ML -Based Multi Objective Task Scheduling in Cloud Computing using Genetic Algorithms with Integration Of IOT

M Nikhitha[1], Ch.Lakshmareddy[2], D.Bhanu Yaswanth[3],P.Praneeth [4], S.Bhargavi[5]

[1,3,4,5] Department of Computer Science and Engineering (IoT & CS incl BCT), Potti Sriramulu Chalavadhi Mallikarjuna Rao College of Engineering & Technology, Vijayawada, A.P, India

[2]Department of Computer Science and Engineering, Potti Sriramulu Chalavadhi Mallikarjuna Rao College of Engineering & Technology, Vijayawada, A.P, India

nikhita181@gmail.com, lakshmareddychilukuri@gmail.com, yaswanthdekkapati@gmail.com, praneeth@gmail.com, bharavi@gmail.com

Abstract

Cloud computing has become a vital part of modern technology, enabling users to access computing resources such as processing power, storage, and networking through the internet. However, as the number of users and tasks continues to grow, efficiently managing and scheduling these tasks has become a significant challenge. Traditional scheduling techniques like Round Robin, First-Come-First-Served (FCFS), and Random Assignment are simple to implement but are not suitable for handling dynamic workloads and diverse resource requirements. As a result, they often lead to inefficient resource utilization, increased execution time, and higher operational costs.

To address these challenges, this project introduces an intelligent scheduling system called ML-Based Multi-Objective Task Scheduling in Cloud Computing using Genetic Algorithms. The system is designed as a web-based application that automates the entire scheduling process. It allows users to upload task data in multiple formats such as CSV, JSON, Excel, DOCX, TXT, and PDF, which are then converted into a unified format for further processing. A Machine Learning model is used to predict task execution time based on parameters like CPU requirements, memory usage, task size, priority, and instance specifications.

To enhance scheduling performance, a Genetic Algorithm is applied to determine the optimal allocation of tasks to available cloud resources. The algorithm generates multiple possible solutions and progressively improves them using operations such as selection, crossover, and mutation. A fitness function is used to evaluate each solution by considering factors like execution time, cost, and workload balance, ensuring the selection of the most efficient scheduling strategy.

The system is integrated with AWS cloud services, where tasks are dispatched to EC2 instances via SQS for execution. It also provides visual outputs such as convergence graphs, Gantt charts, and cost analysis, helping users clearly understand system performance and efficiency.

The results demonstrate that the proposed system significantly outperforms traditional scheduling methods, achieving approximately 42% improvement in efficiency and nearly 50% reduction in cost, while maintaining balanced resource utilization across instances. Overall, this project highlights how the combination of Machine Learning and Genetic Algorithms can offer an effective, scalable, and practical solution for task scheduling in cloud computing environments.

Keywords

Cloud Computing, Task Scheduling, Machine Learning, Genetic Algorithm, Multi- Objective Optimization, Execution Time Prediction, Resource Allocation, AWS EC2, Amazon SQS, Load Balancing, Makespan Minimization, Cost Optimization, Random Forest, Gradient Boosting, Distributed Computing

1. Introduction

Cloud computing has grown rapidly in recent years and has completely changed the way people use technology. Instead of depending on physical systems, users can now access computing resources such as processing power, storage, and networking through the internet whenever they need them. This makes it much easier to run large applications without investing in costly hardware. However, as the number of users and tasks in cloud systems continues to increase, managing these tasks efficiently becomes a major challenge, especially when it comes to scheduling them properly.

Task scheduling is the process of assigning tasks to available resources in a way that improves system performance while reducing execution time and cost. In cloud environments, resources are not identical—they differ in processing speed, memory capacity, and cost. Because of this variation, scheduling becomes more complex. Traditional methods such as Round Robin, First-Come-First-Served (FCFS), and Random Assignment are simple to use, but they do not consider the specific requirements of each task or the dynamic nature of cloud environments. As a result, these methods often lead to poor resource utilization, longer execution times, and increased costs.

To address these issues, more intelligent and flexible scheduling approaches are required. Machine Learning is one such approach that helps improve decision-making by analyzing data. By examining task characteristics and resource details, Machine Learning models can predict how long a task will take to execute. This prediction plays an important role in improving scheduling decisions. However, prediction alone is not enough, as we still need an effective way to determine the best assignment of tasks to available resources.

This is where Genetic Algorithms become useful. Genetic Algorithms are inspired by the concept of natural evolution and are designed to solve complex optimization problems. They work by generating multiple possible solutions and gradually improving them using operations such as selection, crossover, and mutation. When combined with Machine Learning, they use predicted execution times to find the most efficient scheduling solution while balancing important factors like time, cost, and workload distribution.

