

**PROPOSAL TUGAS AKHIR - IT184802**

# **CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK**

## **CLOUD PROVISIONING USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK**

**BRYAN YEHUDA MANNUEL**  
NRP 05311940000021

Dosen Pembimbing I  
**Henning Titi Ciptaningtyas, S.Kom, M.Kom.**  
NIP 19840708 201012 2 004

Dosen Pembimbing II  
**Ridho Rahman Hariadi, S.Kom, M.Sc**  
NIP 19870213 201404 1 001

**DEPARTEMEN TEKNOLOGI INFORMASI**  
Fakultas Teknologi Elektro dan Informatika Cerdas  
Institut Teknologi Sepuluh Nopember  
Surabaya  
2022



**PROPOSAL TUGAS AKHIR - IT184802**

# **CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK**

**BRYAN YEHUDA MANNUEL**

**NRP 05311940000021**

**Dosen Pembimbing I**

**Henning Titi Ciptaningtyas, S.Kom, M.Kom.**

**NIP 19840708 201012 2 004**

**Dosen Pembimbing II**

**Ridho Rahman Hariadi, S.Kom, M.Sc**

**NIP 19870213 201404 1 001**

**DEPARTEMEN TEKNOLOGI INFORMASI**

**Fakultas Teknologi Elektro dan Informatika Cerdas**

**Institut Teknologi Sepuluh Nopember**

**Surabaya**

**2022**



**FINAL PROJECT PROPOSAL - IT184802**

# **CLOUD PROVISIONING USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK**

**BRYAN YEHUDA MANNUEL**  
NRP 05311940000021

Supervisor I  
**Henning Titi Ciptaningtyas, S.Kom, M.Kom.**  
NIP 19840708 201012 2 004

Supervisor II  
**Ridho Rahman Hariadi, S.Kom, M.Sc**  
NIP 19870213 201404 1 001

**DEPARTEMEN OF INFORMATION TECHNOLOGY**  
Faculty of Intelligent Electrical and Informatics Technology  
Institut Teknologi Sepuluh Nopember  
Surabaya  
2022

## **LEMBAR PENGESAHAN**

### **CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK**

#### **PROPOSAL TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat  
Memperoleh Gelar Sarjana Komputer pada  
Program Studi S-1 Departemen Teknologi Informasi  
Departemen Teknologi Informasi  
Fakultas Teknologi Elektro dan Informatika Cerdas  
Institut Teknologi Sepuluh Nopember

Oleh : **Bryan Yehuda Mannuel**

NRP : **05311940000021**

Disetujui oleh Tim Penguji Tugas Akhir :

1. Henning Titi Ciptaningtyas, S.Kom, M. Kom.

NIP. 19840708 201012 2 004

  
(Pembimbing 1)

2. Ridho Rahman Hariadi, S.Kom, M. Sc.

NIP. 19870213 201404 1 001

  
(Pembimbing 2)

**SURABAYA**

**September 2022**

## **APPROVAL SHEET**

### **CLOUD PROVISIONING USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK**

#### **FINAL PROJECT PROPOSAL**

Submitted to fulfill one of the requirements  
for obtaining a degree Sarjana Komputer at  
Undergraduate Study Program of Department of Information Technology  
Department of Information Technology  
Faculty of Intelligent Electrical and Informatics Technology  
Institut Teknologi Sepuluh Nopember

By : **Bryan Yehuda Mannuel**

NRP : **05311940000021**

Approved by Final Project Examiner Team :

1. Henning Titi Ciptaningtyas, S.Kom, M. Kom.

NIP. 19840708 201012 2 004



(Supervisor 1)

2. Ridho Rahman Hariadi, S.Kom, M. Sc.

NIP. 19870213 201404 1 001



(Supervisor 2)

**SURABAYA**

**September 2022**

# CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK

Nama Mahasiswa / NRP : Bryan Yehuda Mannuel / 05311940000021  
Departemen : Teknologi Informasi  
Dosen Pembimbing 1 : Henning Titi Ciptaningtyas, S.Kom, M. Kom.  
Dosen Pembimbing 2 : Ridho Rahman Hariadi, S.Kom, M. Sc.

## Abstrak

Di era modern dimana penggunaan teknologi semakin pesat dan meningkat secara cepat, penggunaan *Cloud Computing* semakin banyak diminati. Oleh karena itu, diperlukan sebuah sistem manajemen sumber daya yang baik bagi sebuah *Cloud Service Provider* agar sistem *Cloud Computing* mereka dapat memanfaatkan kemampuan virtualisasi sumber daya secara maksimal dan meningkatkan tingkat penggunaan sumber daya *Cloud*. Namun, tantangan terbesar dalam membangun sebuah sistem manajemen sumber daya dalam *Cloud Computing* adalah mencari algoritma yang bisa memaksimalkan penggunaan sumber daya *Cloud*.

Untuk bisa mengatasi tantangan tersebut maka diadakanlah penelitian menggunakan algoritma *Genetic Algorithm* yang terinspirasi dari proses seleksi natural dan implementasi *Artificial Neural Network* yang didasarkan pada jaringan saraf biologis yang membentuk otak untuk membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) untuk memaksimalkan penggunaan sumber daya *Cloud*.

Penelitian dilakukan dalam dua skenario, dimana untuk skenario pertama akan digunakan *Genetic Algorithm* saja dan untuk skenario kedua akan digunakan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Pada kedua skenario ini, akan dihasilkan peningkatan efisiensi penjadwalan tugas pada tiap iterasi penggunaan *Genetic Algorithm* ataupun tiap iterasi pelatihan *Artificial Neural Network*. Iterasi ini akan dilakukan secara terus menerus hingga tidak didapati peningkatan efisiensi yang signifikan atau kondisi terminasi telah dipenuhi. Kedua skenario tersebut kemudian akan dilakukan perbandingan untuk mencari tahu mana sistem *Cloud Provisioning* yang lebih efisien berdasarkan berbagai macam parameter penilaian.

