DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

INTERNATIONAL JOURNAL OF SCIENCE FOR GLOBAL SUSTAINABILITY
(A PUBLICATION OF FACULTY OF SCIENCE, FEDERAL UNIVERSITY GUSAU, NIGERIA)

Cloud-Based Resource Management Optimization Model based on Twin-Dynamic Programming Algorithms: A Simulation-Based Approach

Shehu Abdulwahab[1], Etemi Joshua Garba[2] and Mohammed Bashir Ribadu[1]
[1]Department of Information Technology, Modibbo Adama University,Yola
[2]Department of Computer Science, Modibbo Adama University Yola
Corresponding Author's Email & Phone No.: abdulwahabshehu@mau.edu.ng, +2348036989864

Received on: October, 2025 **Revised and Accepted on:** November, 2025 **Published on:** December, 2025

ABSTRACT

Cloud computing has emerged as a paradigm for delivering scalable and on demand computing resources over the internet. However, efficient resource management remains a significant challenge in cloud computing environments. This study proposes a novel cloud-based resource management optimization model based on twin-dynamic programming algorithm, which combines the strengths of dynamic programming and simulation based optimization. The proposed model aims to optimize resource allocation and minimize cost in cloud computing environments. We implement the proposed model using the CloudSim simulator evaluate its performance under various workload scenarios. The simulation results demonstrate that the model outperforms existing traditional resource management approaches in terms of resource utilization, execution time, throughput, makespan, cost savings and quality of service (QOS). The proposed model provides a promising solution for cloud providers to optimize resource management and improve the efficiency of cloud computing environments. Future studies can explore the proposed model in real world computing environments and investigate the impact of different workload patterns and system configuration on the performance of the proposed model.

Keywords: Cloud Computing, Resource Management, Dynamic Programming, Cloud Simulation, Optimization Model

1.0 INTRODUCTION

The management of resources in digital environment is fundamental to cloud computing. The use of highly scalable distributed infrastructure for processing, storage, transmission, and accessibility has gathered attention by researchers across the globe. The emanation of cloud computing have advanced the support in optimizing the organization of those resources that drives the need to implement and/or integrate IT systems, infrastructures, and platforms to connect the physical and digital world as an ecosystem (Ghedass and Ben, 2021). This provides a distributed, adaptive, open socio-technical system with properties of virtualization, self-organization, scalability and sustainability inspired from natural ecosystems (Saleh & Abel, 2015). This aims to transfer the workload of managing the data from physical to the cloud as much as possible to ensure reliability, performance, and efficiency requirements. Such developments have led to many issues and future challenges of the cloud adoption where complexity of services and data grow exponentially (Llwaah, 2024).

Furthermore, an important factor that depicts the overall performance of the cloud computing is the use of parallel task and heterogeneous resources in complex workflow. Hence, this must ensure that computing tasks are completed in parallel and efficiently, where the resource allocation minimizes time and maximizes the utilization of available resources (Mukherjee, 2024). Consequently the essential feature that describes the quality of a Cloud computing system is its scalability nature where the persistent challenge is to achieve higher performance and greater scalability in managing data and Cloud infrastructure that optimized workload distribution (Loncar & Loncar, 2022,). Hence, some systems do not scale to the volume of data requested by the data-intensive application and as well the suitable metrics when evaluating its scalability and the cloud infrastructures are not fully defined. The concept of scalability in cloud computing denotes to the effectiveness of optimization methods/techniques (Ghedass & Ben, 2021). These have urge for algorithms and advanced digital processes in managing data resources. The emerging use of such methods/techniques coupled with a software-defined approach to resource management creates space for prompt and agile customization and management of the dynamic needs of workflows, which have address the optimal use of resources for dynamically requested services with an intelligent and centrally controlled approach (Mahmoud et al., 2021).

The main question this paper posed to answer is how to optimize centered resource management considering

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

scalability in cloud environments using the task scheduling approach. It delves on the performance evaluation based on analysis of the cloud facility in executing resource-intensive tasks. This work leverages dynamic programming algorithm approach to execute heterogeneous data processing tasks on large-scale resource infrastructures through simulations. A novel task allocation approach in optimizing resource allocation, task scheduling, cost optimization, performance optimization and overall cloud efficiency and productivity using heterogeneous cloud data centers was proposed. An extensive evaluation of the algorithm performance was conducted, and comprehensively compared with the First-Come-First-Serve (FCFS) algorithms approaches solution using a cloud simulator. The paper mainly contributes to knowledge in the following ways:

i. A novel Optimization Model built based on Dynamic Programming algorithm with Longest and Shortest Job First broker policy for the task scheduling problem in Cloud data centers.
ii. Simulation-based performance evaluation of the proposed model.

1.1 Resource Management Optimization in Cloud Computing

Since cloud computing resources are distributed across various locations, its application continues to grow rapidly. Therefore companies, universities and research centers have high-performance needs which lead to the resource management problems. The physical resources are dispatched across multiple compute requests through virtualization and provisioning while the virtual resources (VR) are dynamically assigned and reassigned according to consumer demand (Fredrick & Yan, 2017; Farootani et al., 2020; Aldossary, 2021). All resources are managed in an optimal manner. Optimization in computing is referred to as the selection of best element from a set of alternatives with regard to some criteria in managing the cloud resources (Sumalatha & Anbarasi, 2019). Cloud service provider and cloud consumer are the two major entities in the cloud, whereas the cloud service providers provision their resources on rental basis to cloud consumers and cloud consumers submit their requests for provisioning the resources (Pradeep, et al., 2018). Both entities have their own inspirations of being in the cloud environment. The consumers are concerned with the performance of their applications, whereas providers are more interested in efficient utilization of their resources. The concept of optimization in cloud based ecosystem design refers to the process of refining and adjusting the models architecture, parameters, and other components to achieve the optimal performance, scalability, and efficiency within the cloud based infrastructure (Simi'c et al., 2023). The aim of adjusting the parameters/components is primarily to achieve better results that improve performance and efficiency to the cloud user, and efficient utilization of resource to the cloud service provider.