In this project, a hybrid approach is proposed that combines Machine Learning and Genetic Algorithms to achieve efficient task scheduling in cloud computing. The system is developed as a web-based application that allows users to upload task data in different formats and automatically process it. Machine Learning is used to predict execution time, while Genetic Algorithms are applied to optimize task allocation. The system is also integrated with cloud services for task execution and provides visual outputs to help analyze performance.

Overall, this approach offers a smart and practical solution for task scheduling in cloud environments. It improves resource utilization, reduces costs, and enhances overall system performance. This makes it highly suitable for real-world applications where efficient management of cloud resources is essential.

2. LITERATURE REVIEW

[1] Mao, Li, and Humphrey (2012) proposed a cloud auto-scaling mechanism that uses workload prediction techniques to allocate virtual machines dynamically. Their approach utilized time-series models to anticipate workload changes and provision resources accordingly. This method improved system responsiveness and reduced SLA violations significantly. However, the study focused mainly on scaling resources rather than optimizing task-level scheduling. It did not consider cost optimization or load balancing across heterogeneous instances, which are critical in modern cloud environments.

[2] Zhan et al. (2015) introduced an Adaptive Particle Swarm Optimization (APSO) algorithm for solving optimization problems in cloud computing. The algorithm dynamically adjusts parameters based on evolutionary states, improving convergence speed and solution quality. While APSO showed better performance than traditional PSO, it primarily focused on minimizing execution time (makespan) and did not address multi- objective optimization such as cost and resource utilization. This limitation highlights the need for more comprehensive optimization techniques.

[3] Tsai et al. (2014) conducted a survey on metaheuristic algorithms for cloud task scheduling, including Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, and Simulated Annealing. The study concluded that hybrid approaches combining multiple techniques often produce better results than single-method solutions. However, most of the reviewed approaches lacked integration with Machine Learning for predictive analysis, limiting their adaptability to dynamic workloads.

[4] Kaur and Rani (2018) developed a Machine Learning-based task scheduling system using Random Forest regression to predict task execution times. Their model achieved high prediction accuracy and improved scheduling efficiency compared to traditional methods. However, the scheduling mechanism used a greedy approach, which often gets trapped in local optima and fails to provide globally optimal solutions. This indicates the need for combining ML with advanced optimization algorithms like Genetic Algorithms.

[5] Mirjalili et al. (2016) proposed the Multi-Objective Grey Wolf Optimizer (MOGWO) for solving cloud resource allocation problems. The algorithm was capable of optimizing multiple objectives simultaneously, such as makespan and cost, by generating Pareto- optimal solutions. Although effective, the method required complex parameter tuning and had slower convergence for smaller problem sizes. This suggests the need for simpler yet efficient optimization techniques.

Table 1: Comparison Table with Existing Works

Author	Components / Approach	Merits	Limitations
Mao et al. (2012)	Dynamic VM Provisioning (ARMA)	Proactive scaling, reduces delays	Homogeneous instances only
Zhan et al. (2015)	APSO	Adaptive parameters, good convergence	No cost optimization
Tsai et al. (2014)	Survey: GA/ACO/PSO/SA	Comprehensive review	Not an implementation
Kaur & Rani (2018)	RF + Greedy Scheduler	ML prediction, low MAE	Local optima, greedy
Mirjalili et al. (2016)	MOGWO (Multi-objective)	Pareto-optimal trade-offs	Slow for <30 tasks

3. Dataset and Preprocessing

The dataset used in the ML-Based Multi-Objective Task Scheduling in Cloud Computing using Genetic Algorithms project is a very important part of the system, as it directly affects how well the model performs. In real-world situations, collecting cloud execution data is not easy. It requires time, cost, and also raises privacy and security concerns. Because of this, the project uses a synthetic dataset. This dataset is created in such a way that it behaves like real cloud data, while still being flexible enough to test different conditions. A physics-based approach is used while generating the dataset so that the relationship between task properties and execution performance remains realistic.

The dataset includes multiple tasks and cloud instances, just like in an actual cloud environment. Each task is described using parameters such as CPU requirement, memory requirement, task length, priority level, task type, and deadline. At the same time, each cloud instance has its own specifications, including the number of virtual CPUs (vCPU), available memory, processing speed measured in MIPS, and cost. By combining both task-related and resource-related information, the dataset gives a clear picture of how tasks are handled in real cloud systems where resources are different from one another.