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

# CLOUD PROVISIONING USING GENETIC ALGORITHM AND ARTIFICIAL NEURAL NETWORK

Name of Student / NRP : Bryan Yehuda Mannuel / 05311940000021  
Department : Information Technology  
Advisor 1 : Henning Titi Ciptaningtyas, S.Kom, M. Kom. /  
Advisor 2 : Ridho Rahman Hariadi, S.Kom, M. Sc.

## Abstract

In this modern era where the use of technology is growing rapidly and increasing rapidly, the use of Cloud Computing is increasingly in demand. 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 resource utilization of Cloud resources. However, the biggest challenge in building a resource management system in Cloud Computing is to find an algorithm that can maximize the use of Cloud resources.

To be able to overcome these challenges, this research was conducted using a Genetic Algorithm inspired by the natural selection process and the implementation of an Artificial Neural Network which is based on a biological neural network that forms the brain to build a task scheduling and virtual machine (VM) allocation system to maximize the use of cloud resources.

The research was conducted using two different scenarios, the first of which just used the Genetic Algorithm and the second of which combined it with an Artificial Neural Network. With each iteration of the Genetic Algorithm or Artificial Neural Network training in these two scenarios, task scheduling effectiveness will rise. Until no appreciable improvement in efficiency is identified or the termination conditions have been satisfied, this iteration will continue. After that, the two scenarios will be contrasted to determine whether Cloud Provisioning solution is more effective based on several evaluation parameter.

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

## DAFTAR ISI

LEMBAR PENGESAHAN.....	iv
APPROVAL SHEET .....	v
Abstrak.....	vi
Abstract.....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	x
DAFTAR TABEL.....	xi
BAB I PENDAHULUAN .....	1
1.1    Latar Belakang .....	1
1.2    Rumusan Masalah.....	2
1.3    Batasan Masalah .....	2
1.4    Tujuan .....	2
1.5    Manfaat .....	2
1.6    Metodologi.....	2
1.6.1    Penyusunan Proposal Tugas Akhir .....	3
1.6.2    Studi Literatur .....	3
1.6.3    Analisis dan Desain Perangkat Lunak .....	3
1.6.4    Implementasi Perangkat Lunak.....	4
1.6.5    Pengujian dan Evaluasi.....	4
1.6.6    Penyusunan Buku Tugas Akhir .....	4
1.7    Sistematika Penulisan .....	4
BAB II TINJAUAN PUSTAKA.....	6
2.1    Tinjauan Penelitian Sebelumnya .....	6
2.2    Dasar Teori .....	8
2.2.1    Cloud Computing .....	9
2.2.2    Cloud Provisioning .....	11
2.2.3    Teknologi Virtualisasi .....	11
2.2.4    Knapsack Problem .....	12
2.2.5    Genetic Algorithm .....	13
2.2.6    Alasan Pemilihan Genetic Algorithm .....	15
2.2.7    Artificial Neural Network.....	15



2.2.8	Alasan Pemilihan Artificial Neural Network.....	18
2.2.9	CloudSim.....	19
2.2.10	Library CloudSim .....	19
2.2.11	Alasan Pemilihan CloudSim.....	21
2.2.12	Library Encog.....	22
2.2.13	Alasan Pemilihan Encog .....	22
2.2.14	Definisi Parameter Penggunaan Sumber Daya Cloud .....	22
2.2.15	Tingkatan Penggunaan Sumber Daya.....	24
2.2.16	Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon Log.....	24
2.2.17	Eclipse IDE.....	24
<b>BAB III METODOLOGI .....</b>		<b>26</b>
3.1	Tahapan Metodologi Penelitian .....	26
3.2	Deskripsi Metodologi Penelitian .....	26
3.2.1	Identifikasi Permasalahan .....	26
3.2.2	Studi Literatur.....	27
3.2.3	Analisis dan Desain Perangkat Lunak .....	27
3.2.4	Implementasi Perangkat Lunak.....	32
3.2.5	Pengujian dan Evaluasi.....	34
3.2.6	Penyusunan Buku Tugas Akhir .....	34
3.3	Jadwal Pengerjaan Tugas Akhir.....	35
<b>DAFTAR PUSTAKA .....</b>		<b>36</b>

## DAFTAR GAMBAR

Gambar 2.1 Model Layanan <i>Cloud Computing</i> .....	10
Gambar 2.2 <i>Knapsack Problem</i> .....	12
Gambar 2.3 <i>Flowchart</i> Cara Kerja <i>Genetic Algorithm</i> .....	14
Gambar 2.4 <i>Artificial Neural Network</i> dengan 4 Lapisan .....	16
Gambar 2.5 <i>Flowchart</i> Cara Kerja <i>Artificial Neural Network</i> .....	18
Gambar 2.6 <i>Melbourne CLOUDS Lab</i> Pengembang <i>CloudSIM</i> .....	19
Gambar 2.7 Logo <i>Encog</i> .....	22
Gambar 2.8 Contoh Penjadwalan Tugas .....	23
Gambar 2.9 Logo <i>Eclipse IDE</i> .....	25
Gambar 3.1 Representasi Kromosom.....	28
Gambar 3.2 <i>Flowchart</i> Skenario Pertama .....	29
Gambar 3.3 <i>Flowchart</i> Skenario Pertama .....	31
Gambar 3.4 <i>Activation Function Sigmoid</i> dan <i>ReLu</i> .....	32
Gambar 3.5 Skema Implementasi Perangkat Lunak .....	32
Gambar 3.6 Skema Implementasi Perangkat Lunak Pembagian <i>VM</i> .....	33