Optimization is a critical aspect of designing and developing cloud-based system as it bridge the gap between the model and real world deployment in order to reach its full potentials (Sumalatha & Anbarasi, 2019). The best technique/algorithm in optimizing models depends on the specific requirements and constraints of the entire system/model (Farootani et al., 2021). The most popular techniques/algorithms for optimization target an aspect of resources in the cloud environment specifically in resource management and provisioning. Some of such algorithms include:

I. **Linear programming (LP):** Linear programming is a commonly used algorithm for optimization in cloud-based systems (Mahmoud et al., 2021). It focuses solely on resource allocation and scaling (vertical/horizontal).

II. **Integer programming (IP)**: IP is an extension of linear programming that focus on resource allocation decisions (Sumalatha & Anbarasi, 2019). It uses integer variables in optimizing discrete resource allocation decision that fits the cloud-based system.

III. **Dynamic programming (DP):** DP is a powerful technique for optimizing complex systems, breaking them into smaller interdependent components to ease computation process (Sumalatha & Anbarasi, 2019).

IV. **Genetic Algorithm (GA):** GA is a metaheuristic algorithm inspired by bio-inspired operators such as mutation, crossover and selection suitable for optimizing complex system commonly used to generate high-quality solutions to optimization and search problems (Wang, 2016; Hu et al., 2017). Genetic algorithms do not scale well with complexity, and operating on dynamic data sets is difficult (Sumalatha & Anbarasi, 2019).

V. **Simulated Annealing (SA):** SA is also a metaheuristic algorithm that uses temperature schedule to control the exploration and exploitation of system tradeoffs (Liwaah, 2024). It is appropriate for optimization of complex systems with multiple local optima.

VI. **Ant Colony Optimization (ACO):** ACO is a metaheuristic algorithm stimulated with the concept of ant colonies. It is also suitable for optimization of complex systems with multiple optima (Ruonan Li, 2016; Ashish & Ritu, 2017).

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

VII. **Particle Swarm Optimization (PSO)**: PSO is a metaheuristic algorithm inspired by grouping behavior, suitable for optimization of complex systems with multiple optima (Zhifeng et al., 2016).

VIII. **Reinforcement learning (RL):** RL is a machine learning algorithm that learns to make decisions by interacting with an environment (internal/external) (Sumalatha & Anbarasi, 2019; Tuli et al., 2022; Moazeni et al., 2022). It is appropriate for optimizing complex systems with dynamic feedback.

Important factors are used to determine the best algorithm that suits a cloud environment resource management, such factors depend on the: problem complexity, scalability, computational efficiency, solution quality, robustness and interoperability of the system (Pradeep, et al., 2018; Loncar & Loncar, 2022; Trimarchi et al., 2025). Cost, latency, throughput, and reliability are the essential metrics in evaluating and comparing the performance of different techniques to determine the best algorithm for any model/system optimization (Pradeep, et al., 2018). Hence, the efficiency of the model depends on the technique applied.

1.2 Dynamic Programming (DP)

Dynamic Programming is originated by Richard Bellman in 1950s as one of the powerful algorithmic techniques for solving complex problems. It breaks a problem into smaller sub-problems, by solving each sub problem only once and stores the solutions to the sub-problems to avoid redundant computation or iteratively builds solutions (Sumalatha & Anbarasi, 2019). It is applied for optimization problems by finding the shortest paths, minimum spanning trees, and maximum flow (Souza et al., 2022). It is evident that dynamic programming is characterized with the principle of Top-down (memorization), and bottom-up (tabulation) approach in optimization. While both types share the core principles of DP – overlapping sub-problems, Bellman's principle of optimality, and solution storage: they differ in their approach to solving problems.

A. Memoization (Top-Down) Approach: Memoization utilizes recursion to break down the main problem into smaller sub-problems, where a function is defined for each sub-problem, and solutions are stored in a data structure (memo) to avoid redundant calculations (Sumalatha & Anbarasi, 2019). Memoization approach can be more intuitive for problems with a natural recursive structure, and its memory usage might be efficient for problems with a limited number of unique subproblems. It may result to recursive calls can incur overhead, and may not be suitable for problems with a vast number of sub-problems, leading to excessive function calls and potential stack overflow errors (Gonzalez et al., 2017).

B. Tabulation (Bottom-Up) Approach: Tabulation builds solutions using the ground-up principle, iteratively solving smaller sub-problems and storing the results in a table. The solution to the larger problem is then constructed from the pre-computed sub-problem solutions (Sumalatha & Anbarasi, 2019). Tabulation approach is generally faster and more memory efficient for problems with many sub-problems, and is easier to parallelize for potential speedup on multi-core processors. It may as well require more upfront design effort to identify the order of sub-problem solutions, and may not be as intuitive as memoization for problems with a strong recursive nature (Gonzalez et al., 2017).

The choice between memoization and tabulation depends on the specific problem characteristics in terms of the problem structure, number of sub-problems and memory constraints (Mahmood et al., 2021). The problem structure that can naturally lend itself to a recursive breakdown is fit for memorization. A problems with a large number of sub-problems, tabulation's iterative approach is often more efficient. While if memory is a concern, tabulation's table-based storage might be preferable, particularly for problems with many sub-problems.
Generally dynamic programming has to do with optimization in finding shortest path, maximizing profit/minimizing cost, and discovering the maximum/minimum range of the algorithmic query. Wang et al., (2020) proposes a dynamic programming process that follows a finite sequence of task to achieve the desired principles. These steps include:

i. Define the Problem: Identify the problem and its constraints.
ii. Breakdown the problem: divide the problem into smaller sub problems.
iii. Solve sub problems: solve each sub-problem and store its solution
iv. Combine solutions: combine the solutions of sub-problems to obtain the solution to the original problem.