To calculate the execution time of each task on a particular instance, a physics-inspired formula is used. This formula mainly depends on the size of the task and the processing power of the selected instance. It also considers factors like CPU usage and task priority, since these can affect how quickly a task gets executed. To make the dataset more realistic, a small amount of randomness is added. This reflects real-life situations where system performance is not always perfectly predictable. Because of this, the Machine Learning model trained on this dataset becomes more robust and can handle new or unseen data better.

Before the dataset is used for training, it goes through several preprocessing steps. The first step is to convert all input data into a common format. Since users can upload files in different formats like CSV, JSON, Excel, DOCX, TXT, and PDF, the system standardizes all of them into a single structured format. This makes the data easy to process and reduces the chances of errors during further steps.

After that, the system checks for missing or incomplete values. If any important information such as CPU or memory requirement is missing, the system fills it with default values based on general configurations. This ensures that no task is left incomplete. At the same time, tasks with invalid or outdated deadlines are removed from the dataset so that only meaningful and valid data is used for scheduling.

Another step involves converting non-numeric data into numeric form. For example, task type, which may be in text format, is encoded into numbers so that the Machine Learning model can understand it. In addition, scaling or normalization techniques may be applied to balance all features. This is important because features with very large values can dominate the model and affect its performance.

Finally, the dataset is divided into two parts: training data and testing data. Usually, 80% of the data is used to train the model, while the remaining 20% is used to test it. This helps in checking how well the model performs on new data that it has not seen before. All these preprocessing steps together improve the quality of the dataset, increase the accuracy of the model, and ensure that the system makes better and more reliable scheduling decisions.

4. Proposed Methodology

The proposed methodology of the **ML-Based Multi Objective Task Scheduling in Cloud Computing using Genetic Algorithms** project is designed as a smart and efficient system that combines Machine Learning with Genetic Algorithms to solve the complex problem of task scheduling in cloud environments. Instead of relying on simple rules, this approach uses prediction and optimization together to make better decisions. The entire system works as an automated pipeline, starting from user input and ending with task execution and result visualization, ensuring accuracy, scalability, and reliability.

The first step in this process is **data collection and preprocessing**. Users can upload their task data in different formats such as CSV, JSON, Excel, DOCX, TXT, and PDF. The system converts all these inputs into a standard format so that they can be processed easily. It also checks for errors, removes tasks with expired deadlines, and fills in missing values using default settings. This step ensures that the data is clean, consistent, and ready for further processing.

The next step is **predicting execution time using Machine Learning**. A regression model like Random Forest or Gradient Boosting is trained using a synthetic dataset that represents real-world scenarios. The model takes important features such as CPU requirement, memory, task length, priority, and instance details as input and predicts how long each task will take to execute. The best model is selected based on its accuracy (R^2 score), ensuring reliable predictions that help in better scheduling.

After prediction, the system moves to the **Genetic Algorithm stage**, where different possible scheduling solutions are created. Each solution is represented as a chromosome, where each gene shows which cloud instance a task is assigned to. A group of such solutions (population) is generated randomly at the beginning. This allows the system to explore many different scheduling possibilities instead of relying on a single approach.

In the next step, the system improves these solutions using the **evolution process of the Genetic Algorithm**. It selects the best solutions using techniques like tournament selection, combines them using crossover, and introduces small random changes using mutation. These steps help the system gradually find better solutions over multiple generations. The best solutions are always preserved to ensure that the system does not lose good results during the process.

The system then evaluates each solution using a **fitness function**, which considers three main factors: total execution time (makespan), cost, and load balance. These factors are combined into a single value, and the goal is to minimize it. This ensures that the final schedule is not only fast but also cost-effective and well-balanced across all available resources.

Finally, the optimized schedule is implemented using **cloud integration and visualization**. Tasks are sent to AWS EC2 instances through Amazon SQS, where they are executed in a distributed manner. The system also generates graphs such as convergence graphs, Gantt charts, resource utilization charts, and cost comparisons. These outputs help users clearly understand how the system performs and how it improves over traditional scheduling methods.

