# CLOUD PROVISIONING USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK

**Bryan Yehuda Mannuel[1)], Henning Titi Ciptaningtyas[2)], and Ary Mazharuddin Shiddiqi[3)]**
[1, 2)] Department of Information Technology, Faculty of Electrical and Intelligent Information Technology, Sepuluh Nopember Institute of Technology (ITS)
[3)]Department of Informatics, Faculty of Electrical and Intelligent Information Technology, Sepuluh Nopember Institute of Technology (ITS)
e-mail: bryanyehuda@gmail.com[1)], henning@if.its.ac.id[2)], ary.shiddiqi@if.its.ac.id[3)]

**ABSTRACT**

*This research aimed to build a resource management system in cloud computing to maximize resource utilization by implementing a genetic algorithm and an artificial neural network. The research was conducted in two scenarios: one using only the genetic algorithm and the other using both the genetic algorithm and the artificial neural network. The results showed that the implementation of a genetic algorithm alone can increase resource utilization by 48%-60%, and the implementation of both the genetic algorithm and the artificial neural network can increase resource utilization by 38%-59%. The genetic algorithm alone is effective in reducing energy use and execution time, while the combination of the two is useful for quickly handling tasks with large data imbalances in cloud provisioning.*

***Kata Kunci**: Artificial Neural Network, Cloud Computing, Genetic Algorithm, Task Scheduling, Virtual Machine*

# CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK

**Bryan Yehuda Mannuel[1)], Henning Titi Ciptaningtyas[2)], and Ary Mazharuddin Shiddiqi[3)]**
[1, 2)] Departemen Teknologi Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS)
[3)]Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: bryanyehuda@gmail.com[1)], henning@if.its.ac.id[2)], ary.shiddiqi@if.its.ac.id[3)]

**ABSTRAK**

*Penelitian ini bertujuan untuk membangun sistem manajemen sumber daya pada komputasi awan untuk memaksimalkan pemanfaatan sumber daya dengan mengimplementasikan genetic algorithm dan artificial neural network. Penelitian dilakukan dalam dua skenario: satu hanya menggunakan genetic algorithm dan yang lainnya menggunakan genetic algorithm dan artificial neural network. Hasil penelitian menunjukkan bahwa penerapan genetic algorithm saja dapat meningkatkan pemanfaatan sumber daya sebesar 48%-60%, dan penerapan genetic algorithm dan artificial neural network dapat meningkatkan pemanfaatan sumber daya sebesar 38%-59%. genetic algorithm saja efektif dalam mengurangi penggunaan energi dan waktu eksekusi, sedangkan kombinasi keduanya berguna untuk menangani tugas dengan cepat dengan ketidakseimbangan data yang besar dalam penyediaan cloud.*

**Kata Kunci**: *Artificial Neural Network, Cloud Computing, Genetic Algorithm, Mesin Virtual, Penjadwalan Tugas*

## I. INTRODUCTION

In the modern era where the use of technology is increasing rapidly and increasing rapidly, the use of Cloud Computing is increasingly in demand [1]. Cloud Computing is the availability of computer system resources on demand, such as data storage and computing power, without direct management by the user [2]. However, recent research states that the level of usage of Cloud resources in many data centers is still quite low. This is caused because there are still many Cloud Service Providers who do not use the virtualization capabilities possessed by Cloud Computing to the fullest, resulting in wasted energy and effort [3]. Therefore, a good resource management system is needed for a Cloud Service Provider so that their Cloud Computing system can take full advantage of resource virtualization capabilities and increase the level of use of Cloud resources.

Cloud provisioning is a key feature of the Cloud Computing system, which relates to how customers obtain Cloud resources from Cloud Service Providers [4]. Task scheduling and virtual machine (VM) allocation play an important role in cloud provisioning. This is because the Cloud Computing system relies on virtualization technology which allows resources from one physical Cloud resource to be divided into several isolated environments

running on virtual machines (VMs) [5].

The biggest challenge in building a task scheduling and virtual machine (VM) allocation system in Cloud Computing is finding an algorithm that can maximize the use of Cloud resources. This challenge is commonly referred to as the "Knapsack Problem" where "Given a set of objects, each with a certain weight and value, then determine the number of each object to be included in the collection so that the total weight is less than or equal to the limit given and the total value is as large as possible" [6]. This challenge often arises in resource allocation where decision-makers must choose from a series of indivisible tasks under a fixed budget or time constraint [7].

To be able to overcome this, research was carried out using the Genetic Algorithm which was inspired by the natural selection process and the implementation of Artificial Neural Networks which are based on biological neural networks that form the brain to build a task-scheduling system and virtual machine (VM) allocation to maximize resource use. cloud. Genetic Algorithm is used to produce high-quality solutions for optimizing the use of Cloud resources by relying on biologically inspired operators such as mutation, crossover, and selection [8]. Added with the implementation of Artificial Neural Networks to learn, process, and predict the results of optimization solutions [9].

## II. Related Works

Farouk et al. [5] discusses the challenges of building a task scheduling system and virtual machine (VM) allocation in a Cloud Computing system due to heterogeneity in the system. The journal suggests using a modified Genetic Algorithm to address multiple achievement targets, including maximizing Resource Utilization, Load Balancing, and Power Management. The modified Genetic Algorithm uses a matrix structure to represent chromosomes and was found to outperform other algorithms in terms of Makespan, Scheduling Length, Throughput, Resource Utilization, Energy Consumption, and Balance Degree. The author references this research and proposes using a Genetic Algorithm assisted by an Artificial Neural Network to see if even better results can be achieved. The use of ID Tasks, ID VM, and ID Data Center as chromosomal representations is also referenced.

Pradeep et al. [10] studies the efficiency of various task scheduling algorithms in a Cloud Computing system, including BB-BC, GA-Cost, GA-Exe, and GA-ANN (Genetic Algorithm – Artificial Neural Network). The study compares these algorithms using various parameters, such as Average Start Time, Average Finish Time, Average Execution Time, and Wait Time. The research also includes control variables, such as the number of tasks, virtual machines, and data centers, as well as bandwidth, CPU, and RAM. The study used two data sources, self-generated data and the San Diego Supercomputer Center's Blue Horizon logs dataset. The results showed that the GA-ANN algorithm was the most efficient, with an 82.63% reduction in error rate, 26.81% increase in successfully completed tasks, 10.66% reduction in execution time, and 69.94% reduction in wait time. The authors plan to further study the effectiveness of the GA-ANN algorithm compared to a Genetic Algorithm alone.

Michael et al. [11] discusses the energy efficiency of the current Cloud Computing system. It was found that the Utilization Rate, which measures the percentage efficiency of using the system, was only 30-42%. The low Utilization Rate was attributed to a lack of proper utilization of virtualization capabilities by Cloud Service Providers, leading to wasted energy and manpower. The paper calls for immediate action to be taken by Cloud Service Providers to solve this issue and suggests the creation of a task scheduling system and VM allocation to increase the Utilization Rate.

Henning et al. [12]presents methods for evaluating the performance of task scheduling and virtual machine allocation systems, including definitions and formulas for measures such as Makespan, Throughput, and Resource Utilization. The authors of this research use this journal as a reference in their study.

By using Genetic Algorithms and Artificial Neural Networks to solve the "Knapsack Problem" in Cloud Provisioning, we aim to build a system that can effectively schedule tasks and allocate virtual machines in a way that maximizes the use of Cloud resources. This not only prevents performance degradation, but also increases the usage rate of Cloud resources and reduces execution time and wasted energy and effort. Overall, this research aims to create a more efficient and effective way of utilizing Cloud resources.

## III. Proposed Stages of Research

The following are the stages of the method that will be used by the author in this research. This section is important to understand what are the stages and processes that will be carried out by researchers in carrying out this research and getting the results.

### *A. Dataset Used*

In this study, two scenarios will be carried out where for the first scenario, task scheduling and virtual machine (VM) allocation will be carried out using only the Genetic Algorithm, and for the second scenario, it will be carried out using the Genetic Algorithm together with an Artificial Neural Network. Both of these scenarios will use two datasets as shown in Table 1 as data sources for the simulation of task scheduling and virtual machine (VM) allocation scenarios. Later these two scenarios will be compared to find out which scenario is more efficient. Before that, we need to do some preprocessing on the dataset that will be used. The results of preprocessing on this dataset will produce a clean dataset and have the appropriate data format so that it is ready for use in research.