Dynamic Programming is set to provide efficient computation, scalability, optimal performance, real time optimization, cost saving, flexibility, adaptability, and simplified management of

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

resources when implemented (Souza et al., 2022). Conversely, it can be associated to disadvantages like: complexity, dependence on cloud provider, high cost, debugging challenge and Security concern and limited controls.

2.0 MATERIALS AND METHOD(S)

2.1 Material

The study adheres to experimental study methodology where the proposed Dynamic Programming algorithm is extensively evaluated based on simulation environment with heterogeneous workloads as depicted in Table I. The experiments were performed by CloudSim version 4.0 using the Eclipse environment and executed in a 64-bit Windows 10 Operating System running on an Intel Core i7 with 8 GB of RAM.

2.2 Method

The Method section describes in detail how the study was conducted, including conceptual and operational definitions of the variables used in the study, Different types of studies will rely on different methodologies; however, a complete description of the methods used enables the reader to evaluate the appropriateness of your methods and the reliability and the validity of your results, It also permits experienced investigators to replicate the study, Ifyour manuscript is an update of an ongoing or earlier study and the method has been published in detail elsewhere, you may refer the reader to that source and simply give a brief synopsis of the method in thissection.

The study uses large number of datacenters consisting of heterogeneous large number host computers attached: these include both computers and PDAs devices. The study further used different Virtual Machine (VM) instances that correspond to real life situation in the CloudSim environment. The device (both servers and client) used have very high processor speed, reliable Random Access Memory (RAM) and maximum network bandwidth configuration.

2.3 Experimental Parameters, Configurations and Policy

A number of parameters were used to measure the result reliability and how to format the result in a simple manner for easy understanding. These parameters include: resource provisioning and scheduling, scalability, cost of service and simulation computation and compilation time. The study set those parameters in order to have effectual and reliable result.

Table 1: Experimental Parameters, Configurations and Policy

Main Parameters	Sub Parameters	Value
Data Center Specification	Number of Data Centers	6 (Maximum)
	Name	DC1,DC2,DC3,DC4,DC5,DC6
	Number of Virtual Machines (VM)	100(Maximum)
	Architecture	X86
	OS	Linus
	VMM	Xen
VMs Configurations	MIPS	5,10, 20,50,100
	No. of CPU	10
	RAM (MB)	512
	Bandwidth (Mbps)	1000
	VM scheduling policy	Time shared
	Load Balancing policy	Round Robin
Cloudlets Configuration	Length (MI)	1000 (Maximum)
	File Size (bytes)	300

2.4 Proposed Twin Dynamic-Programming Algorithms Model

It is prominent that most cloud service providers are using the traditional method of First-Come-First-Serve bases of executing tasks. The traditional and widely used FCFS method are characterize with a number of difficulties and challenges. Dynamic programming (DP) was adapted as a powerful algorithmic technique for optimizing model efficiency. It breaks the entire cloudlets into smaller sub problems, and solving each problem once and stores the solutions to sub problems to avoid redundant computation. DP is applied to the cloud environment to optimize resource allocation, task scheduling, cost, performance and improve overall cloud efficiency and productivity. Therefore A number of algorithms are used to test the validity of the model based on the principles of Dynamic programming (Shortest Job First (DP-SJF) and Dynamic programming (Longest Job First(DP-LJF)) strategies against the conventional traditional (First-Come-First-Serve) strategy through the deployment of synthetic, single cloud computing service request initiated by the cloud user (startups). The client

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

service request to the service provider or broker based on Twin-Dynamic Programming algorithm with Longest and Shortest Job First broker policy are process based on workflows as illustrated in Figure 1.

