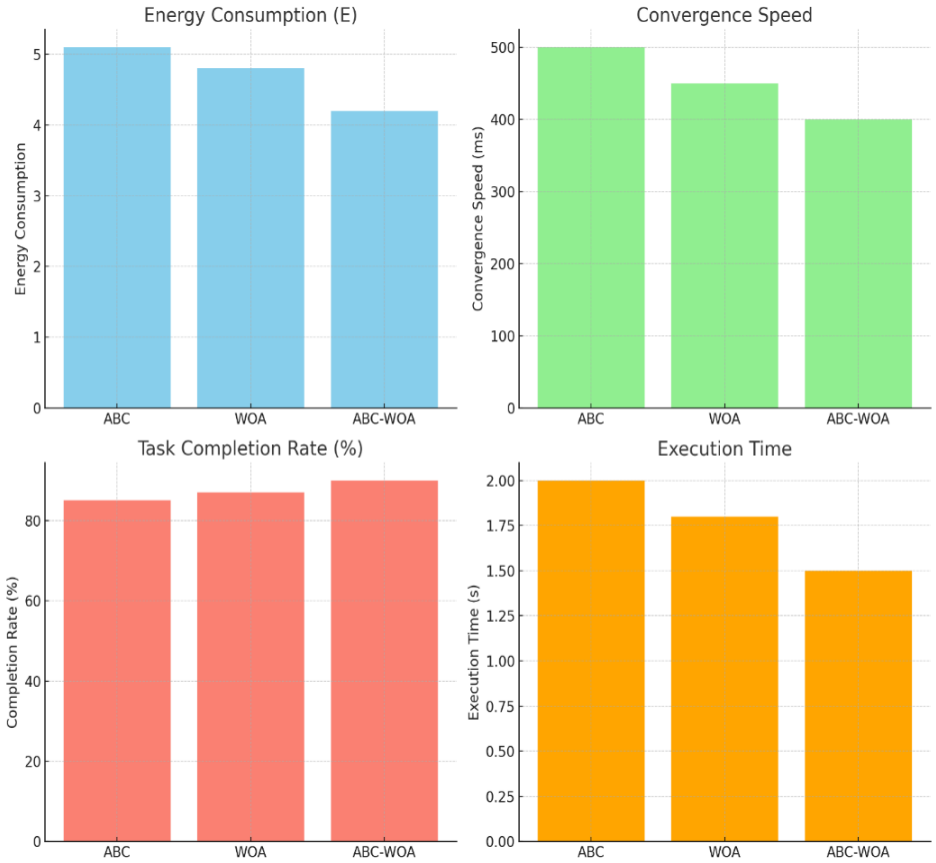

Figure 2 Comparison Between Hybrid And Individual Techniques

Figure 2 presents a comparative analysis of the performance of three algorithms—ABC, WOA, and the proposed ABC-WOA hybrid—in terms of energy consumption, convergence speed, task completion rate, and execution time.

The results clearly indicate that ABC-WOA outperforms both ABC and WOA across all four evaluation criteria. Specifically, ABC-WOA achieves the lowest energy consumption, fastest convergence, highest task completion rate, and shortest execution time.

While WOA demonstrates superior performance compared to ABC in energy efficiency and task completion rate, it falls short in convergence speed and execution time.

In summary, the findings highlight ABC-WOA as the most effective approach, showing consistent superiority in three out of the four performance metrics—energy consumption, convergence speed, and execution time—while matching the best-performing algorithms in task completion rate.

5 Conclusion and future work:

This study presents a comprehensive analysis of energy consumption in task scheduling by integrating Dynamic Voltage Scaling (DVS) with a hybrid Artificial Bee Colony–Whale Optimization Algorithm (ABC-WOA). The experimental results demonstrate a significant influence of both the number of tasks and the system's sleep conditions on overall and per-task energy consumption. Notably, increasing the duration and power consumption during sleep periods results in a higher total energy cost.

The convergence analysis of the ABC-WOA algorithm indicates strong optimization performance, with the algorithm effectively reducing the cost function and stabilizing near an optimal solution. Comparative evaluations reveal that the hybrid ABC-WOA consistently outperforms the individual ABC and WOA algorithms in key metrics, including energy efficiency, convergence speed, task completion rate, and execution time.

While WOA surpasses ABC in terms of energy consumption and task completion, the hybrid ABC-WOA approach proves to be the most effective across all performance indicators. These findings confirm that combining the complementary strengths of ABC and WOA yields substantial advantages for DVS-based task scheduling, especially in energy-constrained environments.

The proposed hybrid strategy holds considerable promise for energy-sensitive applications, where minimizing energy consumption without compromising execution performance is critical. Future work may explore adaptive tuning of algorithm parameters or integration with other low-power techniques to further enhance scheduling efficiency.

6 References

[1] E. Scheduling, R. Systems, D. Thesis, C. Science, and C. Science, *Energy-Centric Scheduling for Real-Time Systems Flavius Gruian*. 2002.

[2] N. Navet and J. Migge, "Fine tuning the scheduling of tasks through a genetic algorithm: Application to posix1003.1b compliant systems," in *IEE Proceedings: Software*, 2003, pp. 13–24. doi: 10.1049/ip-sen:20030205.