## DAFTAR TABEL

Tabel 2.1 Penelitian Sebelumnya .....	6
Tabel 2.2 Penulisan <i>Bounded Knapsack Problem</i> .....	12
Tabel 2.3 <i>Namespace CloudSim</i> .....	20
Tabel 2.4 Parameter Tingkat Penggunaan Sumber Daya <i>Cloud</i> .....	23
Tabel 3.1 Dataset yang Digunakan .....	26
Tabel 3.2 Spesifikasi <i>Genetic Algorithm</i> .....	27
Tabel 3.3 <i>Fitness Function</i> .....	28
Tabel 3.4 Total Waktu Eksekusi .....	28
Tabel 3.5 Persentase Kesalahan .....	29
Tabel 3.6 Spesifikasi <i>Artificial Neural Network</i> .....	30
Tabel 3.7 Spesifikasi <i>Node Artificial Neural Network</i> .....	30
Tabel 3.8 Spesifikasi <i>Virtual Machine</i> .....	33
Tabel 3.9 Spesifikasi <i>Host</i> .....	34
Tabel 3.10 Spesifikasi <i>Data Center</i> .....	34
Tabel 3.11 Parameter yang Digunakan .....	34
Tabel 3.12 Jadwal Pengerjaan .....	35



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Di era modern dimana penggunaan teknologi semakin pesat dan meningkat secara cepat, penggunaan *Cloud Computing* semakin banyak diminati (Ray, 2017). *Cloud Computing* adalah ketersediaan sumber daya sistem komputer sesuai permintaan, seperti penyimpanan data dan daya komputasi, tanpa pengelolaan langsung oleh pengguna (Ahmadreza Montazerolghaem, 2020). Namun, penelitian terbaru menyatakan bahwa tingkat penggunaan sumber daya *Cloud* di banyak pusat data masih terbilang cukup rendah. Hal ini diakibatkan karena masih banyak *Cloud Service Provider* yang tidak menggunakan kemampuan virtualisasi yang dimiliki oleh *Cloud Computing* secara maksimal sehingga berakibat kepada pembuangan energi dan tenaga secara sia-sia (Michael Pawlish A. S., 2012). Oleh karena itu, diperlukan sebuah sistem manajemen sumber daya yang baik bagi sebuah *Cloud Service Provider* agar sistem *Cloud Computing* mereka dapat memanfaatkan kemampuan virtualisasi sumber daya secara maksimal dan meningkatkan tingkat penggunaan sumber daya *Cloud*.

*Cloud provisioning* adalah fitur utama dari sistem *Cloud Computing*, yang berkaitan dengan cara pelanggan mendapatkan sumber daya *Cloud* dari *Cloud Service Provider* (Montgomery, 2020). Penjadwalan tugas dan alokasi mesin virtual (VM) memainkan peran penting dalam *Cloud provisioning*. Hal ini dikarenakan sistem *Cloud Computing* bergantung pada teknologi virtualisasi yang memungkinkan sumber daya dari satu sumber daya *Cloud* fisik untuk dibagi menjadi beberapa lingkungan terisolasi yang berjalan di mesin virtual (VM) (Farouk A. Emara, 2021).

Tantangan terbesar dalam membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) dalam *Cloud Computing* adalah mencari algoritma yang bisa memaksimalkan penggunaan sumber daya *Cloud*. Tantangan ini biasa disebut sebagai “*Knapsack Problem*” dimana “Diberikan sekumpulan benda, masing-masing dengan bobot dan nilai tertentu, maka tentukan jumlah setiap benda untuk dimasukkan kedalam koleksi sehingga bobot totalnya kurang dari atau sama dengan batas yang diberikan dan nilai totalnya sebesar mungkin. (G. B. Mathews, 1896)”. Tantangan ini sering muncul dalam pengalokasian sumber daya di mana pengambil keputusan harus memilih dari serangkaian tugas yang tidak dapat dibagi di bawah anggaran tetap atau batasan waktu (Dantzig, 2007).

Untuk bisa mengatasi hal tersebut maka diadakanlah penelitian menggunakan algoritma *Genetic Algorithm* yang terinspirasi dari proses seleksi natural dan implementasi *Artificial Neural Network* yang didasarkan pada jaringan saraf biologis yang membentuk otak untuk membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) untuk memaksimalkan penggunaan sumber daya *Cloud*. *Genetic Algorithm* digunakan untuk menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya *Cloud* dengan mengandalkan operator yang terinspirasi secara biologis seperti mutasi, penyilangan dan seleksi (Mitchell, 1996). Ditambahkan dengan implementasi *Artificial Neural Network* untuk mempelajari, memproses, dan memprediksi hasil dari solusi optimasi. (Kalita, 2022).

## 1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan *Genetic Algorithm* untuk menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya *Cloud*?
2. Bagaimana cara mengimplementasikan *Artificial Neural Network* bersamaan dengan *Genetic Algorithm* untuk mempelajari, memproses, dan memperkirakan model dari solusi optimasi penggunaan sumber daya *Cloud*?

## 1.3 Batasan Masalah

Batasan masalah untuk pengerjaan tugas akhir ini adalah sebagai berikut:

1. Proses penelitian akan berupa simulasi *Cloud Environment* dengan menggunakan *CloudSIM Java*, *Library Encog* untuk pengkodean *Artificial Neural Network* dan *Eclipse IDE*.
2. Dataset yang digunakan untuk penelitian ini akan menggunakan dataset yang dibuat sendiri secara acak dan juga mengambil dari dataset *The San Diego Supercomputer Center (SDSC) Blue Horizon logs* (log, 2003).
3. Proses penelitian akan ditulis menggunakan Bahasa *Java* menggunakan *IDE Eclipse*.
4. Proses penelitian ini tidak akan membahas mengenai biaya penggunaan sistem *Cloud Computing*.