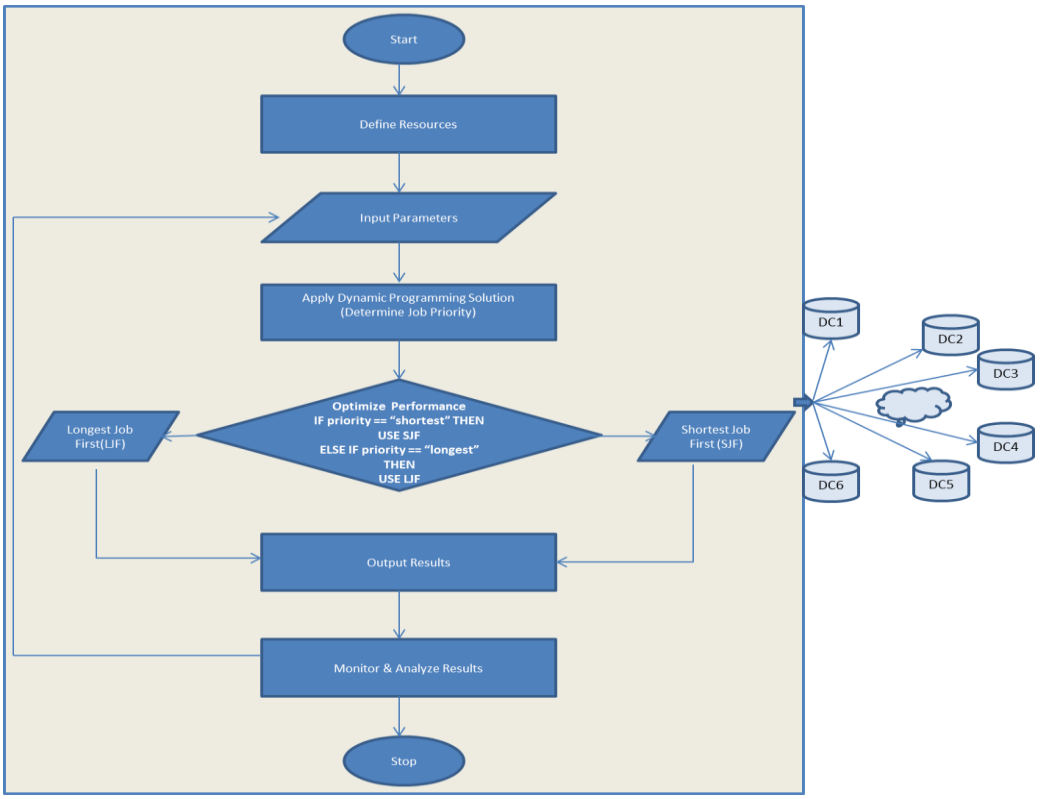

Figure 1: Optimization approach based on Twin Dynamic Programming algorithm with Longest and Shortest Job First broker policy

2.4.1 Longest Job First data centre broker policy

The Dynamic programming strategy is used for the optimization process of load balancing and management of the resource utilization for the processing of computational intensive tasks (cloudlets), the Longest Job First broker policy is added to the Dynamic programming algorithm. The aim is to select and process longer and more complex tasks first. The policy performs sorting tasks in descending order of their pre-assigned instruction length and enabled their submission to VMs specified by the dynamic programming procedure. Memory and CPU resources are primarily assigned to tasks having a greater length. The steps of the algorithm for achieving better system utilization for increased system loads based on the dynamic programming algorithm for task allocation with the implemented Longest Job First broker policy are shown in Algorithm I.

ALGORITHM I: Pseudocode of Dynamic Programming algorithm with Longest Job First broker policy.

INPUT:

jobs : a list of tasks with their processing time
n : number of tasks

OUTPUT:

schedule: a list of tasks in the order they should be executed

ALGORITHM:

1. begin
2. dp **= a 2D table of size** (n+1) * (n+1) initialized with 0s
3. schedule = an empty list
4. for i from 1 to n:

- for j from 1 to n:
- if jobs [i] is compatible with jobs [j] (e.g no conflicts**)**:

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

- dp[i] [j] = max (dp [i-1] [j-1] + processing _time (jobs [i]),
dp [i - 1] [j])
else:
dp [i] [j] = dp [i-1] [j]

5. i = n, j = n
6. while i > 0 and j > 0:

- if jobs [i] is compatible with jobs [j]**:**
- schedule .**a**ppend (jobs [i]**)**
- i -= 1, j -= 1
else
i -=1

7. return schedule
8. end

Note:
processing_time (jobs[i]) returns the processing time of task **i**
compatible checks if task **i** and **j** can be executed together (e.g no resource conflicts)

This algorithm used DP to build a table **dp** that stores the maximum processing time that can be achieved by executing tasks up to **i** and **j** . The **schedule** is constructed by tracing back the optimal path in the **dp** table.

2.4.2 Shortest Job First data centre broker policy

The Dynamic programming strategy with Shortest Job First scheduling algorithm is applied to batch-type processing of computationally intensive tasks. The algorithm places tasks into a queue based on their arrival time and provide a fast-scheduling dynamic programming Shortest Job First algorithm where the tasks with the shortest execution time are given priority to be executed first. This minimize the waiting time for the execution of task (cloudlets). With the dynamic programming strategy with Shortest Job First, the waiting time for the longer tasks is shorter than the waiting time for shorter tasks in the case of dynamic programming strategy with Longest Job First. The algorithm of the dynamic programming algorithm for task allocation with the implemented shortest Job First broker policy is shown in Algorithm II.

ALGORITHM II: Pseudocode of Dynamic Programming algorithm with shortest Job First broker policy.

INPUT:
jobs : a list of tasks with their processing time
n : **the** number of tasks

OUTPUT:
schedule: a list of tasks in the order they should be executed

ALGORITHM:

1. begin
2. dp **= a 2D table of siz**e (n+1) * (n+1) initialized with 0s
3. schedule = an empty list
4. for i from 1 to n:

- for j from 1 to n:
- if jobs [i] is compatible with jobs [j] (e.g no conflicts**)**:
- dp[i] [j] = **min** (dp [i-1] [j-1] + processing _time (jobs [i]),
dp [i - 1] [j])
else:
dp [i] [j] = dp [i-1] [j]

5. i = n, j = n

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

6. while i > 0 and j > 0:
- if jobs [i] is compatible with jobs [j]**:**
- schedule .**a**ppend (jobs [i]**)**
- i -= 1, j -= 1
else
i -=1
7. return schedule
8. **end**

Note:
processing_time (jobs[i]) returns the processing time of task **i**
compatible checks if task **i** and **j** can be executed together (e.g no resource conflicts)
This algorithm used Dynamic Programming to build a table **dp** that stores the maximum processing time that can be achieved by executing tasks up to **i** and **j** . The **schedule** is constructed by tracing back the optimal path in the **dp** table.

The key difference from the longest job first algorithm is the use of min and max in the dynamic programming table which prioritizes either of both actions.
The .**append** used in the algorithms is a method used to add an element to the end of list in python. Therefore the **.append** as used in the algorithms is used to add tasks to the schedule list in order they should be executed.

