International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 5, Number 3, Year 2025
DOI: https://doi.org/10.62051/ijcsit.v5n3.02
Journal homepage: https://wepub.org/index.php/IJCSIT/index

Analysis of Virtual Machine Resource Scheduling Strategies in Cloud Computing Environments

Yufei Liu

School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2033, Australia
z5502598@ad.unsw.edu.au

ABSTRACT

With the rapid development of cloud computing technology, effective virtual machine resource scheduling has become a key factor in improving the operational efficiency of data centers. This paper comprehensively evaluates existing resource scheduling algorithms, including traditional polling and priority queuing methods, as well as meta-heuristic based ant colony algorithms, genetic algorithms, and artificial bee colony algorithms, and focuses on analyzing the applicability of these algorithms in cloud environments and the challenges they face. Through case studies, this paper reveals how algorithmic optimization can be used to achieve efficient resource allocation in highly dynamic cloud environments. It is found that although the existing algorithms can optimize resource allocation under specific conditions, they are still deficient in dynamic adaptation, resource conflict handling, computational overhead and multi-objective optimization.

KEYWORDS

Cloud computing; Resource scheduling; Ant colony algorithm; Genetic algorithm; Artificial bee colony algorithm

1. INTRODUCTION

Cloud computing is an Internet-dependent service model that enables users to remotely utilize and manage computing resources such as servers and applications without having to directly own or maintain these hardware devices. Although cloud computing provides great flexibility and scalability, it also faces the problem of over- and under-allocation in computing resource allocation, which affects cost efficiency and system performance. Since the scheduling of computing resources in the cloud environment is a multi-user, multi-task and concurrent execution process, efficient scheduling of virtual machine resources is the key to optimize the performance of cloud computing. And with the wide application of virtualization technology, effective resource scheduling has become one of the key challenges for cloud service providers. Resource scheduling refers to the process of reasonably allocating and managing computing resources in a cloud computing environment to ensure that virtual machines can run efficiently. A good resource scheduling strategy is not only related to maximizing resource utilization, but also directly affects service quality and user satisfaction. However, existing resource scheduling strategies, such as polling scheduling, priority queuing, and shortest job first. Although they perform well in specific scenarios, they still face problems such as resource wastage and unfair scheduling in cloud computing environments that deal with high concurrency and dynamic changes.

These challenges have inspired the need for research on more efficient virtual machine resource scheduling strategies. To address these challenges, researchers have begun to explore more

sophisticated optimization algorithms, such as ant colony algorithms and genetic algorithms, which mimic behavioral and evolutionary processes in nature to find optimal computational resource allocation solutions. These methods can significantly improve the efficiency and fairness of scheduling in some cases, but at the same time increase the computational complexity and execution overhead. This topic explores the related development trends by comprehensively analyzing these computational resource scheduling strategies and their performance in real-world applications, and comparing them with traditional scheduling strategies.

2. VIRTUAL MACHINE SCHEDULING IN CLOUD COMPUTING

2.1. Fundamentals of Vm Scheduling

2.1.1. Virtualization technology and resource isolation

Virtual machine resource scheduling is first based on virtualization technology, which allows multiple virtual machines to run in parallel on a single physical server, each of which emulates a complete hardware system and thus runs a separate operating system and applications. The core of virtualization is resource isolation, which ensures that each virtual machine is independent of each other in terms of resource usage and avoids interfering with each other.

2.1.2. Dynamic resource allocation

A key principle of VM resource scheduling is dynamic resource allocation. Resource allocations (such as CPU time slices, memory space, storage, and network bandwidth) are dynamically adjusted to fit the needs of different VMs based on the current workload and system performance metrics. This approach allows the system to react quickly in the face of load changes, optimizing resource utilization and minimizing performance bottlenecks.

2.1.3. Load balancing

Resource scheduling policies must take into account load balancing, Specifically, how the workload is evenly distributed among multiple virtual machines or physical servers in the cloud. With an effective load balancing strategy, certain servers can be prevented from being overloaded while others are idle, thus improving overall system efficiency and reliability.

2.1.4. Priority and policy-driven scheduling

In VM resource scheduling, resource allocation is usually required based on predefined policies and VM priorities. For example, for critical business applications, the system may provide higher resource guarantees to ensure service response time and availability. For non-critical applications, on the other hand, resource scheduling can be more flexible to reduce costs.