TABLE 1
DATA SOURCES.

| No | Scenario Name | First Dataset | Second Dataset |
|---|---|---|---|
| **1** | Genetic Algorithm | Randomized Dataset | Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [13]. |
| **2** | Genetic Algorithm + Artificial Neural Network | Randomized Dataset | Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [13]. |

### *B. First Scenario: Using Genetic Algorithm Only*

For the first scenario, research will be carried out using only the Genetic Algorithm. First of all, initial population initialization of the schedule will be carried out as a guideline for allocating Tasks to appropriate virtual machines (VMs) using only the Genetic Algorithm based on the dataset used. The specifications for the Genetic Algorithm to be used in this research can be seen in Table 2.

TABLE 2
GENETIC ALGORITHM SPECIFICATIONS.

| Dataset | Population Size | Mutation Rate | Crossover Rate | Elitism Count | Loop Number |
|---|---|---|---|---|---|
| Randomized Dataset | 20 | 30% | 95% | 2 | 20 |
| SDSC Dataset | 20 | 30% | 95% | 2 | 20 |

This specification will affect how the Genetic Algorithm will produce an efficient schedule solution. Population size is the number of individuals in one population in one generation in the Genetic Algorithm [8]. The mutation rate is the probability that a gene on a chromosome can randomly change into another gene [8]. The crossover rate is the probability that two parents from the previous generation will have their genes crossed into a new individual in the next generation that has a combination of genes from both parents [8]. Meanwhile, the elitism count is the number of individuals who have high fitness function values from each generation who will be included directly as members of the next generation [8].

Chromosomes of each individual in the population will contain the number 0 (zero) up to 8 (eight). This numbers will be randomly filled in and duplication is allowed. These numbers represent the ID of the Virtual Machine that will be the place where Task will be processed.

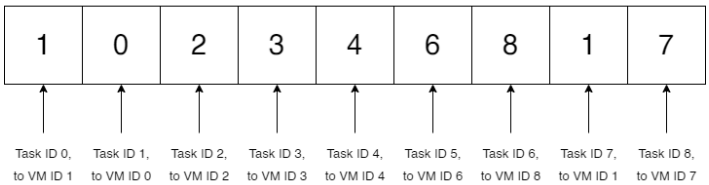


Fig. 1. Chromosome Representation

This schedule will be in the form of a chromosomal representation of an individual in the Genetic Algorithm. This chromosome has a length of 9 and contains the ID of the virtual machine (VM) that will be the place where the Task will be processed. This chromosome representation can be seen in Figure 1.

This schedule will be calculated in accordance to the Fitness Function to find out whether they are efficient enough in scheduling tasks and allocating virtual machines (VMs) or not. The Fitness Function can be seen in Equation 1 to Equation 3.

Here Total Execution Time means the total time spent by the Task in execution and the smaller the value, the better the solution. Failure Rate means the number of Tasks that fail to be processed per unit of time and the smaller the value, the better the solution. $\alpha$ and $\beta$ are the percentage of how important the Total Execution Time

value and the Failure Rate value are in influencing the quality of the solution and the total value of both must be equal to 1. The formula for Total Execution Time is the total sum of all Task lengths divided by the MIPS (Million Instructions Per Second) of the virtual machine (VM) where it will be processed. While the Failure Rate formula is the number of Tasks that fail to be processed divided by the simulation time.

$$Fitness\ Function = \alpha * \frac{1}{Total\ Execution\ Time} + \beta * \frac{1}{Failure\ Rate} \quad (1)$$

$$Total\ Execution\ Time = \sum_{task_i=1}^{n} \frac{TaskLength_i}{MIPS_j} \quad (2)$$

$$Failure\ Rate = \frac{Task\ Failed\ Count}{\Delta T} \quad (3)$$

These schedules will then be selected, mutated, and crossed using Genetic Algorithm to produce a new generation that is more efficient than the previous generation when calculated from its Fitness Function. When it is felt that there is no significant increase in the Fitness Function or the termination condition has been reached, the Genetic Algorithm will be stopped and the schedule that has the highest Fitness Function will be the output of this first scenario. Tasks can be allocated to virtual machines (VMs) based on this schedule and parameter assessments can be obtained from the simulation results. For more details regarding the first implementation scenario, see Figure 2.

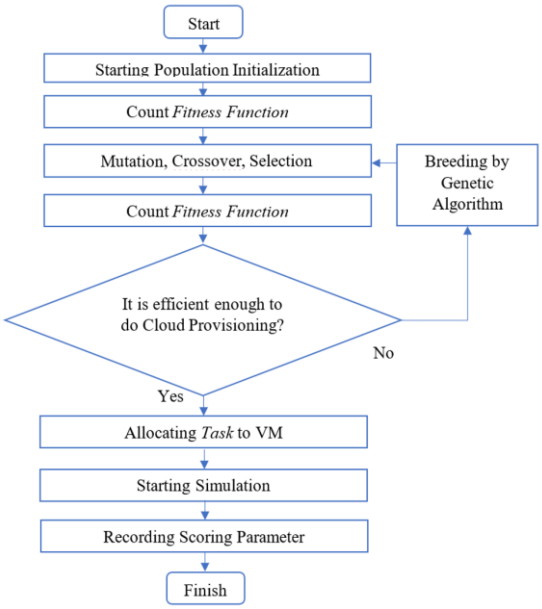


Fig. 2. First Scenario Implementation Flowchart

### C. *Second Scenario: Genetic Algorithm and Artificial Neural Network*

For the second scenario, a combination of Genetic Algorithm and Artificial Neural Network will be carried out. First of all, the results of the schedule output from the first scenario will be divided into two, namely the learning dataset of 80% and the testing dataset of 20% [5]. The learning dataset will be analyzed and studied first by the Artificial Neural Network to produce a scheduling model that has appropriate accuracy based on the schedule output from the Genetic Algorithm. The specifications of the Artificial Neural Network for this research can be seen in Table 3 below.

TABLE 3
ARTIFICIAL NEURAL NETWORK SPECIFICATIONS.

| Dataset | Input Node | Hidden Node | Output Node | Propagation Method | Epoch Number | Accuracy | Input | Output | Hidden Node Activation Function | Output Node Activation Function |
|---|---|---|---|---|---|---|---|---|---|---|
| Randomized Dataset | 9 | 18 | 9 | Manhattan | 100000 | 88% | 9 Task Lengths | 9 VM IDs | ReLu | Sigmoid |
| SDSC Dataset | 9 | 18 | 9 | Manhattan | 100000 | 88% | 9 Task Lengths | 9 VM IDs | ReLu | Sigmoid |

The 9 Input Node specifications are useful for receiving input of 9 Task lengths which will later become the initial information for the Artificial Neural Network to output as many as 9 VM IDs on its Output Node. Between the Input Node and Output Node, there will be 1 Hidden Node containing 18 Node as an intermediary between the two. This Hidden Node will use the ReLu Activation Function and the Output Node will use the Sigmoid Activation Function [14].

Manhattan propagation is used in this Artificial Neural Network so that the author can directly control how much weight is reduced in the Artificial Neural Network so the author can get the appropriate level of accuracy. This is because Manhattan Propagation uses a very small value to reduce or add weight, unlike the usual propagation methods that use Gradient so that the change in value on the weight is too large and the resulting accuracy cannot be controlled by the author [14]. The accuracy that the author aims for in this research is 88% because if it is too high it will result in an Overfit and if it is too small it will result in an Underfit of the model.

After the learning dataset has been analyzed and studied by the Artificial Neural Network, a model for predicting the schedule will be generated. This model will be tested using the test dataset and the results of the resulting schedule will be entered into the simulation in CloudSim. Tasks can be allocated to virtual machines (VMs) based on this schedule and parameter assessments can be obtained from the simulation results. For more details regarding the second implementation scenario, see Figure 3.