2.5 Simulation Framework /Procedure

The simulation framework for the cloud based environment typically involves finite stages of conducting the simulation. The Model flow began with the activation of simulation process accordingly: Start, Initialization, Resource provisioning, Workload submission, Simulation execution, monitoring and Analysis, Output and visualization, termination and the End process. Figure 2 illustrates the simulation framework for the simulation.

Figure 2: Simulation framework

i. **Initialization of CloudSim**: The simulation is initialized with the definition of the cloud simulation parameters (e.g simulation Time and warm up period). And set up the cloud infrastructure (number of Data centers, hosts, and virtual machines (VMs). The user also defined the workload and application requirements.
ii. **Resource Provisioning**: At this stage, the simulator provisions the required resources (e.g VMs, storage, network bandwidth) based on user request. The resources are allocated to the appropriate data centers and hosts.
iii. **Workload Submission**: At this stage the user submits the workload (e.g applications, tasks to the simulator. The workload is further into smaller tasks and assigned to the provisioned resources.
iv. **Simulator Execution:** At this stage, the simulator executes the workload tasks on the provisioned resources. The simulator models the behavior of the cloud infrastructure, including

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

resources utilization, network traffic, and energy consumption.

v. **Monitoring and Analysis:** The simulator collects performance metrics and statistics during the simulation. The user then analyzes the results to evaluate the performance, scalability, and energy efficiency of the cloud infrastructure.

vi. **Output and Visualizations:** At this stage, the simulator generates output files and visualizations in form of graphs, charts etc to help user understand the simulation results.

vii. **Termination:** At this stage the simulation is terminated when the workload is completed or a specified time limit is reached.

2.6 Evaluation Metrics

In the experiment, the Execution Time, Throughput, Average Resource Utilization (ARU), Scalability, Cost efficiency, Quality of Service (QoS), Makespan (Chen et al., 2023), were used to evaluate and have been considered for improving the performance of the proposed simulation algorithm. The optimization goals of Dynamic programming with longest Job First and Shortest Job First-based algorithms are to reduce average execution time, maximize resource utilization, increase throughput, minimize makespan, reduce cost of processing power and execute all tasks assigned to a multi-tiered Cloud environment with respect to VMs. The central aim is achieving resource scalability, and efficiency with minimal cost.

3.0 RESULTS AND DISCUSSION

3.1 Results

This section presents the results of the experimental implementation of the optimization algorithms based on the principles of the twin-Dynamic programming principle. The dynamic programming strategy is used against a traditional method (First-Come-First-Serve (FCFS)) and discussed for task scheduling and balancing in a multi-tiered Cloud environment. The Two Dynamic Programming algorithms are used to optimize Quality of Service (QoS): Longest Job First (DP-LJF), and dynamic programming Shortest Job First (DP-SJF) strategies. The simulation tests the reliability of the two DP algorithms and the FCFS algorithm based on time required to deliver cloud service task. Experimental results of the same metrics obtained from the strategies are compared and analyzed based on the performance metrics enlisted in the methodology section.

The optimization goals of Dynamic programming with longest Job First and Shortest Job First-based algorithms are to reduce average execution time, maximize resource utilization, increase throughput, minimize makespan, reduce cost of processing power and execute all tasks assigned to a multi-tiered Cloud environment with respect to VMs. The central aim is achieving resource scalability with minimal cost.

The Simulation was tested with different numbers of tasks in an increasing order (10, 100, 200, 500, and 1000 tasks) from the dataset. The increasing order ensures complexity is achieved by testing the performance with different loads. Each process of the algorithm is evaluated several times, while the average results are analyzed. The Dynamic programming strategies algorithm considers the task characteristics and matches task characteristics to the most appropriate VM instance. The validation is performed for each algorithm and the results were plotted as shown in the following Figures.

I. Average Execution Time

The task scheduling algorithms were primarily aim to reduce the total execution time for the computational task assigned. Such Tasks are executed simultaneously on VM resources as selected by the service broker/policy. Figure 3 shows the measurement results, which demonstrate a high difference in the average cloudlet execution time between the Dynamic programming (DP-LJF and DP-SJF) and the First-Come-First-Serve (FCFS) algorithms approaches. It shows how the average execution time changes with the increasing number of tasks (cloudlets). Thus, the execution time of the Longest Job First strategy was slightly higher than the Shortest Job first strategy due to the waiting time for the allocated VM resource. This performance characteristic is related to data from the workloads.

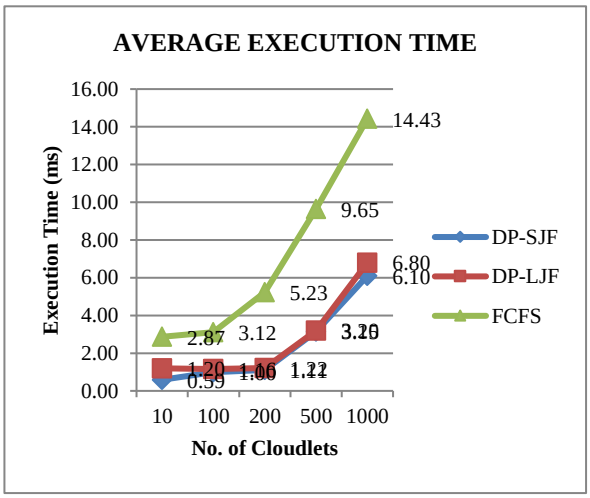

Figure 3: Average Execution Time

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

II. Average Resource Utilization

The average resource utilization is an important performance indicator of optimal management across complex Cloud infrastructure intended for supporting scientific research. Figure 4 reveals that the value of resource utilization rate increases with the increase in the number of tasks (Cloudlets). Meanwhile there is huge difference in the average resource utilization rate between the FCFS and the DP (LJF & SJF) algorithms approaches. It shows how the average resource utilization rate changes with the increasing number of tasks (cloudlets). Thus, the average resource utilization rate of DP (LJF & SJF), were higher than the First-Come-First-Serve strategy. The Shortest Job First approach tends to outperform other approach in utilizing available resources. Thus, the average resource utilization rate shown in the graph corresponds to the simulation resource usage related to data from the workload.