## 1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah menyelesaikan “*Knapsack Problem*” yang ditemui pada saat melakukan *Cloud Provisioning* menggunakan *Genetic Algorithm* dan *Artificial Neural Network* untuk membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) yang bisa memaksimalkan penggunaan sumber daya *Cloud* sehingga meningkatkan tingkat penggunaan sumber daya *Cloud* dan mengurangi energi dan tenaga yang terbuang sia-sia.

## 1.5 Manfaat

Manfaat dari pengerjaan tugas akhir ini adalah sebagai berikut:

1. Membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM).
2. Mencegah penurunan performa sumber daya *Cloud*.
3. Meningkatkan tingkat penggunaan sumber daya *Cloud*.
4. Mengurangi *Execution Time* penggunaan sumber daya *Cloud*.
5. Mengurangi energi dan tenaga yang terbuang sia-sia saat penggunaan penggunaan sumber daya *Cloud*.

## 1.6 Metodologi

Metodologi pengerjaan yang diterapkan pada Tugas Akhir ini mempunyai tahapan-tahapan sebagai berikut

### 1.6.1 Penyusunan Proposal Tugas Akhir

Penyusunan tugas akhir ini berisi tentang pendahuluan dari tugas akhir yang akan di laksanakan dimana terdiri dari latar belakang dimana menjelaskan alasan pengambilan judul tugas akhir, rumusan masalah, batasan masalah, tujuan akhir dari tugas akhir, serta manfaat dari tugas akhir. Pada proposal ini juga terdapat juga tinjauan Pustaka yang digunakan dalam referensi pembuatan tugas akhir.

### 1.6.2 Studi Literatur

Proposal tugas akhir ini menggunakan beberapa literatur yang sudah pernah dibuat sebelumnya seperti “*Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment*” (Farouk A. Emara, 2021), “*Resource provisioning in scalable cloud using bio-inspired artificial neural network model*” (Pradeep Singh RawatPriti, 2020), “*A Call for Energy Efficiency in Data Centers*” (Michael Pawlish A. S., 2014), dan “*Survey on Task Scheduling Methods in Cloud RPS System*” (Henning Titi Ciptaningtyas, 2022).

### 1.6.3 Analisis dan Desain Perangkat Lunak

Langkah-langkah dari analisis dan desain perangkat lunak yang akan dibuat adalah melakukan analisa dan *preprocessing* pada dataset yang akan digunakan. Hasil dari *preprocessing* pada dataset ini akan menghasilkan dataset yang bersih sehingga siap untuk dilakukan proses selanjutnya. Pada penelitian ini akan dijalankan dua skenario dimana untuk skenario pertama akan dilakukan penjadwalan tugas dan alokasi mesin virtual (VM) menggunakan *Genetic Algorithm* saja dan untuk skenario kedua akan dilakukan dilakukan penjadwalan tugas dan alokasi mesin virtual (VM) menggunakan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Kedua skenario ini nantinya akan dilakukan perbandingan untuk ditemukan kombinasi algoritma mana yang lebih efisien dalam melakukan penjadwalan tugas dan alokasi mesin virtual (VM).

Untuk skenario pertama akan dilakukan inisialisasi populasi awal dari jadwal tugas (*Schedule*) menggunakan *Genetic Algorithm*. Jadwal (*Schedule*) ini akan berisi ID dari tiap mesin virtual (VM) yang dianggap sesuai oleh *Genetic Algorithm* berdasarkan *Fitness Function* sebagai tempat *Task* dijalankan. Jadwal-jadwal ini kemudian akan dilakukan seleksi, mutasi, dan penilangan menggunakan *Genetic Algorithm* kembali untuk menghasilkan generasi baru yang lebih efisien dari generasi sebelumnya berdasarkan *Fitness Function*.

Untuk skenario kedua, dataset dari Jadwal (*Schedule*) yang telah dihasilkan oleh *Genetic Algorithm* akan dibagi menjadi 2, yaitu dataset pembelajaran dan dataset pengujian. dataset pembelajaran akan dilakukan analisa dan dipelajari terlebih dahulu oleh *Artificial Neural Network* untuk menghasilkan model penjadwalan *Task* kepada mesin virtual (VM) yang sesuai berdasarkan *Fitness Function* milik *Genetic Algorithm*. Model ini nantinya akan dilakukan pengujian menggunakan dataset pengujian untuk mencari tahu apakah penggunaan model ini bisa menghasilkan efisiensi dalam melakukan penjadwalan tugas dan alokasi mesin virtual (VM) yang lebih tinggi daripada penggunaan *Genetic Algorithm* saja.

#### **1.6.4 Implementasi Perangkat Lunak**

Implementasi dari perangkat lunak akan menggunakan *CloudSIM*. Sebuah kerangka kerja *Open Source*, yang digunakan untuk mensimulasikan layanan *Cloud Computing* dan menggunakan *Library Encog* untuk pengkodean *Artificial Neural Network*.

#### **1.6.5 Pengujian dan Evaluasi**

Pengujian dan evaluasi akan dilaksanakan dengan uji coba menggunakan simulasi *Cloud Environment* yang dijalankan pada *CloudSIM* untuk menguji efisiensi melakukan *Cloud Provisioning* menggunakan algoritma *Genetic Algorithm* dan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*.

#### **1.6.6 Penyusunan Buku Tugas Akhir**

Pada tahap ini akan dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

#### **1.7 Sistematika Penulisan**

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

1. Bab I : Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, Batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II : Dasar Teori

Bab ini berisi kajian teori dari metode dan algoritma yang digunakan dalam penyusunan Tugas Akhir ini.

3. Bab III : Desain

Bab ini menjelaskan desain algoritma yang akan dibangun berdasarkan dasar teori dan digunakan dalam penyelesaian.