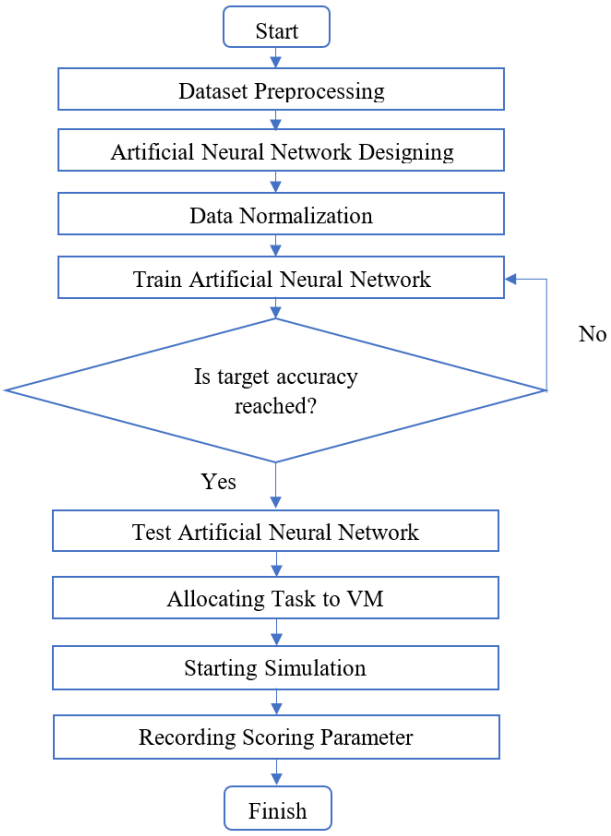


Fig. 3. Second Scenario Implementation Flowchart

### *D. Implementation Environment*

The implementation of the scenario will use CloudSim. An Open-Source framework, which is used to simulate Cloud Computing services. The scenarios will be implemented on 54 Virtual Machines in 18 Hosts, which are in 6 Data Centers. Each of these Data Centers will be connected to a Data Center Broker which functions as the brain for task scheduling and virtual machine (VM) allocation. This Data Center Broker will be connected to the VM List as a list of existing Virtual Machines and their status, to the Task List as a list of existing Tasks and their status, and finally to the user as the entity that assigns Tasks and receives the Processed Output. This implementation scheme can be seen in Figure 4 and Figure 5.

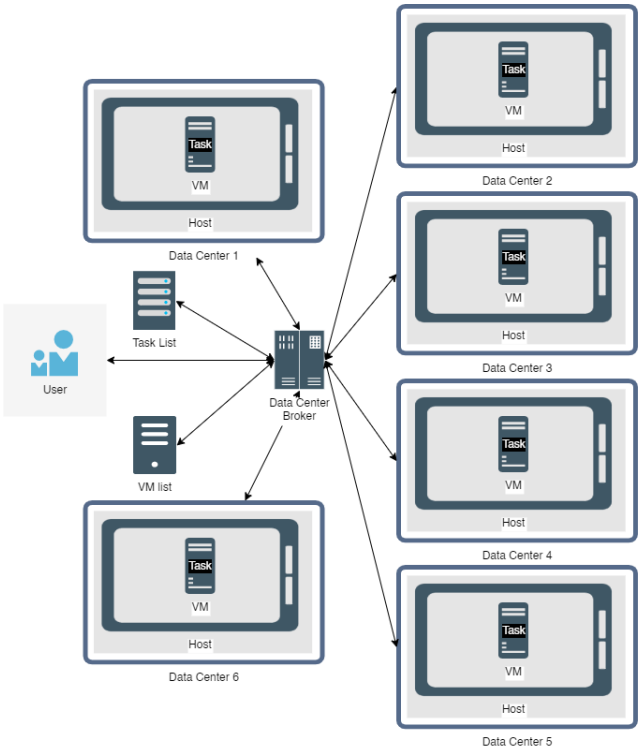


Fig. 4. Implementation Scheme


Fig. 5. VM Division Implementation Scheme

Each Virtual Machine and Data Center will have its specifications so that authors can see heterogeneous results. The specifications of the Virtual Machine, Host, and Data Center can be seen in detail in Table 4, Table 5, and Table 6.

TABLE 4
VIRTUAL MACHINE SPECIFICATIONS.

| ID | Memory (MB) | RAM (MB) | Processor MIPS | Processor Count | Bandwidth (MBps) |
|---|---|---|---|---|---|
| VM 1 | 1000 | 512 | 400 | 1 | 1000 |
| VM 2 | 1000 | 1024 | 500 | 1 | 1000 |
| VM 3 | 1000 | 2048 | 600 | 1 | 1000 |

TABLE 5
HOST SPECIFICATIONS.

| ID | Memory (MB) | RAM (MB) | Core Count | Core 1 MIPS | Core 2 MIPS | Core 3 MIPS | Core 4 MIPS | Max Power (W) | Static Power (%) |
|---|---|---|---|---|---|---|---|---|---|
| H1 | 1000000 | 128000 | 4 | 300 | 400 | 500 | 600 | 117 | 50 |
| H2 | 1000000 | 128000 | 4 | 300 | 400 | 500 | 600 | 117 | 50 |
| H3 | 1000000 | 128000 | 4 | 300 | 400 | 500 | 600 | 117 | 50 |

TABLE 6
DATA CENTER SPECIFICATIONS.

| ID | Processor Count | Host Count | Bandwidth (MBps) | Latency (ms) |
|---|---|---|---|---|
| D1 | 6 | 3 | 10000 | 6 |
| D2 | 6 | 3 | 10000 | 6 |
| D3 | 6 | 3 | 10000 | 8 |
| D4 | 6 | 3 | 10000 | 8 |
| D5 | 6 | 3 | 10000 | 10 |
| D6 | 6 | 3 | 10000 | 10 |

## *E. Scoring Parameters Used*

Testing and evaluation will be carried out by testing using a Cloud Environment simulation that runs on the CloudSim framework to test the efficiency of performing task scheduling and virtual machine (VM) allocation. The two scenarios, Genetic Algorithms only and Genetic Algorithms together with Artificial Neural Networks, will then be compared to find out which Cloud Provisioning system is more efficient. The comparison will be based on several parameters according to Table 7.

TABLE 7
SCORING PARAMETERS.

| No | Parameters Used |
|---|---|
| 1 | Makespan |
| 2 | Average Start Time |
| 3 | Average Finish Time |
| 4 | Average Execution Time |
| 5 | Total Wait Time |
| 6 | Scheduling Length |
| 7 | Throughput |
| 8 | Resource Utilization |
| 9 | Energy Consumption |
| 10 | Imbalance Degree |

This is important because to be able to find out the efficiency comparison between one task scheduling algorithm and the others the writer needs to know what parameters will be used to be able to measure them as well as the definitions and how to calculate these parameters. The following are the parameters that will be used to find out the efficiency of one algorithm (examples will be calculated based on Figure 6) [12].

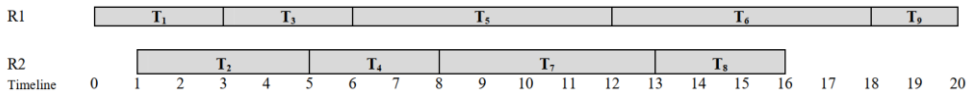


Fig. 6. Task Scheduling Example

TABLE 8
HOW TO CALCULATE PARAMETERS.

| Nama | Definition | How To Calculate |
|---|---|---|
| Makespan | The time it took to finish the last task given [12] | $max_{i \in tasks}\{F_i\}$<br><br>Example: 20-time unit because T9 which is the last task given finished processing at 20 unit |
| Average Start Time | The average time it took to start processing all the task [10] | $\frac{\sum_{i=1}^{n} The\ Time\ Ri\ Started}{nR}$<br><br>Example: (0+1)/2 = 0.5 |
| Average Finish Time | The average time it took to finish processing all the task [10] | $\frac{\sum_{i=1}^{n} The\ Time\ Ri\ Finished}{nR}$<br><br>Example: (20+16)/2 = 18 |
| Average Execution Time | The average time the task took to be processed [10] | $\frac{\sum_{i=1}^{n} Time\ Ti}{nT}$<br><br>Example:<br>(3+3+6+6+2+4+3+5+3)/9 = 32.3 |
| Total Wait Time | The time it took (or the delay) to start processing the first task [10] | Example: From Figure 4, the total wait time cannot be concluded since the task processing begins without a delay |
| Scheduling | The time it took to finish the simulation | $Scheduling\ Time + Makespan$ |

| | | |
|---|---|---|
| Length | from start to end [5] | Example: 0+20 = 20 |
| Throughput | The number of task finished per unit time [12] | $\frac{nT}{Makespan}$<br>Example: 9/20 = 0.45 |
| Resource Utilization | The rate of resource usage to process all the task given [12] | $\frac{\sum_{i=1}^{n} The\ Time\ Ri\ Finished}{Makespan * nR}$<br>Example: (20+15) / (20*2) = 0.875 |
| Energy Consumption | The total energy needed to process all the task given [5] | |
| Imbalance Degree | The degree of imbalance between the highest task length and the lowest in the dataset used [5] | $\frac{ETmax - ETmin}{ETavg}$<br>Dimana $ETmax, ETmin, ETavg$ is Maximum Execution Time, minimum, dan average in order from all the available data center.<br>Example: (6-2)/32.3 = 0.12 |