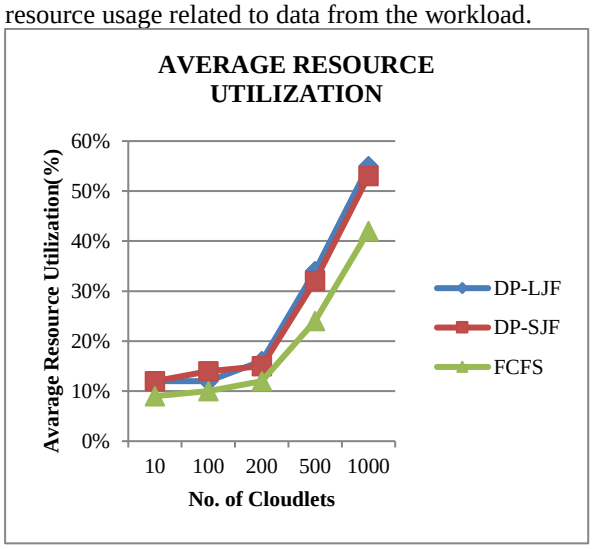

Figure 4: Average Resource Utilization (ARU)

III. Throughput

Throughput is one of the main metric for assessing the productivity of task processing that consequently affects QoS, as well as a factor for analyzing scalability properties under different loads. The results of the two compared approaches are shown in Figure 5. The results show that the DP (SJF & LJF) significantly outweighs the FCFS approach, which indicates improved throughput in all test scenarios. Thus, the Shortest Job slightly outweighs the DP-First longest Job First approach. This signifies the applied approach results in increased throughput as more tasks are processed in less time and consequently has greater average resource utilization.

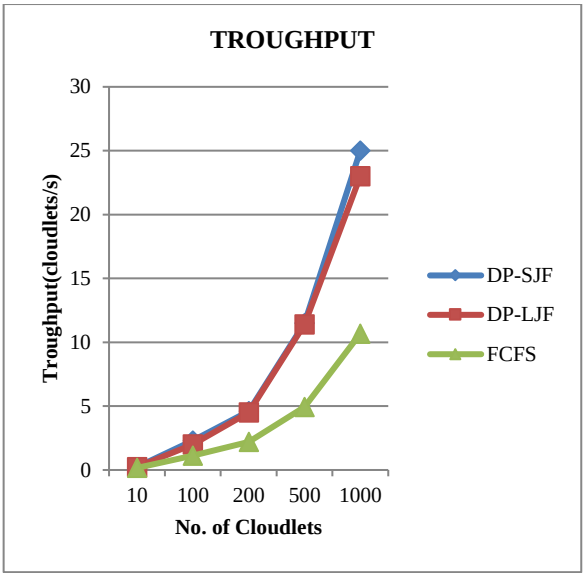

Figure 5: Throughput

IV. Makespan

Reducing the makespan depicts better performance. The graph in Figure 6 shows the makespan values of the DP (LJF & SJF) and the FCFS algorithms. It reveals a significant time difference between the start and finish times of the task list, expected to be minimized. Therefore, the DP (Longest Job First and Shortest Job First) approach shows the lowest makespan as compared to the FCFS approach. Thus, there is a slight difference between the DP approaches. The values show the expected increase in the overall makespan as the number of cloudlets ratio increases.

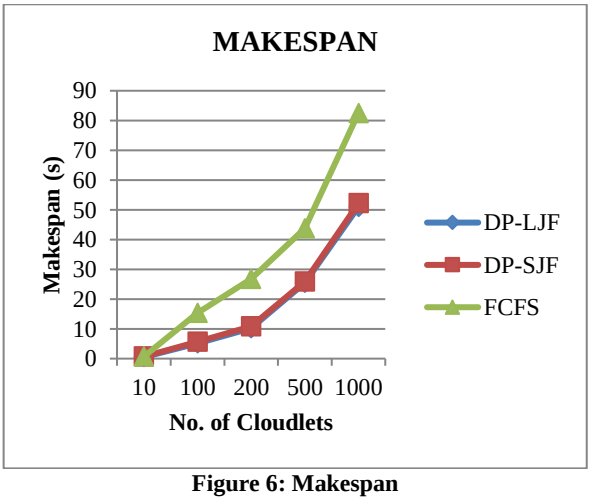

Figure 6: Makespan

V. Cost Analysis

The cost of cloud service in the African region appears expensive; therefore it is paramount to

DOI: **https://doi.org/10.57233/ijsgs.v11i4.992** ISSNp: 2488-9229; ISSNe: 3027-1118

IJSGS FUGUSAU VOL. 11(4) **WEBSITE: https://fugus-**

reduce the cost of service charge on startups due to initial financial and market limitations. The study employs the use of scalable (DP-SJF and DP-SJF) virtual machine cost and compares with non-scalable virtual machine service cost. The non-scalable virtual machine cost is very high to new entrant with limited resources. The proposed scalable techniques minimize the cost of virtual machine usage, and hence reduce the cloud consumer service charge based on the expected number of target consumer or the size of the business. The optimize result as shown (Figure 7) reduces cloud consumers cost of service charged by the service provider by 20% as compared to non-scalable Virtual machine cost.