4. Bab IV : Implementasi

Bab ini membahas implementasi dari perancangan yang telah dibuat pada bab sebelumnya. Terdapat juga penjelasan berupa kode program yang digunakan untuk proses implementasi.



## 5. Bab V : Uji Coba dan Analisis

Bab ini membahas tahapan uji coba, kemudian hasil uji coba dievaluasi dan dianalisis terhadap kinerja dari algoritma yang diimplementasikan.

## 6. Bab VI : Kesimpulan dan Saran

Bab ini merupakan bab yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses dan tertulis saat pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Tinjauan Penelitian Sebelumnya

Dalam penelitian ini, akan digunakan beberapa penelitian terdahulu sebagai pedoman dan referensi dalam melaksanakan proses pengerjaan tugas akhir. Informasi yang disampaikan dalam Tabel 2.1 berisi informasi mengenai penelitian sebelumnya, hasil penelitian, dan hubungannya terhadap tugas akhir.

Tabel 2.1 Penelitian Sebelumnya

No	Tahun, Penulis	Pembahasan
1	<i>Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment</i> (Farouk A. Emara, 2021)	
	Farouk A. Emara, Ahmed. A. A. Gad-Elrab, Ahmed Sobhi, K. R. Raslan	<p>Jurnal ini membahas mengenai berbagai macam tantangan dalam membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) dalam sebuah sistem <i>Cloud Computing</i>. Hal ini dikarenakan banyaknya heterogenitas dalam sistem <i>Cloud Computing</i> yang mengakibatkan banyak algoritma dan penelitian sebelumnya hanya memperhitungkan satu target capaian saja. Jurnal ini menyarankan penggunaan <i>Genetic Algorithm</i> yang dimodifikasi untuk bisa menyelesaikan lebih dari satu target capaian yaitu memaksimalkan <i>Resource Utilization</i>, <i>Load Balancing</i>, dan <i>Power Management</i> dalam sebuah sistem <i>Cloud Computing</i>.</p> <p>Algoritma yang disarankan ini menggunakan sebuah struktur matriks untuk merepresentasikan kromosom dari GAS (komponen yang dicari optimisasinya dalam penggunaan <i>Genetic Algorithm</i>) yang terdiri dari <i>ID Tasks</i>, <i>ID VM</i>, dan <i>ID Data Center</i>. Hasil penelitian dari jurnal ini menunjukkan bahwa penggunaan <i>Genetic Algorithm</i> yang dimodifikasi akan menghasilkan performa penjadwalan tugas dan alokasi mesin virtual (VM) yang lebih baik dari beberapa algoritma lainnya (<i>ETVMC</i>, <i>TSACS</i>, dan <i>ACO</i>) dinilai dari <i>Makespan</i>, <i>Scheduling Length</i>, <i>Throughput</i>, <i>Resource Utilization</i>, <i>Energy Consumption</i>, dan <i>Imbalance Degree</i>.</p> <p>Jurnal ini dijadikan referensi untuk penelitian ini dikarenakan hasil dari penelitiannya membuktikan bahwa penggunaan <i>Genetic Algorithm</i> yang dimodifikasi bisa memberikan hasil yang lebih baik daripada algoritma penjadwalan tugas dan alokasi mesin virtual (VM) lainnya. Penulis mengambil referensi ini dan ingin membandingkan apakah jika <i>Genetic Algorithm</i> yang dibantu dengan <i>Artificial Neural Network</i> bisa memberikan hasil yang lebih baik lagi daripada dengan <i>Genetic Algorithm</i> saja jika dinilai dari <i>Makespan</i>, <i>Scheduling Length</i>, <i>Throughput</i>, <i>Resource</i></p>

No	Tahun, Penulis	Pembahasan
		<i>Utilization, Energy Consumption, dan Imbalance Degree.</i> Terakhir, penulis juga mengambil referensi penggunaan <i>ID Tasks, ID VM, dan ID Data Center</i> sebagai representasi kromosom dari penggunaan <i>Genetic Algorithm</i> .
2	<i>Resource Provisioning in Scalable Cloud Using Bio-inspired Artificial Neural Network Model</i> (Pradeep Singh Rawat, 2020)	
	Pradeep Singh Rawat, Priti Dimri, Punit Gupta, G.P. Saroha	<p>Jurnal ini membahas mengenai berbagai macam parameter yang bisa digunakan untuk mengukur efisiensi sebuah algoritma penjadwalan tugas dan alokasi mesin virtual (VM) dalam sebuah sistem <i>Cloud Computing</i>. Parameter tersebut contohnya adalah <i>Average Start Time, Average Finish Time, Average Execution Time, dan Scheduling Time</i>. Parameter ini kemudian digunakan untuk membandingkan beberapa algoritma yaitu <i>BB-BC, GA-Cost, GA-Exe, dan GA-ANN (Genetic Algorithm – Artificial Neural Network)</i> sebagai algoritma utama yang dihipotesiskan penelitian ini menjadi yang terbaik diantara algoritma lainnya.</p> <p>Penelitian ini pun dilakukan menggunakan beberapa variabel kontrol seperti jumlah <i>Task</i>, jumlah <i>VM</i>, jumlah <i>Data Center</i>, <i>Bandwith, CPU, Ram</i>, dan lain sebagainya. Hal ini berguna agar nantinya perbandingan antar algoritma bisa dilaksanakan secara objektif. Penelitian ini juga menggunakan dua sumber data yaitu menggunakan data buatan sendiri dan juga dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i>. Pada akhirnya ditemukan bahwa benar algoritma <i>GA-ANN</i> bisa menghasilkan sistem penjadwalan tugas dan alokasi mesin virtual (VM) yang paling efisien dengan pengurangan 82.63% pada tingkat kesalahan yang terjadi, peningkatan 26.81% pada jumlah <i>Task</i> yang berhasil diselesaikan secara sukses, 10.66% pada <i>Execution Time</i>, dan 69.94% pada <i>Scheduling Time</i>.</p> <p>Jurnal ini dijadikan referensi untuk penelitian ini dikarenakan hasil dari penelitiannya membuktikan bahwa penggunaan <i>GA-ANN</i> bisa memberikan hasil yang lebih baik daripada algoritma penjadwalan tugas dan alokasi mesin virtual (VM) lainnya yang dibahas. Penulis mengambil referensi ini dan ingin membandingkan apakah jika <i>Genetic Algorithm</i> yang dibantu dengan <i>Artificial Neural Network</i> bisa memberikan hasil yang lebih baik lagi daripada dengan <i>Genetic Algorithm</i> saja jika dinilai dari <i>Average Start Time, Average Finish Time, Average Execution Time, dan Scheduling Time</i>. Penulis juga mengambil referensi penggunaan dua sumber data yaitu menggunakan data buatan sendiri dan juga dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i>. Terakhir, penulis juga ingin membuat lingkungan percobaan yang sama dengan menggunakan variabel kontrol yang sudah didefinisikan dalam penelitian ini seperti jumlah <i>Task</i>, jumlah <i>VM</i>, jumlah <i>Data Center, Bandwith, CPU, Ram</i>, dan lain sebagainya.</p>