## IV. Results And Discussion

### A. *Makespan*

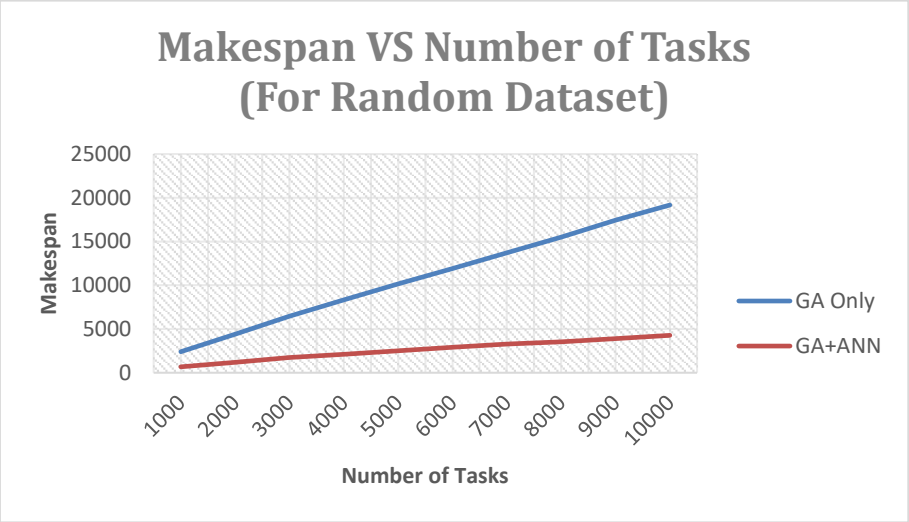


Fig. 7. Makespan Comparison Chart on Random Dataset

From Figure 7 it can be seen that the second scenario, namely the implementation of the Genetic Algorithm together with the Artificial Neural Network has a consistently lower value with an increase in the number of Tasks.

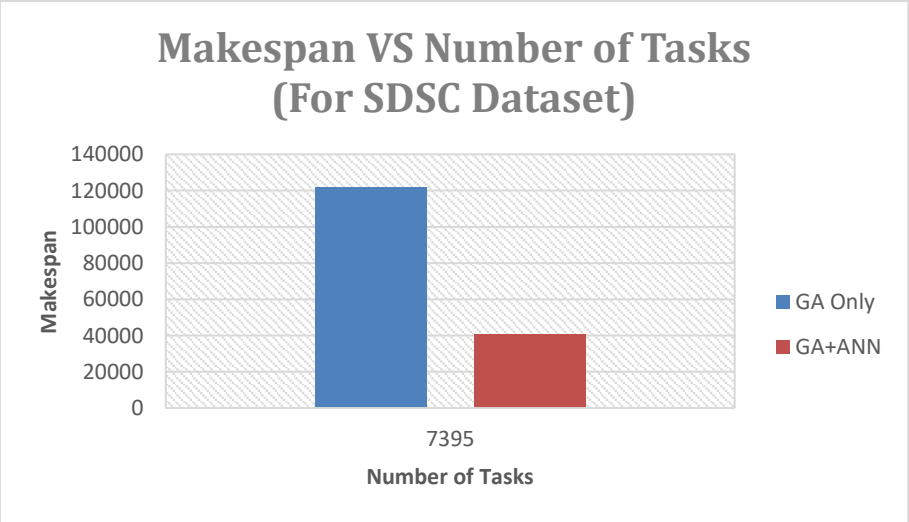


Fig. 8. Makespan Comparison Chart on SDSC Dataset

In the SDSC dataset in Figure 8, the same results can be seen where the implementation of the Genetic Algorithm together with the Artificial Neural Network produces a lower Makespan value compared to the implementation of the Genetic Algorithm alone. Here it can be concluded that the implementation of the Genetic Algorithm together with the Artificial Neural Network can indeed complete the last task given faster than the implementation of the Genetic Algorithm alone.

### *B. Average Start Time*

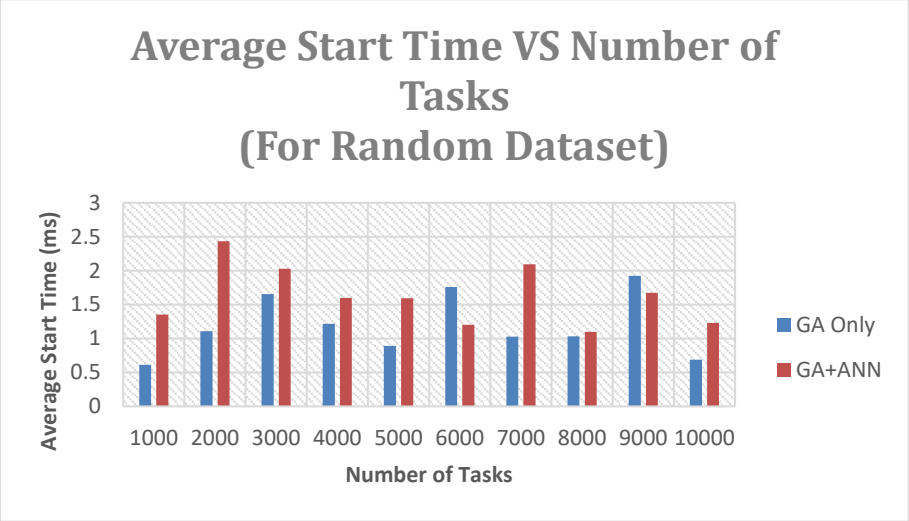


Fig. 9. Average Start Time Comparison Chart on Random Dataset

In Figure 9, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently lower values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets. This means that the implementation of the first scenario can start processing the Task faster than the implementation of the second scenario.

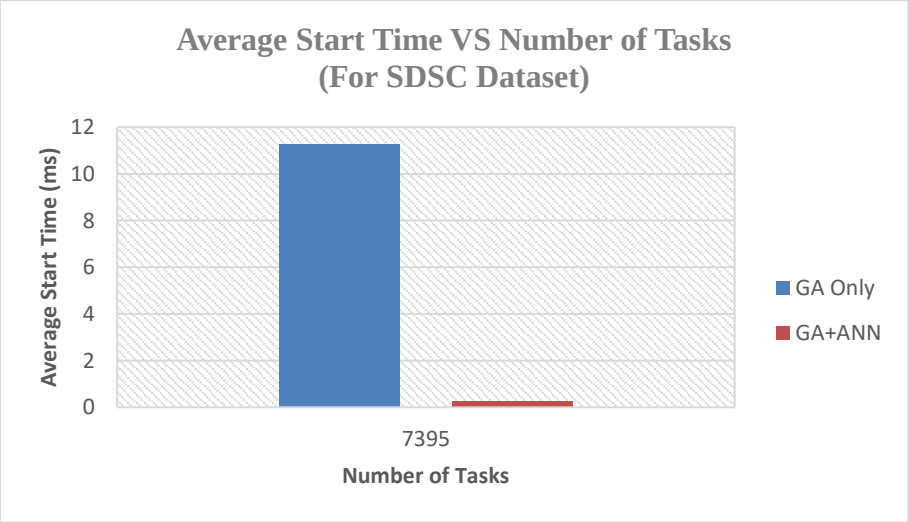


Fig. 10. Average Start Time Comparison Chart on SDSC Dataset

However, there are interesting things when looking at Figure 10. Here it can be seen that the Average Start Time implementation of the second scenario, namely the Genetic Algorithm together with the Artificial Neural Network produces a lower value. This means that the implementation of the second scenario can start processing the Task faster than the implementation of the first scenario. This is due to the larger and more varied range of data from the SDSC dataset compared to the random dataset. From this, it can be concluded that Genetic Algorithm is better when dealing with datasets whose data range and variety are not too large, and Genetic Algorithm together with Artificial Neural Networks can be used to deal with datasets with large data range and variety.