2.1.5. Automation and flexibilization

High-level principles of virtual machine resource scheduling include automated processing and elastic tuning. Automated processing allows the system to allocate and adjust resources without human intervention. Elastic tuning, on the other hand, refers to the ability of the system to automatically scale up or down resources based on actual application requirements, which is a major advantage offered by cloud computing.

2.2. Key Metrics for Vm Scheduling

During the evaluation and optimization of VM resource scheduling, the explicit setting of key metrics is critical to ensure system performance and efficiency. Below are a few core scheduling metrics that collectively determine the effectiveness of a cloud computing resource scheduling strategy.

2.2.1. Resource utilization

Resource utilization measures how efficiently virtual machines and underlying physical hardware resources (such as CPU, memory, storage, etc.) are used. Higher resource utilization indicates that resources are fully utilized, resulting in less wasted resources and lower operating costs. However, too high utilization may lead to resource competition, which may affect system stability and performance. Therefore, optimizing resource utilization requires finding a balance between ensuring system performance and avoiding resource overload.

2.2.2. Time of task completion

Task completion time is the total time from the submission of a task to the completion of processing by the system. This index is a direct expression of evaluating the processing efficiency of the system, reflecting the speed of the system for task processing. In a multitasking environment, the key to optimizing task completion time is to reasonably schedule resources to shorten processing time and ensure that important tasks are given priority in obtaining the necessary resources, so as to achieve the optimization of overall efficiency.

2.2.3. System throughput

System throughput refers to the number of tasks that can be successfully completed by the system per unit of time. This metric evaluates the processing capability of the system, and a high throughput indicates that the system is able to handle a large number of tasks efficiently. Improving system throughput relies on an efficient resource scheduling strategy and an optimized task processing flow to ensure that each task receives the required computing resources in a timely manner.

2.2.4. Response time

Response time measures the latency from the time a user makes a request to the time the system starts processing the request. In cloud computing services, especially in real-time data processing and interaction-intensive applications, low response time is key to enhancing user experience. Strategies to optimize response time include ensuring immediate availability of resources and employing efficient load distribution mechanisms to reduce waiting time and processing latency of tasks.

2.3. Main goals of virtual machine scheduling

2.3.1. Improve resource utilization

Improving resource utilization is one of the core objectives in virtual machine resource scheduling. Through effective resource scheduling strategies, we can ensure that CPU, memory, storage and network resources on physical servers are maximized, thus reducing resource waste and increasing economic benefits. Achieving this goal requires accurate load prediction, dynamic resource allocation, and efficient resource recovery strategies.

2.3.2. Optimizing performance and response time

Performance optimization and response time reduction have a direct impact on user satisfaction with cloud services. Response time optimization is to reduce the delay between user request and system response, which is especially important in interactive applications and real-time data processing scenarios. Through optimized scheduling algorithms, the system's processing power and response time can be significantly increased, thus improving the overall service quality.

2.3.3. Achieving equity among multiple users

Ensuring fairness in resource allocation among different users is another important goal in multi-tenant cloud computing environments. Resource scheduling should ensure that no user or application is adversely affected by uneven resource allocation. This includes rationalizing the allocation of resource shares, managing the resource requirements and priorities of different users, and rationalizing the resource allocation in high load situations. Strategies to achieve fairness may employ

fair queues, resource quota systems and dynamic adjustment mechanisms to ensure the stable operation of their applications and services.

3. LITERATURE REVIEW

In the domestic research on virtual machine resource scheduling, scholars have proposed a variety of innovative strategies and algorithms to cope with the challenges in cloud computing environments, optimize resource allocation, and improve service efficiency and user satisfaction.

Wei team proposed an improved ant colony algorithm (DSFACO) based on the shortest task delay time, which improves the task scheduling efficiency and user satisfaction by optimizing the pheromone accumulation and usage strategies, and at the same time demonstrates the applicability of the ant colony algorithm in complex scheduling problems [1]. The team of Qiyao Guo improved the pheromone updating mechanism by introducing a feedback factor and combined the global search advantage of the frog hopping algorithm, which significantly improved the algorithm's optimization ability and convergence speed, providing a more powerful and flexible solution for complex resource scheduling [2]. In addition, Wenzhong Xu team proposed a virtual machine load balancing scheduling strategy based on genetic algorithm, which optimizes virtual machine allocation and balances node loads by simulating the biological evolution process, and at the same time, reduces dynamic migration, improves the resource utilization efficiency and reduces the system maintenance cost, which provides an effective method for resource optimization in cloud computing environment [3].