No	Tahun, Penulis	Pembahasan
3	<i>A Call for Energy Efficiency in Data Centers</i> (Michael Pawlish A. S., 2014)	
	Michael Pawlish, Aparna S. Varde, Stefan A. Robila, Anand Ranganathan	<p><i>Conference Paper</i> ini berusaha mencari tahu apakah sistem <i>Cloud Computing</i> yang ada saat ini sudah memiliki tingkat efisiensi penggunaan energi yang sudah tinggi atau belum. Beberapa parameter yang digunakan untuk menilai tingkat efisiensi ini adalah penggunaan energi dan efektivitas. Kedua parameter ini nantinya akan menghasilkan <i>Utilization Rate</i> yang menggambarkan persentase efisiensi penggunaan sistem <i>Cloud Computing</i>. Penelitian ini dilakukan menggunakan data nyata dengan pengaturan universitas.</p> <p>Didapati bahwa <i>Utilization Rate</i> sistem <i>Cloud Computing</i> yang berjalan selama 6 bulan hanya menghasilkan rata-rata 30% hingga 42% saja. Sebuah nilai yang cukup rendah jika dibandingkan dengan saran <i>Utilization Rate</i> yang efisien yaitu di sekitar 70% hingga 80%. <i>Conference Paper</i> ini menyatakan bahwa rendahnya <i>Utilization Rate</i> sistem <i>Cloud Computing</i> ini diakibatkan karena <i>Cloud Service Provider</i> tidak menggunakan kemampuan virtualisasi yang dimiliki oleh <i>Cloud Computing</i> secara maksimal sehingga berakibat kepada pembuangan energi dan tenaga secara sia-sia.</p> <p><i>Conference Paper</i> ini memanggil para <i>Cloud Service Provider</i> dan juga penulis secara tidak langsung agar segera bisa menyelesaikan permasalahan ini. Apabila permasalahan ini tidak segera diselesaikan, akan terjadi sangat banyak pembuangan energi dan tenaga secara sia-sia. <i>Conference Paper</i> ini menjadi tujuan utama penulis ingin membuat sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) agar bisa meningkatkan <i>Utilization Rate</i> sistem <i>Cloud Computing</i>.</p>
4	<i>Survey on Task Scheduling Methods</i> (Henning Titi Ciptaningtyas, 2022)	
	Henning Titi Ciptaningtyas, Ary Mazharuddin Shiddiqi, Diana Purwitasari	<p>Jurnal ini menyediakan berbagai macam parameter yang bisa digunakan untuk mengukur efisiensi sistem penjadwalan tugas dan alokasi mesin virtual (VM). Jurnal ini juga menyediakan berbagai macam definisi mengenai parameter-parameter yang akan digunakan dalam penelitian ini seperti <i>Makespan</i>, <i>Throughput</i>, <i>Resource Utilization</i>. Jurnal ini juga menyediakan rumus dan cara menghitung berbagai parameter diatas. Karena semua alasan ini, penulis memilih menggunakan jurnal ini sebagai referensi dalam penelitian ini.</p>

## 2.2 Dasar Teori

Berikut ini merupakan dasar teori yang akan digunakan oleh penulis dalam penelitian tugas akhir ini. Bagian ini penting untuk memahami apa saja pengertian dasar yang akan digunakan oleh peneliti dalam menjalankan penelitian ini

### 2.2.1 Cloud Computing

*Cloud Computing* adalah ketersediaan sumber daya sistem komputer sesuai permintaan, seperti penyimpanan data dan daya komputasi, tanpa pengelolaan langsung oleh pengguna (Ahmadreza Montazerolghaem, 2020). *Cloud Computing* bergantung pada pembagian sumber daya untuk mencapai koherensi dan biasanya menggunakan model "bayar sesuai penggunaan" yang dapat membantu mengurangi biaya modal, tetapi juga dapat menyebabkan biaya operasional yang tidak terduga bagi pengguna yang tidak sadar (Wray, 2014).

*Cloud Computing* memungkinkan perusahaan untuk menghindari atau meminimalkan biaya infrastruktur di muka. *Cloud Computing* juga memungkinkan perusahaan untuk menjalankan aplikasi mereka lebih cepat, dengan peningkatan pengelolaan dan pemeliharaan yang lebih sedikit, dan memungkinkan perusahaan untuk lebih cepat menyesuaikan sumber daya untuk memenuhi permintaan yang berfluktuasi dan tidak dapat diprediksi (AWS, 2013).