### C. *Average Finish Time*

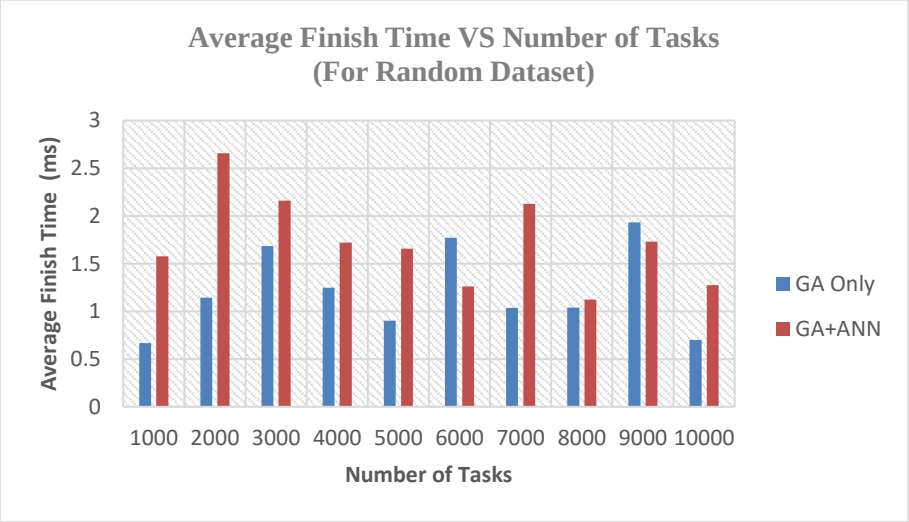


Fig. 11. Average Finish Time Comparison Chart on Random Dataset

In Figure 11, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently lower values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets. This means that the implementation of the first scenario can finish processing the Task faster than the implementation of the second scenario.

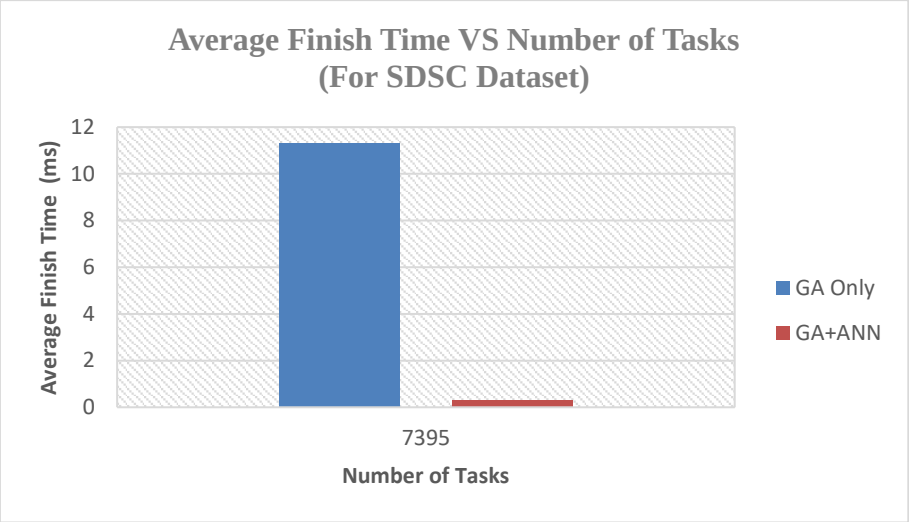


Fig. 12. Average Finish Time Comparison Chart on SDSC Dataset

However, there are interesting things when looking at Figure 12. Here it can be seen that the Average Finish Time implementation of the second scenario, namely the Genetic Algorithm together with the Artificial Neural Network produces a lower value. This means that the implementation of the second scenario can finish processing the Task faster than the implementation of the first scenario. This is due to the larger and more varied range of data from the SDSC dataset compared to the random dataset. From this, it can be concluded that Genetic Algorithm is better when dealing with datasets whose data range and variety are not too large, and Genetic Algorithm together with Artificial Neural Networks can be used to deal with datasets with large data range and variety.

### D. *Average Execution Time*

In Figure 13, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently lower values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets.

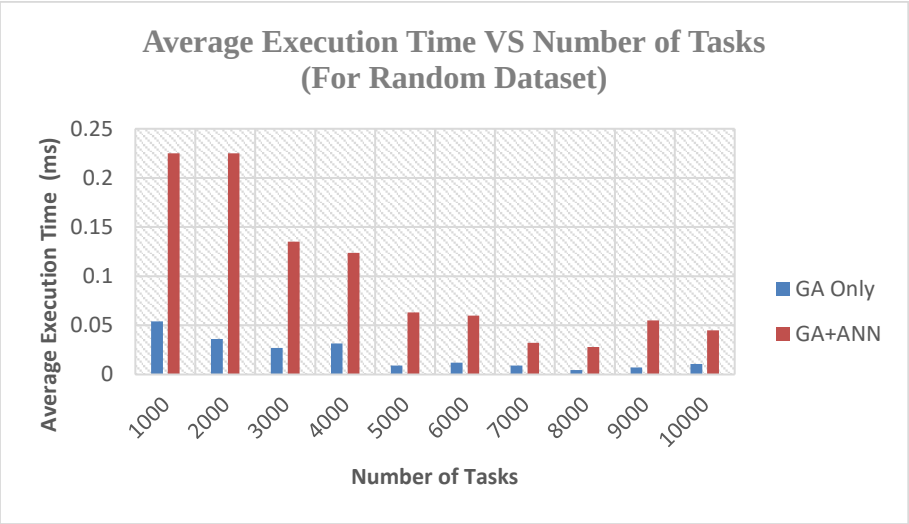


Fig. 13. Average Execution Time Comparison Chart on Random Dataset

In the SDSC dataset in Figure 14, the same results can be seen where the implementation of the Genetic Algorithm alone produces a lower Average Finish Time value compared to the implementation of the Genetic Algorithm together with the Artificial Neural Network.

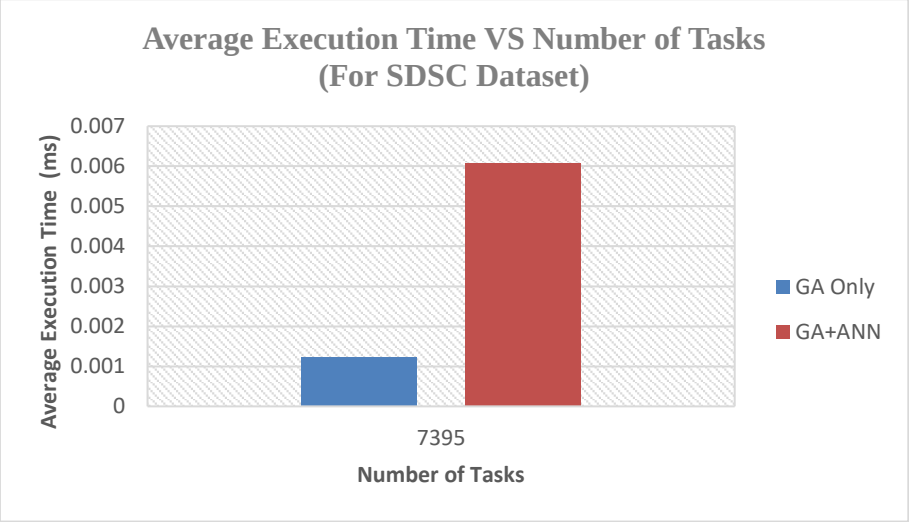


Fig. 14. Average Execution Time Comparison Chart on SDSC Dataset

*E. Total Wait Time*

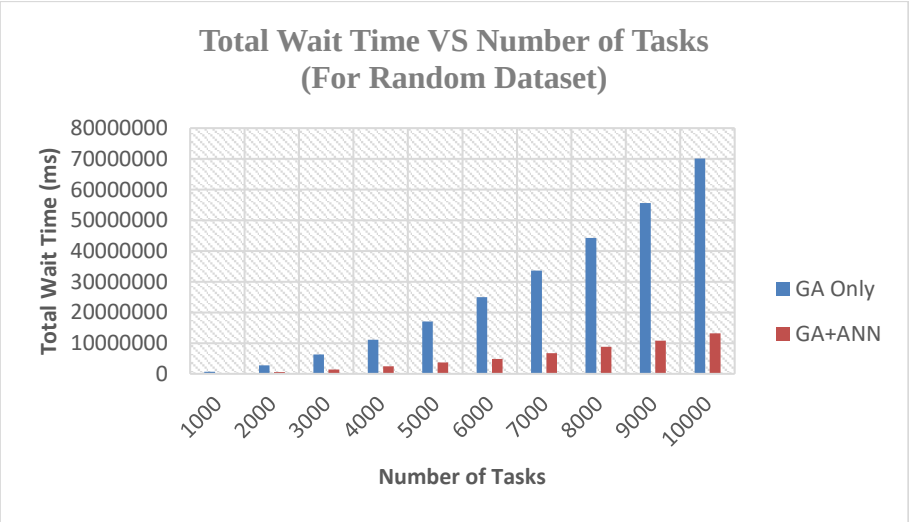


Fig. 15. Total Wait Time Comparison Chart on Random Dataset

In Figure 15, it can be seen that the implementation of the second scenario, namely the Genetic Algorithm together with an Artificial Neural Network, can produce consistently lower values than the implementation of the Genetic Algorithm alone on a random dataset. This means that the implementation of the second scenario is better at reducing the Delay that the Task must receive before processing than the implementation of the second scenario.