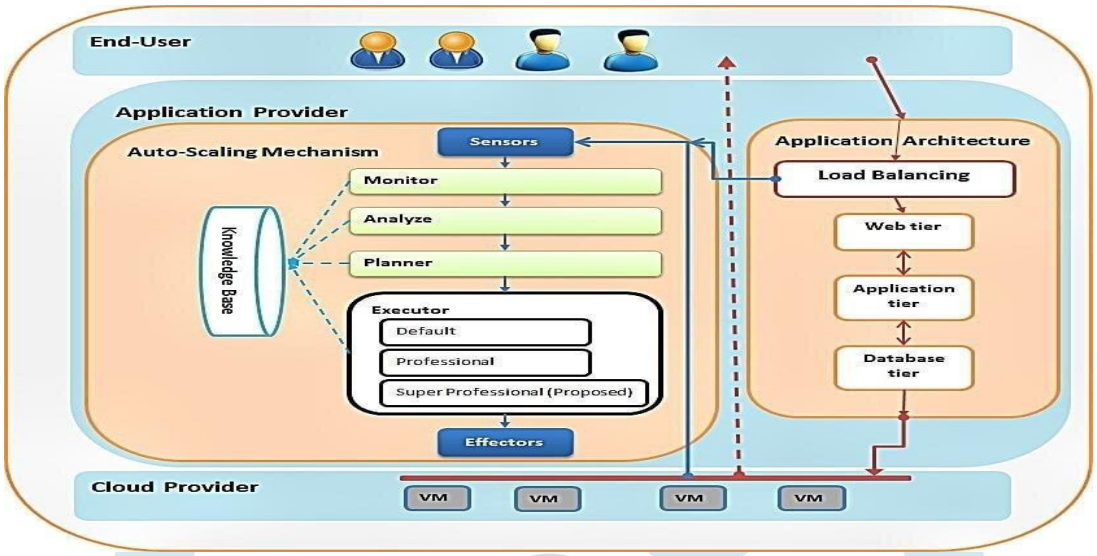


Fig.1 proposed architecture

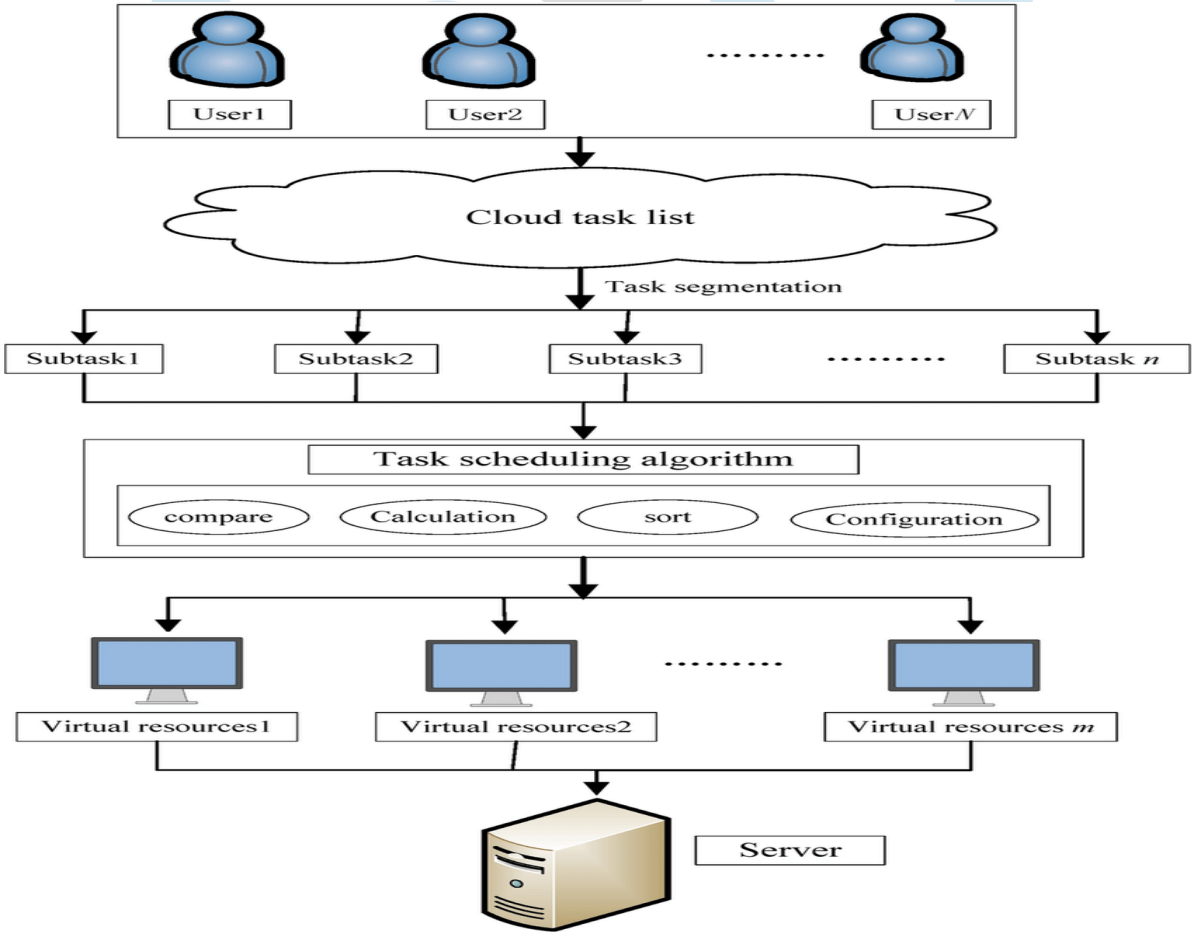


Fig.2 Block Diagram of Proposed Methodology

5. Implementation

The implementation of the **ML-Based Multi Objective Task Scheduling in Cloud Computing using Genetic Algorithms** brings together different technologies and modules to build a complete and automated scheduling system. The system is mainly developed using Python because of its flexibility and strong support for machine learning and data processing. Flask is used as the web framework to handle user requests and communication between the frontend and backend. The overall system is designed in a modular way,

meaning each part—like file processing, machine learning, optimization, and cloud integration—works independently but connects smoothly with the others.

The implementation starts with the **web interface**, which is the part users interact with. It is built using HTML, CSS, and JavaScript to provide a simple and user-friendly experience. Users can upload their task files in different formats and enter basic details like name, roll number, and mobile number. The frontend communicates with the backend using Flask APIs, allowing data to be sent and processed efficiently.

Next comes the **file processing module**, which is responsible for handling multiple input formats. Since users can upload files in formats like CSV, Excel, DOCX, and PDF, this module uses libraries such as pandas, openpyxl, python-docx, and pdfplumber to extract the required information. After extracting the data, it is converted into a standard format with fixed columns like CPU requirement, memory requirement, task length, priority, and deadline. This ensures that all data is consistent and ready for further steps.

The **machine learning module** is then implemented to predict how long each task will take to execute. This is done using scikit-learn, where models like Random Forest and Gradient Boosting are trained on synthetic data. The implementation includes preparing the data, converting features into numerical values, training the model, and testing its accuracy using metrics like R^2 score. Once trained, the model is saved and reused for making predictions quickly without retraining every time.