In the international research area, the exploration of virtual machine resource scheduling has shown similarly broad and deep progress.

Rathor and his team proposed two dynamic scheduling strategies for VMs, Best Fit and Worst Fit, to improve scheduling efficiency and responsiveness by maximizing resource utilization and balancing server load, respectively [4]. Kruekaew team combined artificial bee colony algorithm and heuristic scheduling algorithm to propose an artificial bee colony-heuristic task scheduling (HABC) for both homogeneous and heterogeneous [5]. Ragmani developed a hybrid fuzzy ant colony optimization (FACO) algorithm that combines fuzzy logic to deal with uncertainty and the efficient search mechanism of ACO to significantly improve the response time and load balancing efficiency of VM scheduling in a dynamic multi-node environment. These strategies show good applicability and performance advantages in complex cloud computing environments [6].

These research results demonstrate innovations and advances in the field of virtual machine resource scheduling. With these advanced scheduling techniques, researchers are able to cope with the complexity and dynamics in cloud services more effectively and provide better and more efficient services to cloud computing users around the world.

4. ANALYSIS AND EVALUATION OF SCHEDULING STRATEGIES

4.1. Basic Scheduling Strategy

4.1.1. Round robin scheduling

Working Principle: Polling scheduling is a time-slice based scheduling algorithm in which the system allocates a time slice of fixed length for each task in the queue to be processed sequentially. The length of the time slice is usually preset by the operating system or scheduling manager. All the tasks are listed in the queue in order of entry and if the time slice runs out during execution, then the task is placed at the end of the queue and waits for the next turn.

4.1.2. Priority scheduling

Working Principle: Under the priority scheduling policy, each task is assigned a priority identifier, and the scheduler decides the execution order of tasks based on this priority. Higher priority tasks are given priority, and the system processes lower priority tasks only when the higher priority tasks are completed or blocked. This strategy can be implemented as preemptive or non-preemptive. In preemptive priority scheduling, whenever a high-priority task becomes executable, it preempts the lower-priority task that is currently executing. The preempted task will return to the ready queue and wait for the next scheduling. In contrast, non-preemptive priority scheduling allows currently executing tasks to run to completion or enter a blocking state regardless of their priority. Even if higher-priority tasks become executable in the meantime, the system will not interrupt the current task, but will only consider these high-priority tasks in the next scheduling.

4.1.3. Shortest job first

Working Principle: Shortest Job Prioritization Policy (SJF) is an approach to task scheduling based on the estimated task execution time. Among all the tasks waiting to be executed, the scheduling system will continuously evaluate the estimated completion time of each task and prioritize the scheduling of those tasks with the shortest execution time. The core assumption of this strategy is that the execution time of a task can be accurately predicted or estimated, allowing the scheduler to make optimal scheduling decisions to minimize the total waiting time of the system.

4.2. Improved Scheduling Strategies

4.2.1. Ant colony algorithm

(1) Overview

Ant Colony Optimization (ACO) is a heuristic algorithm based on the paths of ants in nature to find food, with parallel search capability and associated feedback mechanism. In cloud computing environments, this algorithm has been successfully applied to virtual machine resource scheduling to optimize the allocation and utilization of resources to enhance the efficiency of services and reduce energy consumption [7].

(2) Basic principle

The ant colony algorithm optimizes path selection by simulating the mechanism of ants releasing and sensing pheromone, where the concentration of pheromone reflects the superiority of the path. The ants will release pheromone proportional to the quality of the path ($\tau s = \tau c + \tau g$, τc is a constant and τg is the value obtained by the genetic algorithm) when passing through the path, and at the same time, the pheromone will evaporate gradually to maintain the exploration ability while avoiding converging to the local optimal solution too early [1]. In path selection, ants make probabilistic decisions based on pheromone concentration and heuristic information; paths with higher concentration are more likely to be selected, while stochastic exploration enhances the diversity and robustness of the solution space. Heuristic information such as current server load, energy consumption and resource usage efficiency provide additional guidance for path selection, helping to optimize VM deployment for load balancing, reducing energy costs and improving resource utilization [8]. This mechanism enables the algorithm to strike a balance between global optimization and local exploration for complex cloud scheduling problems.