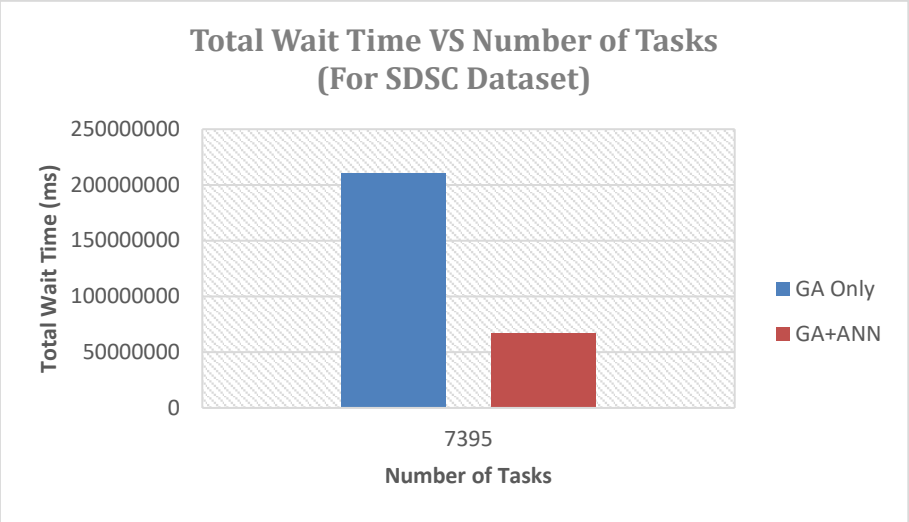


Fig. 16. Total Wait Time Comparison Chart on SDSC Dataset

In the SDSC dataset in Figure 16, the same results can be seen where the implementation of the Genetic Algorithm together with the Artificial Neural Network produces a lower Total Wait Time value compared to the implementation of the Genetic Algorithm alone. Here it can be concluded that the implementation of the Genetic Algorithm together with the Artificial Neural Network is indeed better at reducing the delay time that the Task must receive before processing than the implementation of the Genetic Algorithm alone.

### *F. Scheduling Length*

In Figure 17, it can be seen that the implementation of the second scenario, namely the Genetic Algorithm together with an Artificial Neural Network, can produce consistently lower values than the implementation of the Genetic Algorithm alone on random datasets. This means that the implementation of the second scenario is better at saving Cloud simulation time than the implementation of the second scenario.

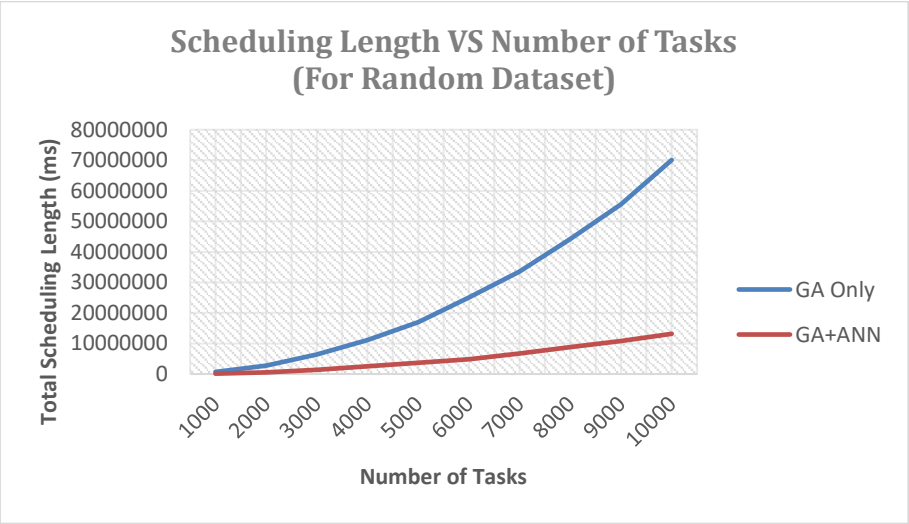


Fig. 17. Scheduling Length Comparison Chart on Random Dataset

In the SDSC dataset in Figure 18, the same results can be seen where the implementation of the Genetic Algorithm together with the Artificial Neural Network produces a lower Scheduling Length value compared to the implementation of the Genetic Algorithm alone. Here it can be concluded that implementing a Genetic Algorithm

together with an Artificial Neural Network is indeed better at saving Cloud simulation time than implementing a Genetic Algorithm alone.

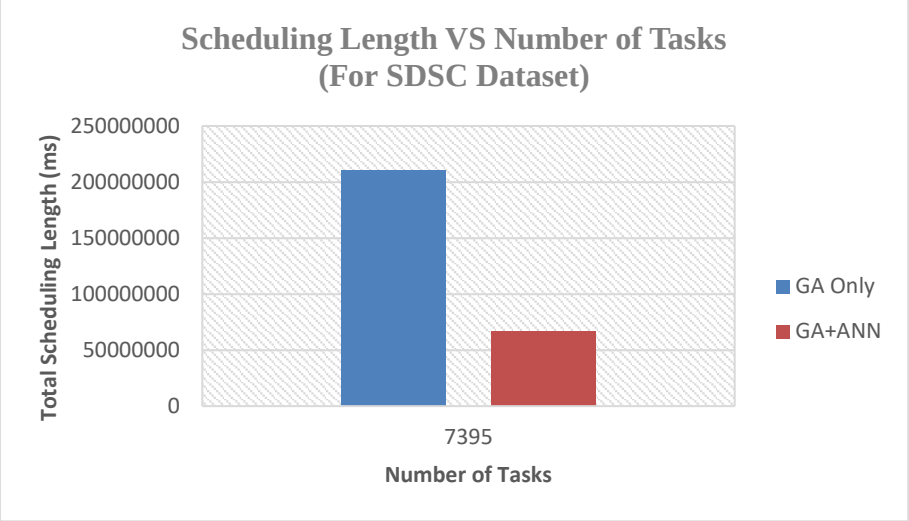


Fig. 18. Scheduling Length Comparison Chart on SDSC Dataset

*G. Throughput*

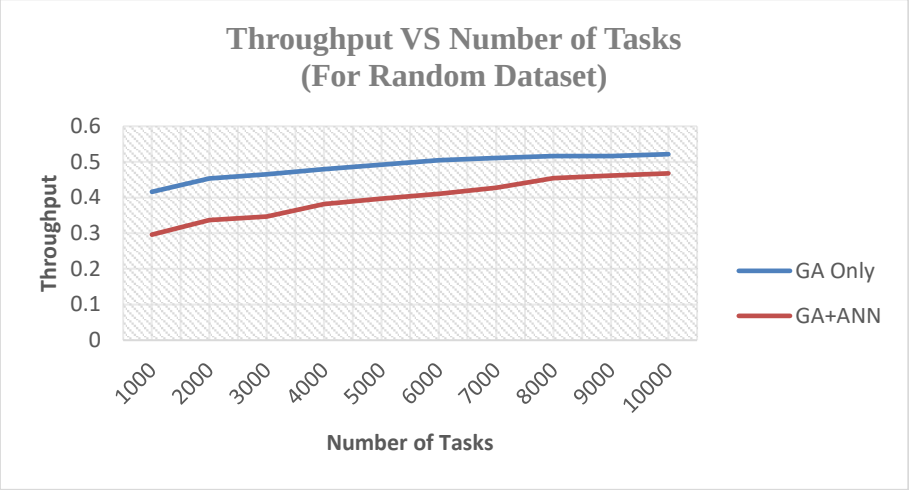


Fig. 19. Throughput Comparison Chart on Random Dataset

In Figure 19, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently higher values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets.