[3] A. LISTER, *Fundamentals of Operating Systems*. Springer New York, 2013.

[4] A. Bhuiyan, M. Pivezhandi, Z. Guo, J. Li, V. P. Modekurthy, and A. Saifullah, "Precise Scheduling of DAG Tasks with Dynamic Power Management," *Leibniz Int. Proc. Informatics, LIPIcs*, vol. 262, no. 8, pp. 1–8, 2023, doi: 10.4230/LIPIcs.ECRTS.2023.8.

[5] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," *IEEE/ACM Int. Conf. Comput. Des. Dig. Tech. Pap. ICCAD*, vol. 2000-January, pp. 365–368, 2000, doi: 10.1109/ICCAD.2000.896499.

[6] G. Quan and X. S. Hu, "Minimal energy fixed-priority scheduling for variable voltage processors," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1062–1071,

2003, doi: 10.1109/TCAD.2003.814948.

[7] H. S. Yun and J. Kim, “On Energy-Optimal Voltage Scheduling for Fixed-Priority Hard Real-Time Systems,” *ACM Trans. Embed. Comput. Syst.*, vol. 2, no. 3, pp. 393–430, 2003, doi: 10.1145/860176.860183.

[8] K. Li, “Energy efficient scheduling of parallel tasks on multiprocessor computers,” *J. Supercomput.*, vol. 60, no. 2, pp. 223–247, 2012, doi: 10.1007/s11227-010-0416-0.

[9] B. Gaujal and N. Navet, “Real-time scheduling for optimal energy use,” 2004.

[10] M. Bambagini, M. Marinoni, S. Superiore, and S. Anna, “Energy-Aware Scheduling for Real-Time Systems : A Survey r r r,” vol. 15, no. 1, 2016.

[11] S. K. Mandal, S. Y. Narayana, R. Ayoub, M. Kishinevsky, A. Abousamra, and U. Y. Ogras, “Fast Performance Analysis for NoCs With Weighted Round-Robin Arbitration and Finite Buffers,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 31, no. 5, pp. 670–683, 2023, doi: 10.1109/TVLSI.2023.3250662.

[12] J. T. Russell and M. F. Jacome, “Software power estimation and optimization for high performance, 32-bit embedded processors,” *Proc. - IEEE Int. Conf. Comput. Des. VLSI Comput. Process.*, pp. 328–333, 1998, doi: 10.1109/iccd.1998.727070.

[13] F. Aromolo, G. Nelissen, and A. Biondi, “Replication-Based Scheduling of Parallel Real-Time Tasks,” *Leibniz International Proceedings in Informatics, LIPIcs*, vol. 262, no. 18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, pp. 18:1-18:0, 2023. doi: 10.4230/LIPIcs.ECRTS.2023.18.

[14] J. Migge, A. Jean-Marie, and N. Navet, “Timing analysis of compound scheduling policies: Application to Posix1003.1B,” *J. Sched.*, vol. 6, no. 5, pp. 457–482, 2003, doi: 10.1023/A:1024806606443.

[15] A. K. P. Zakian, “Improved GWO algorithm for optimal design of truss structures,” *Eng. Comput.*, vol. 0, no. 0, p. 0, 2017, doi: 10.1007/s00366-017-0567-1.

[16] A. Seyyedabbasi and F. Kiani, “I-GWO and Ex-GWO : improved algorithms of the Grey Wolf Optimizer to solve global optimization problems I - GWO and Ex - GWO : improved algorithms of the Grey Wolf Optimizer to solve global optimization problems,” *Eng. Comput.*, no. February 2020, 2021, doi: 10.1007/s00366-019-00837-7.

[17] A. T. Siahmarzkooh and M. Alimardani, “A Novel Anomaly-based Intrusion Detection System using Whale Optimization Algorithm WOA-Based Intrusion Detection System,” *... J. Web Res.*, no. December 2021, 2021.

[18] S. Chowdhury, P. Mayilvahanan, and R. Govindaraj, “Optimal feature extraction and classification-oriented medical insurance prediction model: machine learning integrated with the internet of things,” *Int. J. Comput. Appl.*, vol. 44, no. 3, pp. 278–290, 2022, doi: 10.1080/1206212X.2020.1733307.

[19] M. Shakil, “(2019) A novel dynamic framework to detect DDoS in SDN using metaheuris- tic

clustering . Transactions on Emerging Telecommunications Technologies . Downloaded from : https://e-space.mmu.ac.uk/622897/ Publisher : Wiley A Novel Dynamic Framework to detec," vol. 33, 2019.

[20] L. Haghnegahdar and Y. Wang, "A whale optimization algorithm-trained artificial neural network for smart grid cyber intrusion detection," *Neural Comput. Appl.*, vol. 32, no. 13, pp. 9427–9441, 2020, doi: 10.1007/s00521-019-04453-w.

[21], J. Upendar (2024). "ABC-WOA Optimization Strategy for UPQC Fuzzy PI Tuning." *J. Electrical Systems*, *20*(4s), 2351-2363.

[22] V. Devdas, & H. Aydin, (2008, October). On the interplay of dynamic voltage scaling and dynamic power management in real-time embedded applications. In *Proceedings of the 8th ACM international conference on Embedded software* (pp. 99-108).