(3) Advantages

1) Adaptability: Ant colony algorithms have excellent adaptive capabilities and can dynamically adjust their search strategies in response to environmental changes. This is due to the pheromone updating mechanism that continuously accumulates past search experience, enabling the algorithm to learn and optimize its behavior in changing environments.

2) Strong optimization capabilities: The ACO algorithm efficiently concentrates the search effort on well-performing solutions through a positive feedback mechanism, which allows the algorithm to excel in this aspect of finding globally optimal solutions. The increase in pheromone concentration directs more ants to explore these promising paths, leading to rapid convergence to the optimal or near-optimal solution [9].

3) Distributed computing capabilities: Due to the natural parallel structure of the ACO algorithm, where each ant searches the solution space independently, it is well suited to be implemented as a distributed algorithm. This feature makes ACO particularly suitable for deployment in multiprocessor systems or cloud computing environments, where it can take full advantage of the computational resources provided by modern hardware architectures to accelerate the process.

4) Robustness: The robustness of the ACO algorithm is reflected in its high tolerance to initial conditions and random perturbations. The operation of the algorithm does not depend on a specific initial solution, and the stochastic exploration component of its iterative process enables it to effectively avoid falling into local optima [10]. In the context of resource scheduling, the ACO is able to generate effective scheduling strategies even when the input data is incomplete or some system information is unknown.

4.2.2. Genetic algorithms

(1) Overview

Genetic Algorithm (GA) is a heuristic optimization algorithm based on the theory of biological evolution, which aims to find optimal solutions to complex problems by simulating natural selection and genetic variation. In cloud computing virtual machine resource scheduling, genetic algorithms can be effectively used to optimize the allocation of virtual machines to physical servers to improve resource utilization, reduce energy consumption, balance load, or meet other business requirements [11].

(2) Basic principle

The core idea of genetic algorithms is to start from a collection of solutions (which is called population) and generate new solutions through operations such as selection, crossover and mutation, gradually approximating the global optimal solution.

Genetic algorithms represent the solution space by encoding potential solutions to a problem as chromosomes (binary strings, integers, or real numbers) and begin the search from a randomly generated initial population [12]. Each chromosome is evaluated for the quality of the solution by a fitness function; chromosomes with higher fitness are more likely to participate in the next generation of reproduction by selection. The crossover operation generates diverse offspring by exchanging genes in the parent chromosomes, while the mutation operation enhances the algorithm's ability to explore the optimal solution by randomly altering the genes to introduce new variants. The algorithm generates new populations in each iteration and keeps optimizing until a termination condition is met (maximum number of iterations is reached or solution improvement stalls) [13]. This mechanism simulates the process of natural selection and genetic evolution to effectively solve complex optimization problems.

(3) Advantages

1) Global search capability: Genetic algorithm has the global search property and searches multiple solutions in parallel through populations, which avoids the problem that local search methods are prone to fall into local optimal solutions. It explores a wide range of solution spaces through crossover and mutation operations, and can effectively approximate the global optimal solution in complex virtual machine scheduling problems [14].

2) Adaptation to multi-objective optimization: Scheduling of virtual machines in cloud computing usually requires trade-offs between multiple objectives. Genetic algorithms can handle multiple

optimization objectives at the same time by designing multi-objective fitness functions, making them suitable for complex multi-objective scheduling scenarios [15].

3) Flexibility and Scalability: The coding, fitness function design, and crossover and mutation operations of the genetic algorithm can be flexibly adjusted according to different scheduling needs, and it is highly scalable. It is not only applicable to the virtual machine scheduling problem, but also can be widely applied to other similar resource allocation problems.

4) Parallelism and Distributed Computing Advantages: Since the individual population of genetic algorithm can evolve independently, it is particularly suitable for parallel and distributed computing environments. In cloud computing systems, genetic algorithms can utilize distributed computing resources to handle multiple virtual machine scheduling schemes simultaneously, speeding up the optimization process and improving the efficiency of the algorithm [16].

4.2.3. Artificial bee colony algorithm

(1) Overview

Artificial Bee Colony (ABC) is an optimization algorithm based on the foraging behavior of honey bees, which was proposed in 2005. The algorithm simulates the intelligent behavior of bees during the process of searching for food sources, and in the application of virtual machine scheduling in cloud computing, the artificial bee colony algorithm can efficiently optimize the mapping of virtual machines to physical servers in order to achieve a variety of performance optimization goals.