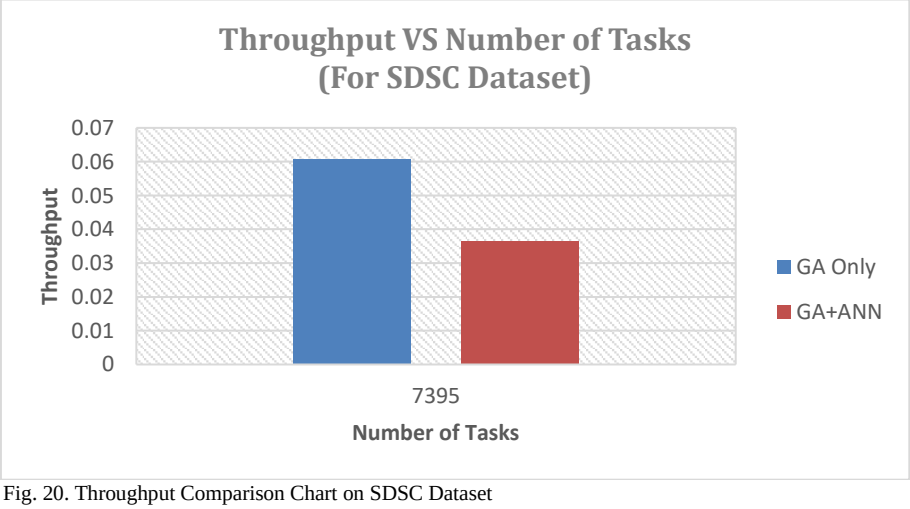


Fig. 20. Throughput Comparison Chart on SDSC Dataset

In the SDSC dataset in 20, the same results can be seen where the implementation of the Genetic Algorithm alone produces a higher Throughput value compared to the implementation of the Genetic Algorithm together with the Artificial Neural Network.

### *H. Resource Utilization*

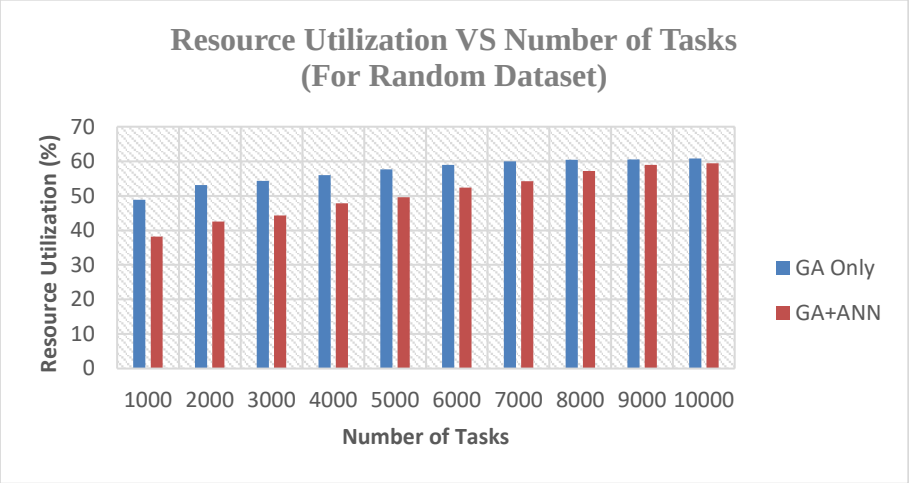


Fig. 21. Resource Utilization Chart on Random Dataset

In Figure 21, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently higher values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets. This means that the implementation of the first scenario is better in terms of the percentage of utilization of Cloud resources than the implementation of the second scenario.

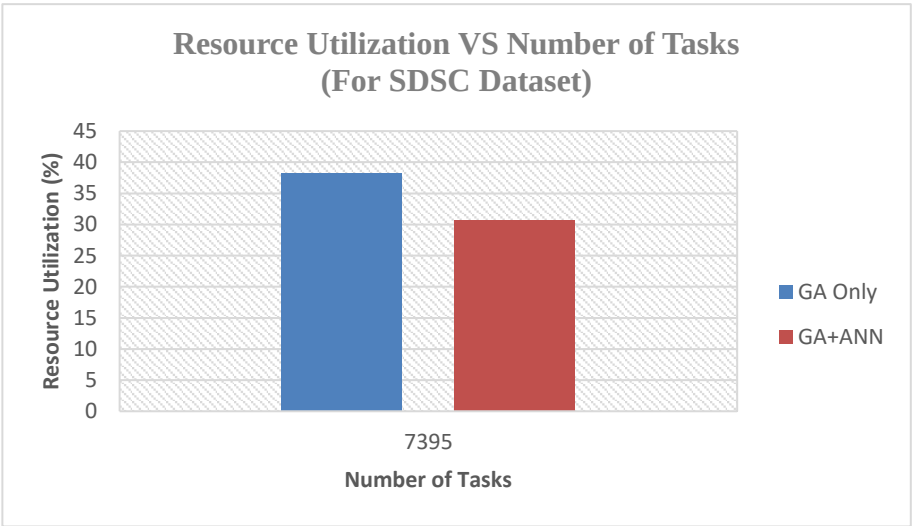


Fig. 22. Resource Utilization Chart on SDSC Dataset

In the SDSC dataset in Figure 22, the same results can be seen where the implementation of the Genetic Algorithm alone produces a higher Resource Utilization value compared to the implementation of the Genetic Algorithm together with an Artificial Neural Network. Here it can be concluded that the implementation of the Genetic Algorithm alone is indeed better in terms of the percentage of utilization of Cloud resources than the implementation of the Genetic Algorithm together with an Artificial Neural Network.

### *I. Energy Consumption*

In Figure 23, it can be seen that the implementation of the first scenario, namely the Genetic Algorithm alone, can produce consistently lower values than the implementation of the Genetic Algorithm together with the Artificial Neural Network on random datasets. This means that the implementation of the first scenario is better at saving energy use than the implementation of the second scenario.

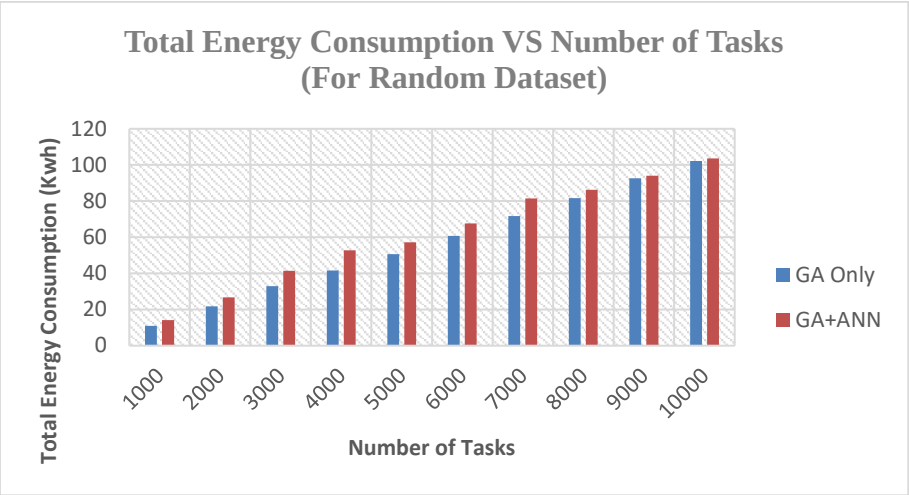


Fig. 23. Total Energy Consumption Chart on Random Dataset

In the SDSC dataset in Figure 24, the same results can be seen where the implementation of the Genetic Algorithm alone results in a lower Total Energy Consumption value compared to the implementation of the Genetic Algorithm together with an Artificial Neural Network. Here it can be concluded that implementing a Genetic Algorithm alone is indeed better at saving energy use than implementing a Genetic Algorithm together with an Artificial Neural Network.