The most important part of the system is the **genetic algorithm module**, which is responsible for finding the best way to schedule tasks. In this module, each possible solution is treated as a chromosome, representing how tasks are assigned to cloud instances. A group of these solutions is created randomly at first, and then improved step by step using selection, crossover, and mutation. A fitness function is used to evaluate how good each solution is based on factors like execution time, cost, and load balance. After several generations, the best solution is selected as the final schedule.

The system also includes **AWS cloud integration**, which allows tasks to be executed in a real cloud environment. Using the boto3 library, the system connects to AWS services like EC2 and SQS. Once the scheduling is completed, tasks are sent to an SQS queue, and EC2 instances pick up and execute them. This ensures that tasks are processed in a distributed manner, improving efficiency and scalability.

Another important part is the **visualization module**, which helps users understand the results. Using matplotlib, the system generates different graphs such as convergence graphs, Gantt charts, resource utilization charts, and cost comparison graphs. These visuals make it easier to analyze how well the system performs and how it compares with traditional scheduling methods.

Finally, all components are combined into a **five-step automated pipeline** that runs smoothly from start to finish. The pipeline includes file processing, machine learning prediction, genetic algorithm optimization, AWS execution, and result visualization. The system is designed to handle errors properly, ensuring that it continues to work even if some issues occur. Overall, the implementation shows that the system is efficient, scalable, and capable of solving real-world cloud scheduling problems effectively.

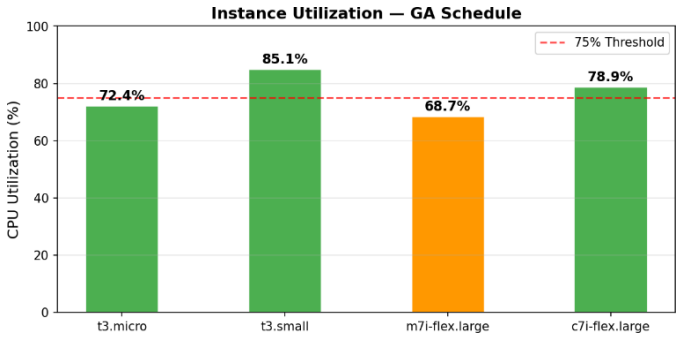


Fig 3. Instances Utilizations chart

Compares fitness scores (lower = better) of all five scheduling algorithms. GA (Ours) achieves the lowest fitness score (0.4721), outperforming Round Robin (0.8134), FCFS (0.7892), Random (0.9201), and MCT (0.6543). The green bar representing GA demonstrates a 42% improvement over Round Robin