(2) Basic principle

The artificial bee colony algorithm accomplishes solution optimization collaboratively through three types of bee roles: scout bees randomly search the solution space to discover new potential solutions and thus jump out of the local optimum; honey harvesting bees locally search around the current food source to find a better solution through random perturbations; and observation bees select high-quality food sources for further development based on the information returned from the honey harvesting bees. When the algorithm is initialized, a number of food sources are randomly generated as potential solutions and their quality is evaluated by a fitness function [17]. The honey harvesting bees share the food source information through "dancing", and the observation bees choose the action direction accordingly to improve the development efficiency of high-quality solutions [18]. When a food source has not been improved for a long time, the corresponding honey bees turn into scout bees and conduct global search again to maintain the diversity of the population. Through successive iterations, the algorithm gradually optimizes the solution in the "exploration-exploitation" cycle, and finally approaches the global optimal or better solution.

(3) Advantages

1) Global optimum: The artificial bee colony algorithm combines global and local search to both improve the quality of the solution and reduce the risk of falling into a local optimum. Scout bees are responsible for global search, randomly exploring the space of new solutions without the restriction of the current solution and discovering potentially better solutions; honey bees perform local search by fine-tuning the known solutions to optimize the local performance of the solutions.

2) Parallel processing: The algorithm naturally supports parallel processing, making it particularly suitable for scenarios requiring massively parallel computation, such as cloud computing. Each bee can perform food source search and evaluation independently, allowing multiple search processes to occur simultaneously. This feature takes advantage of multi-core processors and distributed systems to share food source information and search results over a network, enabling efficient resource allocation and significantly improving algorithm execution efficiency [19].

3) Flexibility and Adaptability: The main parameters of the algorithm include the number of bees and the mode of information exchange, and the user can easily adjust these parameters to optimize the performance according to the specific problem. And the algorithm is applicable to a wide range of

problems. Due to the generality of its fundamentals, the algorithm is not only applicable to virtual machine scheduling, but also can be widely used in other resource allocation, path optimization or scheduling problems.

4) Robustness: The robustness of the artificial bee colony algorithm mainly comes from its randomization and multi-strategy search process, which makes it less dependent on the initial solution. Even if the quality of some solutions is not high, the algorithm can still find better solutions through the exploration of other bees, ensuring the diversity of solutions and the robustness of the algorithm. The artificial bee colony algorithm can also automatically adjust its search strategy in response to dynamic changes in the environment, such as sudden changes in virtual machine load or network state [20].

5. RESEARCH CONCLUSION

In this paper, the core problem of virtual machine resource scheduling in cloud computing environments is studied in depth, and a variety of scheduling strategies from traditional to modern ones are comprehensively evaluated. Traditional methods such as polling scheduling, priority scheduling and shortest job first have certain stability and simplicity in specific scenarios, but their limitations gradually emerge when facing the highly concurrent, multi-task dynamically changing cloud computing environments, including low resource utilization, poor scheduling fairness, and insufficient ability to adapt to dynamic loads.

To address these shortcomings, this paper focuses on analyzing scheduling strategies based on meta-heuristic algorithms, including ant colony algorithms, genetic algorithms and artificial bee colony algorithms. These algorithms provide more flexible and efficient solutions for resource scheduling by simulating intelligent behaviors in nature. The ant colony algorithm achieves a balance between global and local search through the pheromone mechanism, can quickly find the near-optimal solution, and maintains a good adaptive ability in dynamic environments; the genetic algorithm effectively optimizes the solution of multi-objective scheduling problems through the selection, crossover, and mutation operations, and performs well in improving the utilization of resources and load balancing; and the artificial bee colony algorithm makes use of the scouting bees, nectar-gathering bees, and observation bees' collaboration mechanism, through the combination of global search and local exploitation, it significantly improves the task completion efficiency, while reducing the system response time and energy consumption.