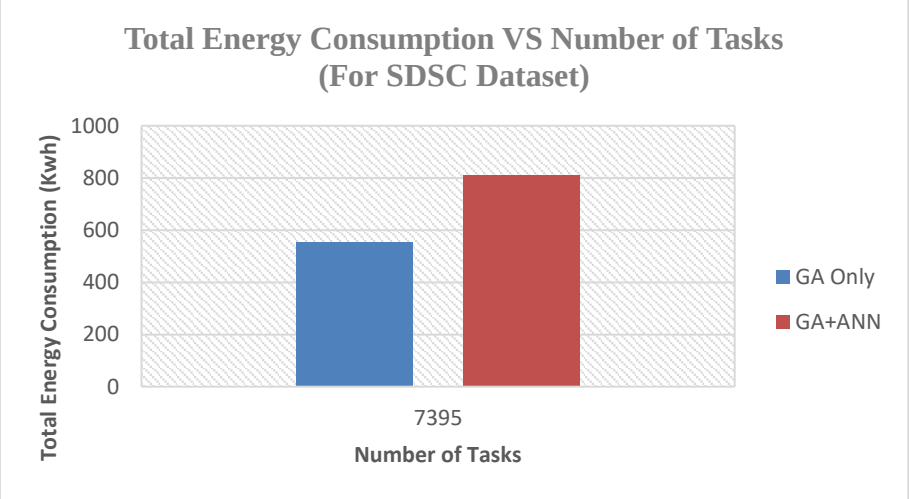


Fig. 24. Total Energy Consumption Chart on SDSC Dataset

## *J. Imbalance Degree*

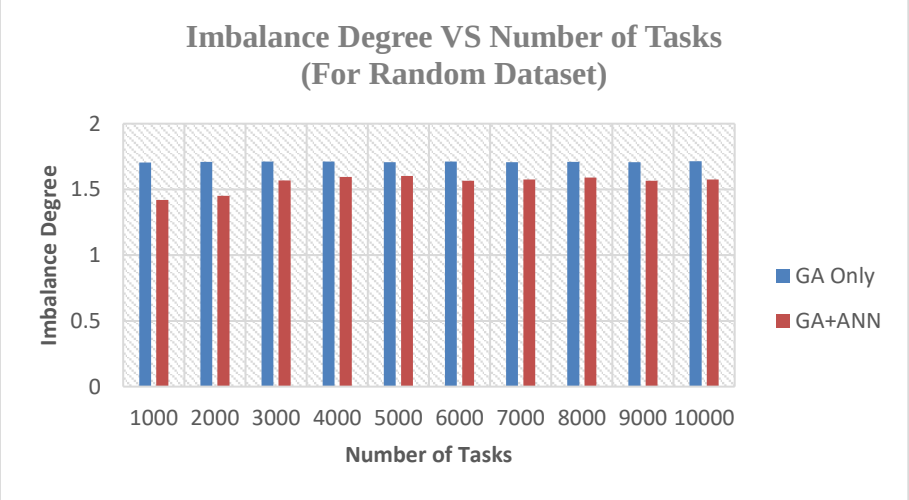


Fig. 25. Imbalance Degree Chart on Random Dataset

In Figure 25, it can be seen that the implementation of the second scenario, namely the Genetic Algorithm together with an Artificial Neural Network, can produce consistently lower values than the implementation of the Genetic Algorithm alone on random datasets.

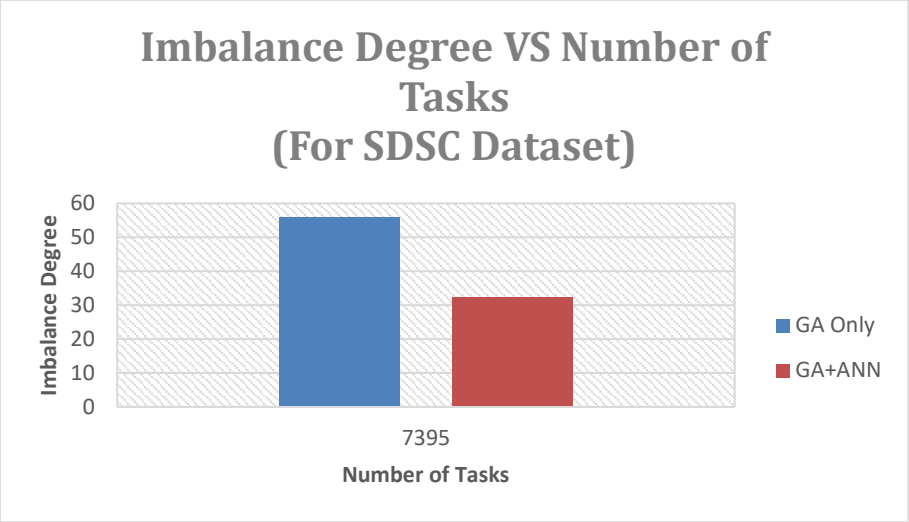


Fig. 26. Imbalance Degree Chart on SDSC Dataset

In the SDSC dataset in Figure 26, the same results can be seen where the implementation of the Genetic Algorithm together with the Artificial Neural Network produces a lower Imbalance Degree value compared to the implementation of the Genetic Algorithm alone. Here it can be concluded that implementing a Genetic Algorithm together with an Artificial Neural Network is indeed better at overcoming data imbalances than implementing a Genetic Algorithm alone.

## V. CONCLUSION

The implementation of a Genetic Algorithm in a scheduling system leads to higher resource utilization rates compared to a system without one, with rates ranging from 48% to 60%. When combined with an Artificial Neural Network, the resource utilization rate ranges from 38% to 59%. The Genetic Algorithm alone performs better at energy conservation and completing tasks within a shorter time frame, while the combination of the two performs better at handling data imbalances and quickly completing cloud simulations. If the goal is to save energy and maximize resource utilization, it is recommended to use just the Genetic Algorithm. If the goal is to quickly handle tasks with large data imbalances and do cloud provisioning, it is recommended to use the combination of a Genetic Algorithm and Artificial Neural Network.

## REFERENCES

[1] P. P. Ray, "An Introduction to Dew Computing: Definition, Concept and Implications," *IEEE*, vol. 6, pp. 723-737, 2017.

[2] M. H. Y. A. L.-G. Ahmadreza Montazerolghaem, "Green Cloud Multimedia Networking: NFV/SDN Based Energy-Efficient Resource Allocation," *IEEE*, vol. 4, no. 3, pp. 873 - 889, 2020.

[3] A. S. V. S. A. R. Michael Pawlish, "Analyzing Utilization Rates in Data Centers for Optimizing Energy," in *2012 International Green Computing Conference (IGCC)*, San Jose, 2012.

[4] J. Montgomery, "Cloud Provisioning," Tech Target, October 2020. [Online]. Available: https://www.techtarget.com/searchitchannel/definition/cloud-provisioning. [Accessed 22 July 2021].

[5] A. A. A. G.-E. A. S. K. R. R. Farouk A. Emara, "Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing," *International Journal of Intelligent Engineering & Systems*, pp. 1-12, 2021.

[6] M. G. B. Mathews, "On the Partition of Numbers," *Proceedings of the London Mathematical Society*, Vols. s1-28, no. 1, p. 486–490, 1896.

[7] T. Dantzig, Number : the language of science (The Masterpiece Science ed.), New York: Plume Book, 2007.

[8] M. Mitchell, An Introduction to Genetic Algorithms, Cambridge, MA: MIT Press, 1996.

[9] D. Kalita, "An Overview and Applications of Artificial Neural Networks," Analytics Vidhya, 6 April 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/an-overview-and-applications-of-artificial-neural-networks-ann. [Accessed 22 July 2022].

[10] P. D. P. G. G. S. Pradeep Singh Rawat, "Resource provisioning in scalable cloud using bio-inspired artificial neural network model," *Elsevier*, pp. 1-16, 2020.

[11] A. S. V. S. A. R. A. R. Michael Pawlish, "A Call for Energy Efficiency in Data Centers," *ACM SIGMOD Record*, pp. 45-51, 2014.

[12] A. M. S. D. P. Henning Titi Ciptaningtyas, "Survey on Task Scheduling Methods," in *2022 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Surabaya, 2022.

[13] T. S. D. S. C. (. B. H. log, "The San Diego Supercomputer Center (SDSC) Blue Horizon log," The San Diego Supercomputer Center (SDSC) , January 2003. [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_blue/index.html. [Accessed 22 July 2022].

[14] J. Heaton, Programming Neural Networks with Encog3 in Java, Saint Louis: Heaton Research, Inc., 2011.