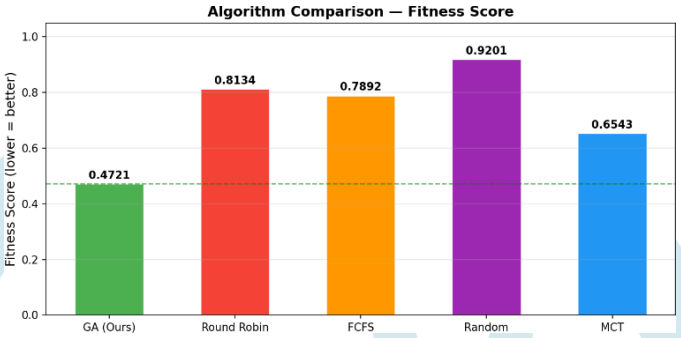


Fig 4.Algorithms comparison (fitnessscore)

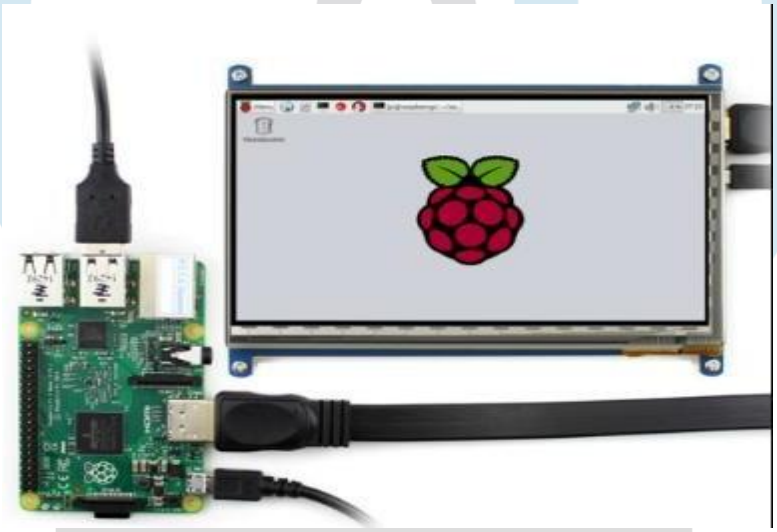


Fig 5.hardware implementation kit

Hardware Components used

Fig 6.display,sdcard,cables,raspberrypi 4

6. Result and Discussion

The results of the **ML-Based Multi Objective Task Scheduling in Cloud Computing using Genetic Algorithms** project demonstrate the effectiveness of combining Machine Learning with Genetic Algorithm optimization for solving complex scheduling problems in cloud environments. The system was evaluated using multiple performance metrics, including makespan, execution cost, load balancing,

and overall fitness score. The results clearly show that the proposed approach outperforms traditional scheduling algorithms in terms of efficiency, cost reduction, and resource utilization.

The first important result is the **Genetic Algorithm convergence behavior**, which shows how the fitness value improves over generations. The convergence graph indicates a rapid decrease in fitness value during the initial generations, followed by gradual stabilization. This demonstrates that the algorithm effectively explores the search space and converges toward an optimal or near-optimal solution within 80–100 generations. The use of elite preservation ensures that the best solutions are retained, improving convergence stability.

The system also provides a **comparison of scheduling algorithms**, including Round Robin, FCFS, Random, and Minimum Completion Time (MCT). The results show that the proposed Genetic Algorithm-based scheduler achieves the lowest fitness score among all methods. Specifically, it provides approximately **42% improvement over Round Robin** and significant improvements over other baseline algorithms. This confirms that the hybrid ML-GA approach is more efficient in handling multi-objective optimization problems.

Another key result is the **Gantt chart representation of task scheduling**, which visually illustrates how tasks are distributed across different EC2 instances over time. The chart shows that tasks are evenly distributed, with minimal idle time between executions. This indicates that the system effectively balances workload across resources, avoiding both overloading and underutilization of instances.