Tujuan *Cloud Computing* adalah untuk memungkinkan pengguna mengambil manfaat dari semua teknologi komputasi terbaru, tanpa perlu pengetahuan mendalam tentang teknologi komputasi tersebut. *Cloud Computing* juga bertujuan untuk memotong biaya dan membantu pengguna fokus pada bisnis inti mereka daripada terhalang oleh hambatan teknologi (Mohammad Hamdaqa, 2012).

Teknologi pendukung utama untuk *Cloud Computing* adalah virtualisasi. Perangkat lunak virtualisasi memisahkan perangkat komputasi fisik menjadi satu atau lebih perangkat "virtual", yang masing-masing dapat dengan mudah digunakan dan dikelola untuk melakukan tugas komputasi. Dengan demikian, sumber daya komputasi yang tidak terpakai dapat dialokasikan dan digunakan secara lebih efisien. Virtualisasi memberikan kelincahan yang diperlukan untuk mempercepat operasi komputasi dan mengurangi biaya dengan meningkatkan pemanfaatan infrastruktur (Mohammad Hamdaqa, 2012).

"Lima Karakteristik Penting" tentang *Cloud Computing* dari Definisi National Institute of Standards and Technology adalah (Technology, 2011):

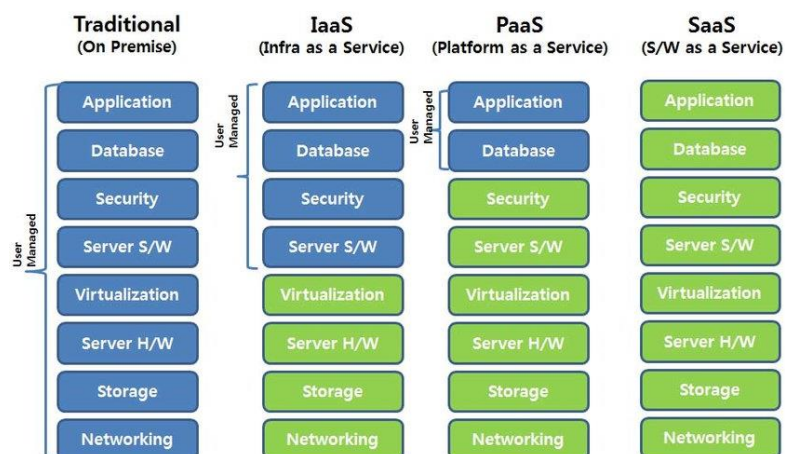
1. *On-demand self-service* dimana konsumen dapat secara langsung menyediakan kemampuan komputasi sesuai kebutuhan secara otomatis tanpa memerlukan interaksi manusia.
2. *Broad network access* dimana kemampuan komputasi tersedia melalui jaringan dan dapat diakses melalui berbagai macam mekanisme (komputer, seluler, server, dll)
3. *Resource Pooling* dimana seluruh sumber daya komputasi milik sebuah *Cloud Service Provider* dikumpulkan untuk melayani banyak konsumen secara sekaligus, dengan sumber daya fisik dan virtual yang berbeda dapat dipindahkan secara dinamis sesuai dengan permintaan konsumen.
4. *Rapid Elasticity* dimana sumber daya komputasi dapat secara elastis disediakan dan dilepaskan sesuai permintaan konsumen
5. *Measured Service* dimana Sistem *Cloud Computing* secara otomatis mengontrol dan mengoptimalkan penggunaan sumber daya dengan memanfaatkan kemampuan pengukuran yang sesuai dengan jenis layanan yang digunakan.

Ada empat model *Cloud Computing* dari Definisi National Institute of Standards and Technology yaitu (Technology, 2011):

1. *Private Cloud* - Infrastruktur *Cloud* yang disediakan secara khusus untuk digunakan oleh satu organisasi yang terdiri dari beberapa unit bisnis. *Private Cloud* dimiliki, dikelola dan dioperasikan oleh satu organisasi, pihak ketiga, atau kombinasi keduanya, dan dapat berada pada satu tempat yang sama ataupun berbeda.
2. *Community Cloud* - Infrastruktur *Cloud* yang disediakan secara khusus untuk digunakan oleh komunitas yang spesifik dari organisasi-organisasi yang memiliki kepentingan bersama. *Community Cloud* dimiliki, dikelola dan dioperasikan oleh satu atau lebih organisasi dalam komunitas tersebut, pihak ketiga, atau kombinasi keduanya, dan dapat berada pada satu tempat yang sama ataupun berbeda.
3. *Public Cloud* - Infrastruktur *Cloud* yang disediakan secara terbuka untuk digunakan oleh masyarakat umum. *Public Cloud* dimiliki, dikelola dan dioperasikan oleh perusahaan, akademis, atau organisasi pemerintah, atau kombinasi dari semuanya. *Public Cloud* berada pada tempat yang ditentukan *Cloud Service Provider*.
4. *Hybrid Cloud* - Infrastruktur *Cloud* yang terdiri dari dua atau lebih infrastruktur *Cloud* yang berbeda (*private*, *community* atau *public*) yang tetap unik, namun terikat pada standar atau paten teknologi yang memungkinkan portabilitas pada data dan aplikasi.

Terdapat tiga macam model *Cloud Computing* utama *Computing* dari Definisi National Institute of Standards and Technology yaitu (Technology, 2011):

1. Infrastructure-as-a-Service (IaaS) dimana perangkat keras komputer disediakan dan dikelola oleh *Cloud Service Provider*. Model ini pada dasarnya ditawarkan kepada mereka yang membutuhkan infrastruktur komputer tetapi tidak ingin berurusan dengan kesulitan mengelolanya.
2. Software-as-a-Service (SaaS) dimana perangkat lunak dilisensikan kepada pelanggan yang membayar. Aplikasi perangkat lunak ini akan *di-host* di Internet, dan pelanggan yang membayar dapat terhubung ke situs *web* tersebut untuk menggunakan perangkat lunak tersebut.
3. Platform-as-a-Service (PaaS) dimana produk model ini adalah platform pengembangan aplikasi berbasis *Cloud*. Model ini memungkinkan pelanggan mengembangkan dan menjalankan aplikasi mereka sendiri tanpa membangun infrastruktur dan menghabiskan uang untuk komponen dan alat yang biasanya mereka perlukan untuk membangun aplikasi mereka.