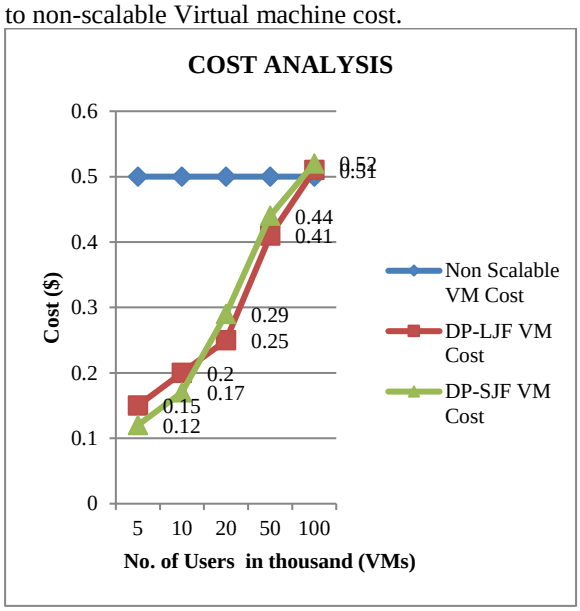

Figure 7: Cost Analysis

3.2 DISCUSSION

The proposed simulation based approach was used to evaluate the optimization problem in the Cloud environment based on dynamic programming. The Dynamic programming strategy is used for the optimization process of load balancing and management of the resource utilization for the processing of computational intensive tasks (cloudlets). Longest Job First broker policy and Shortest Job First broker policy were incorporated into the Dynamic programming algorithm with the aim of optimizing the overall efficiency and effectiveness of cloud based complex tasks. In this simulation experiment, the Dynamic programming strategy algorithms (DP-LJF & DP-SJF) were tested against the traditional First Come First Serve (FCFS) strategy. This approach for the allocation of resources needed to process tasks has to find a near-optimal solution within a reasonable computation time. These tasks are; average execution time, average resource utilization, throughput, makespan, reduced cost and scalability.

The optimization goals of dynamic algorithms have been achieved. The results obtained using DP-LJF and DP-SJF are better compared with results obtained using the traditional FCFS algorithm under the same conditions. The Dynamic Programming-based approaches achieved a smaller execution time, smaller makespan, higher resource utilisation, increased throughput, similar average execution time, reduced cost and maintain scalabilty. In the Dynamic Programming with Longest-Job-First algorithm strategy, the waiting time is higher as longer tasks have to be executed first. As such the tasks with smaller instruction lengths will be waiting for free resources. The Dynamic Programming with Shortest-Job-First algorithm minimises the makespan, increases throughput, and efficiently uses VM resources in most cases. It can be deduced from the analysis of resource utilisation that the number of

data centres and the number of VM instances dynamically follow the number of tasks to be executed.

From the analysis of the task size distribution, it is evident that the characteristics and number of tasks, along with the number of VMs, may affect its completion. The use of scalable (DP-SJF and DP-SJF) virtual machine cost as compared with non-scalable virtual machine service cost proves to provide a reduction in the cost of cloud services charged. It is evident that scaling is achieved through upgrading more resources by increasing the number of data centers to handle as well as adding the number of resource (VMs) to the host data center to handle resources workloads. The scalability technique minimizes the cost of virtual machine usage, and hence reduces the cloud consumer service charge based on the expected number of target consumer or the size of the business. Therefore, it is evident from the data that that the choice of Datacentre variations, and dynamic Programming strategy is of great importance for achieving scalability in a Cloud Ecosystem based on heterogeneous resources, because it will balance the task pattern, number of users and size of workload. As a result, optimality, efficiency and scalability of using cloud ecosystem resources are ensured.

4.0 CONCLUSION

This work proposed a dynamic programming approach using a simulation platform to optimize resource utilization in cloud computing environments Optimizing resource management in cloud environments is one of the biggest challenges facing organizations, especially when dealing with large and heterogeneous data that require intensive and dynamic allocation of computing resources. This produced results that were highly accurate and acceptable, with optimal success rate over other algorithms. The findings of the optimization of resource allocation results have shown remarkable improvement over Evolution based Strategies Algorithm (Loncar & Loncar, (2022). The findings also demonstrated a scalable and cost effective algorithm over Cost-Effective Energy Efficient scheduling algorithm (Vashisht & Kumar, 2022). In conclusion, performance of the algorithm is not a constant value, and also the relative performance for different algorithms as they change related to the changed values of different parameters (Ahmad & Hammo, 2022; Zhu, 2025).

Therefore, this study focused on the accuracy of the simulation in identifying the right-driven point of resource allocation that enhance efficiency and reduce computational/ operational costs. The findings of this study explore recommendations that would help to enhance strategies on resource allocations in cloud environments. This research demonstrates an essential development for simulative technology to be used as an effective tool in proactive and premeditated cloud resource management.

REFERENCES

Ahmad, A. Y. & Hammo, A.Y. (2022). A Comparative Study of the Performance of Load Balancing Algorithms Using Cloud Analyst. Webology, 19 (1), 489-491. doi: 10.14704/WEB/V19I1/WEB19328

Aldossary, M. (2021). A Review of Dynamic Resource Management in Cloud Computing Environments. Computer Systems Science & Engineering, 36(3).

Ashish G., & Ritu, G. (2017). Load Balancing Based Task Scheduling with ACO in Cloud Computing. Paper presented at the International Conference on Computer and Applications.

Chen, W., Gao, S., Chen, W., & Du, J.(2023). Optimizing resource allocation in service systems via simulation: A Bayesian formulation. Production and Operations Management, 32(1), 65-81.