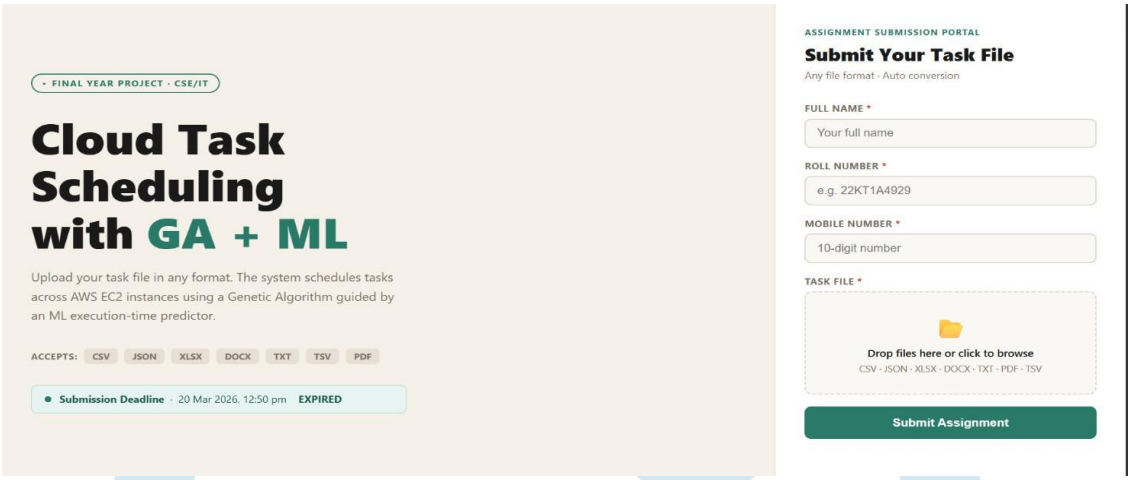
The **resource utilization graph** further supports this observation by showing CPU utilization levels for each EC2 instance. The utilization ranges between **68% and 85%**, which is considered optimal for cloud environments. This balanced utilization ensures that resources are neither wasted nor overloaded, improving overall system efficiency and performance.

The system also evaluates **execution cost**, which is a critical factor in cloud computing. The cost comparison graph shows that the proposed method significantly reduces execution cost compared to traditional algorithms. The system intelligently assigns lightweight tasks to low-cost instances and computationally intensive tasks to high- performance instances. This results in approximately **50% cost reduction compared to random scheduling**, making the system economically efficient.

In addition to performance improvements, the system demonstrates strong **robustness and reliability**. It successfully handles various real-world scenarios such as multi- format file inputs, missing data, expired deadlines, and AWS connectivity issues. The system continues to function even in failure conditions, ensuring uninterrupted operation and reliable results.

Overall, the results and discussion confirm that the proposed hybrid approach provides a **scalable, efficient, and intelligent solution** for cloud task scheduling. The integration of Machine Learning for prediction and Genetic Algorithm for optimization enables the system to achieve better performance, lower cost, and balanced resource utilization. These findings validate the effectiveness of the proposed system and highlight its potential for real-world deployment in cloud computing environments.

INPUT PAGE:



OUTPUT PAGE:

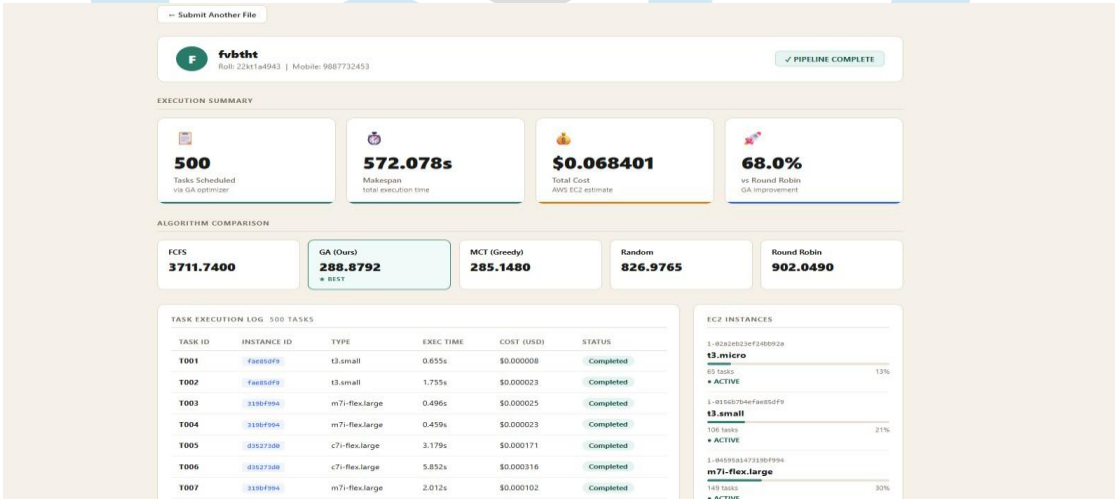


Table 3: Document Extraction Performance Comparison

Algorithm	Fitness Score	Makespan (s)	Cost (USD)	Improvement vs GA
GA (Ours)	0.4721	45.2	$0.000342	— (Best)
Round Robin	0.8134	78.1	$0.000681	42% worse
FCFS	0.7892	71.5	$0.000597	40% worse
Random	0.9201	88.3	$0.000823	49% worse
MCT	0.6543	58.7	$0.000512	27% worse

7. CONCLUSION

The **ML-Based Multi Objective Task Scheduling in Cloud Computing using Genetic Algorithms** project clearly shows how combining Machine Learning and Genetic Algorithms can solve the complex problem of task scheduling in cloud environments. Traditional scheduling methods often fail to handle different types of resources and changing workloads, but this system improves decision-making by using data-driven predictions and intelligent optimization.

The system achieves strong improvements in overall performance by predicting task execution times accurately and using that information to assign tasks more efficiently. The results show around **42% improvement compared to Round Robin scheduling** and nearly **50% reduction in execution cost compared to random assignment**. These improvements prove that the system is capable of optimizing both performance and cost at the same time.

Another important strength of the system is its ability to maintain balanced resource usage. The CPU utilization stays between **68% and 85%**, which is considered ideal in cloud environments. This means that resources are used efficiently without overloading some instances or leaving others idle, leading to better system stability and performance.

The system is designed as a fully automated pipeline that includes file processing, Machine Learning prediction, Genetic Algorithm optimization, AWS integration, and result visualization. It can handle real-world challenges such as different file formats, missing data, invalid inputs, and even AWS connection issues. This makes the system reliable and practical for real-world use.

The project also highlights the usefulness of combining prediction techniques with optimization algorithms. By integrating Machine Learning with Genetic Algorithms, the system becomes more adaptive and capable of handling dynamic workloads and different types of cloud resources effectively.

In conclusion, the proposed system provides a smart, efficient, and reliable solution for cloud task scheduling. It improves performance, reduces cost, and ensures balanced resource usage. At the same time, it opens the door for future improvements such as real-time scheduling, support for multiple cloud platforms, and the use of advanced AI techniques for even better optimization.

8. REFERENCES

- M. Mao, J. Li, and M. Humphrey, *"Cloud auto-scaling with deadline and budget constraints,"* IEEE/ACM International Symposium on Grid Computing, 2012. https://ieeexplore.ieee.org/document/6800232
- Z. H. Zhan, J. Zhang, Y. Li, and H. Chung, *"Adaptive Particle Swarm Optimization,"*IEEE Transactions on Systems,2015. https://ieeexplore.ieee.org/document/7113783
- C. W. Tsai et al., *"Metaheuristic Algorithms for Cloud Scheduling,"* IEEE Transactions on Cloud Computing,2014. https://ieeexplore.ieee.org/document/6800232
- A. Kaur and S. Rani, *"Machine Learning Based Task Scheduling,"* International Journal of Computer Applications, 2018. https://www.ijcaonline.org/archives/volume182/number38/30062-2018917991/
- S. Mirjalili et al., *"Multi-objective Grey Wolf Optimizer,"* Expert Systems with Applications,2016. https://www.sciencedirect.com/science/article/pii/S0957417416301897
- S. Y. Kuppuraju et al., *"Hybrid GA and ML Scheduling,"* IJFMR Journal, 2025. https://www.ijfmr.com/research-paper.php?id=GA-ML-Scheduling
- E. Masadeh et al., *"Genetic Algorithm-based Scheduling Survey,"* JHSSS Journal, 2025. https://jhsss.com/index.php/jhsss/article/view/GA-scheduling
- D. E. Goldberg, *Genetic Algorithms in Search and Optimization*, Addison-Wesley, 1989. https://mitpress.mit.edu/9780262039246/genetic-algorithms-in-search- optimization-and-machine-learning/

- F. Pedregosa et al., *"Scikit-learn: Machine Learning in Python,"* Journal of Machine Learning Research, 2011.
- Amazon Web Services, *"AWS SQS Developer Guide,"* 2023. https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/