The research results show that the scheduling strategies based on meta-heuristic algorithms have significant advantages in terms of resource utilization optimization, system throughput enhancement and response time reduction. However, this paper also finds that these algorithms still have the problems of high computational complexity, sensitive parameter settings, and adaptability to dynamic multi-objective environments that need to be enhanced in practical applications. Therefore, future research can focus on exploring the following directions: first, combining deep learning and reinforcement learning techniques to improve the autonomous optimization ability of the algorithms; second, designing more efficient dynamic adaptation mechanisms so that the algorithms can better cope with complex cloud computing environments; and third, developing optimization models with low computational overheads to achieve efficient deployment of the algorithms in large-scale cloud platforms.

In summary, the research in this paper provides a comprehensive theoretical analysis and methodological support for resource scheduling problems in cloud computing environments, and also explores specific directions for the practical deployment and improvement of related technologies.

REFERENCES

[1] Wei, Y., & Chen, Y. (2015). A cloud computing task scheduling model based on an improved ant colony algorithm. Computer Engineering, 41(2), 12-16.

[2] Guo, Q., & Zhu, F. (2017). Cloud computing resource scheduling algorithm based on ant colony algorithm and frog leap algorithm. Science and Technology Daily, 33(5), 167-170.

[3] Xu, W., Peng, Z., & Zuo, J. (2015). Research on cloud computing resource scheduling strategy based on genetic algorithm. Computer Measurement and Control, 23(5), 1653-1656.

[4] Rathor, V. S., Pateriya, R. K., & Gupta, R. K. (2015). An efficient virtual machine scheduling technique in cloud computing environment. International Journal of Modern Education and Computer Science, 7(3), 39.

[5] Kruekaew, B., & Kimpan, W. (2020). Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. International Journal of Computational Intelligence Systems, 13(1), 496-510.

[6] Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2020). FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing. Journal of Ambient Intelligence and Humanized Computing, 11(10), 3975-3987.

[7] Farahnakian, F., Ashraf, A., Pahikkala, T., Liljeberg, P., Plosila, J., Porres, I., & Tenhunen, H. (2014). Using ant colony system to consolidate VMs for green cloud computing. IEEE transactions on services computing, 8(2), 187-198.

[8] Liu, X. F., Zhan, Z. H., Deng, J. D., Li, Y., Gu, T., & Zhang, J. (2016). An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE transactions on evolutionary computation, 22(1), 113-128.

[9] Alharbi, F., Tian, Y. C., Tang, M., Zhang, W. Z., Peng, C., & Fei, M. (2019). An ant colony system for energy-efficient dynamic virtual machine placement in data centers. Expert Systems with Applications, 120, 228-238.

[10] Wei, X. (2020). Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 1-12.

[11] Mirjalili, S., Song Dong, J., Sadiq, A. S., & Faris, H. (2020). Genetic algorithm: Theory, literature review, and application in image reconstruction. Nature-inspired optimizers: Theories, literature reviews and applications, 69-85.

[12] Quang-Hung, N., Nien, P. D., Nam, N. H., Huynh Tuong, N., & Thoai, N. (2013, March). A genetic algorithm for power-aware virtual machine allocation in private cloud. In Information and communication technology-EurAsia conference (pp. 183-191). Berlin, Heidelberg: Springer Berlin Heidelberg.

[13] Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. Evolutionary algorithms and neural networks: theory and applications, 43-55.

[14] Mirjalili, S., Song Dong, J., Sadiq, A. S., & Faris, H. (2020). Genetic algorithm: Theory, literature review, and application in image reconstruction. Nature-inspired optimizers: Theories, literature reviews and applications, 69-85.

[15] Haldurai, L., Madhubala, T., & Rajalakshmi, R. (2016). A study on genetic algorithm and its applications. Int. J. Comput. Sci. Eng, 4(10), 139-143.

[16] Immanuel, S. D., & Chakraborty, U. K. (2019, July). Genetic algorithm: An approach on optimization. In 2019 international conference on communication and electronics systems (ICCES) (pp. 701-708). IEEE.

[17] Li, X., & Yang, G. (2016). Artificial bee colony algorithm with memory. Applied Soft Computing, 41, 362-372.

[18] Bansal, J. C., Sharma, H., & Jadon, S. S. (2013). Artificial bee colony algorithm: a survey. International Journal of Advanced Intelligence Paradigms, 5(1-2), 123-159.

[19] Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of intelligent manufacturing, 23, 1001-1014.

[20] Karaboga, D., & Kaya, E. (2016). An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training. Applied Soft Computing, 49, 423-436.