Frederic, N. & Yang, Ya. (2017) Efficient Resource Management techniques in Cloud Computing Environment: Review and discussion, TELKOMNIKA. 15(4), 1917-1933 ISSN: 1693-6930,

Ghedass, F. & Ben, F C. (2021). A multi-view learning approach for the autonomic management of big services, in Web Information Systems Engineering – WISE 2021,

Gonzalez, N. M., Carvalho, B & Miers, C. C. (2017). Cloud resource management: towards efficient execution of large-scale scientific applications and workflows on complex infrastructures, Journal of Cloud Computing, 6(1), 1–20.

Farootani, A., Lervolino, R., Tipaldi, M., & Neilson, J.(2020). Approximate dynamic programming for stochastic resource allocation problems. IEEE/CAA Journal of Automatica Sinica, 7(4), 975-990.

Farootani, A., Tipaldi, M., Ghaniee, Z., Liuzza, D., & Glielmo, L.(2021). Modeling and Solving resource allocation problems via a dynamic programming approach. International Journal of Control, 94(6), 1544-1555.

Hu Yao, X. F., Honghui Li, Gaifang Dong, & Jianrong Li. (2017). Cloud Task Scheduling Algorithm based on Improved Genetic Algorithm. International Journal of Performability Engineering, 13, 1070-1076. doi:10.23940/ijpe.17.07.p9.10701076

Llwaah, F. (2024) Resource Utilization Performance of Complex Workflows on the Public Cloud: A Simulation-Based Approach, International Journal of Computing and Digital Systems, 17(1), 1–10.

http://dx.doi.org/10.12785/ijcds/XXXXXX

Loncar, P. & Loncar, P. (2022). Scalable Management of Heterogeneous Cloud Resources Based on Evolution Strategies Algorithm. IEEE Access Journal, 10 (1), 68778-68791. doi: 10.1109/ACCESS.2022.3185987

Mahmood, I. et al., (2021). Task Scheduling Algorithms in Cloud Computing: A Review, Turkish Journal of Computer and Mathematics Education (TURCOMAT). 12(4), 1041-1053. DOI: 10.17762/turcomat.v12i4.612

Mitropoulou, K., Kokkinos, P., Soumplis, P. & Varvarigos. E.(2023). Anomally detection in Cloud Computing using Knowledge Graph Embedding and Machine Learning Mechanisms, Journal of Grid Computing, 22(6). https://doi.org/10.1007/s10723-023-09727-1

Moazeni, A. Khorsand, R., & Ramezanpour, M.(2022).Dynamic resource allocation using an adaptive multi-objective teaching-learning based optimization algorithm in cloud. IEEE Access, 11, 23407-23419.

Mukherjee, D. D, & Buyya, R.(2024). Cloud Computing Resource Management, Springer Nature Singapore, 17–37. [Online]. Available: https://doi.org/10.1007/ 978-981-97-2644-8 2

Pradeep S. R, Priti D. & Gyanendra P., S (2018). Virtual machine allocation to the task using an optimization method in cloud computing environment. International journal of Information Technology.12(1-2), 1-9. https://doi.org/10.1007/s41870-018-0242-9

Ruonan Lin, Q. L. (2016). Task Scheduling Algorithm Based on Pre-Allocation Strategy in Cloud Computing. Paper presented at the International Conference on Cloud Computing and Big Data Analysis.

Saleh, M. & Abel, M. (2015). Information Systems: Towards a system of Information systems. In proceedings of the 7th International joint conference of Knowledge discovery, Knowledge Engineering, knowledge Management (IC3K)-KMIS, 193-200, SciTePress,. doi:10.5220/0005596101930200

Simi´c, S.D, Tankovi´c, N. & Etinger, D. (2023). Big data bpmn workflow resource optimization in the cloud, Parallel Computing, 117, 103025. [Online]. Available: https://www.sciencedirect. com/science/article/pii/S0167819123000315

Souza, E. A. G. de, Nagano, M. S., & Rolim, G. A. (2022). Dynamic Programming algorithms and their applications in machine scheduling: a review, Expert Systems with Applications, 190, 1-35. doi:10.1016/j.eswa.2021.116180

Sumalatha, k. & Anbarasi, M.S (2019). A review on various optimization techniques of resource provisioning in cloud computing. International Journal of Electrical and Computer Engineering (IJECE), 9(1), 629-634. doi: 10.11591/ijece.v9i1.pp629-634

Trimarchi, S., Cassamatta, F., Gamba, L., Grimaccia, F., Lorenzo, M., & Niccolai, A. (2025). A review of Agent-based Models for Energy Commodity Markets and their Natural Integration with Rl Models. Energies 18(12): 3171

Tuli, S., Gill, S.S., Xu, M., Garraghan, P., Bahsoon, R., Dustdar, S., & Jennings, N.R. (2022). HUNTER: AI based Holistic resource management for sustainable cloud computing. Journal of Systems and Software, 184:11124

Vashisht, P. & Kumar, V. (2022). A Cost Effective and Energy Efficient Algorithm for Cloud Computing, International Journal of Mathematical, Engineering and Management Sciences 7(5), 681-696. doi.org/10.33889/IJMEMS.2022.7.5.045

WANG Bei, L. J. (2016). Load Balancing Task Scheduling based on Multi-Population Genetic Algorithm in Cloud Computing. Paper presented at the Proceedings of the 35th Chinese Control Conference, Chengdu, China.

Zhang, W., Zou, L., Maamar, Z. & Chen, L. (2021) Eds. Cham: Springer International Publishing, 2021, 463–479.

Zhifeng, K. C., Xiaojun, Z, & Shuange, Z. (2016). Virtual Machine-Based Task Scheduling Algorithm in a Cloud Computing Environment. Tsinghua Science And Technology, 21, 660-667.

Zhu, Q. (2025). Autonomous Cloud Resource Management through DBSCAN-based unsupervised learning. Optimizations in Applied Machine Learning.