Gambar 2.1 Model Layanan *Cloud Computing* (Kwangwoog Jung, 2017)



### 2.2.2 Cloud Provisioning

*Cloud provisioning* adalah fitur utama dari model *Cloud Computing*, yang berkaitan dengan cara pelanggan mendapatkan sumber daya *Cloud* dari *Cloud Service Provider* (Montgomery, 2020). Penjadwalan tugas dan alokasi mesin virtual (VM) memainkan peran penting dalam *Cloud provisioning*.

*Cloud Provisioning* memiliki banyak sekali manfaat bagi mereka yang menggunakannya. Manfaat pertama adalah skalabilitas. Dalam model penyediaan kemampuan komputasi tradisional, organisasi memerlukan investasi yang besar dalam menyediakan infrastruktur lokalnya. Hal ini memerlukan persiapan dan perkiraan kebutuhan infrastruktur yang ekstensif, karena infrastruktur lokal sering kali disiapkan untuk bertahan selama beberapa tahun. Namun, menggunakan *Cloud Provisioning*, organisasi dapat dengan mudah meningkatkan dan menurunkan sumber daya *Cloud* mereka.

Kita juga dapat memanfaatkan kecepatan *Cloud Provisioning*. Misalnya, pengembang aplikasi dapat dengan cepat menjalankan serangkaian beban kerja sesuai permintaan, dan dengan demikian menghilangkan kebutuhan akan seorang *administrator* yang menyediakan dan mengelola sumber daya komputasi.

Manfaat lain dari *Cloud Provisioning* adalah potensi penghematan biaya. Sementara penyediaan kemampuan komputasi tradisional dapat menuntut investasi awal yang besar dari suatu organisasi, banyak *Cloud Service Provider* memungkinkan pelanggan membayar hanya apa yang mereka gunakan (Montgomery, 2020).

### 2.2.3 Teknologi Virtualisasi

Virtualisasi adalah teknik bagaimana kita bisa memisahkan layanan dari ketersediaan fisik yang mendasari layanan itu. Virtualisasi adalah proses membuat versi virtual dari sesuatu hal yang fisik seperti perangkat keras komputer. Virtualisasi melibatkan penggunaan perangkat lunak khusus untuk membuat versi sumber daya komputasi menjadi virtual. Dengan bantuan Virtualisasi, beberapa sistem operasi dan aplikasi dapat berjalan pada mesin yang sama dan perangkat keras yang sama pada saat yang sama, meningkatkan pemanfaatan dan fleksibilitas perangkat keras (GeeksForGeeks, 2022).

Dengan kata lain, salah satu teknik utama untuk menghemat biaya, mengurangi perangkat keras, dan menghemat energi yang digunakan oleh *Cloud Service Provider* adalah virtualisasi. Hal ini dikarenakan virtualisasi memungkinkan untuk berbagi satu perangkat fisik di antara banyak pelanggan dan organisasi pada waktu yang sama. Hal ini dilakukan dengan menetapkan sebuah *ID* ke penyimpanan fisik dan memberikan *Pointer* ke sumber daya fisik tersebut sesuai permintaan pengguna. Selain itu, teknologi virtualisasi menyediakan lingkungan virtual tidak hanya untuk menjalankan aplikasi saja tetapi juga untuk penyimpanan, memori, dan jaringan (GeeksForGeeks, 2022).

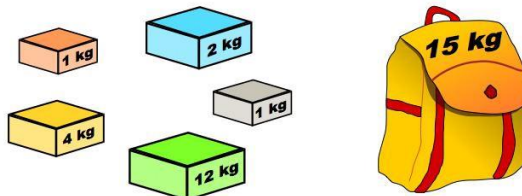
Beberapa manfaat virtualisasi yang sangat penting agar sistem *Cloud Computing* dapat berjalan dengan lancar antara lain (GeeksForGeeks, 2022):

1. Alokasi sumber daya yang lebih fleksibel dan efisien.
2. Meningkatkan produktivitas.
3. Menurunkan biaya infrastruktur teknologi informasi.

4. Akses jarak jauh dan skalabilitas yang cepat.
5. Ketersediaan tinggi dan pemulihan bencana.
6. Sistem pembayaran yang fleksibel
7. Dapat menjalankan banyak sistem operasi dalam satu perangkat fisik.

#### 2.2.4 Knapsack Problem

Tantangan terbesar dalam membangun sebuah sistem penjadwalan tugas dan alokasi mesin virtual (VM) dalam *Cloud Computing* adalah mencari algoritma yang bisa memaksimalkan penggunaan sumber daya *Cloud*. Tantangan ini biasa disebut sebagai “*Knapsack Problem*” dimana “Diberikan sekumpulan benda, masing-masing dengan bobot dan nilai tertentu, maka tentukan jumlah setiap benda untuk dimasukkan kedalam koleksi sehingga bobot totalnya kurang dari atau sama dengan batas yang diberikan dan nilai totalnya sebesar mungkin. (G. B. Mathews, 1896)”.



Gambar 2.2 *Knapsack Problem* (Terh, 2019)

Salah satu contoh dari *Knapsack Problem* adalah dalam penilaian hasil tes di mana peserta tes diberikan pilihan untuk memilih pertanyaan mana yang akan mereka jawab. Untuk jumlah pertanyaan yang sedikit, ini adalah proses yang cukup sederhana. Misalnya, jika ujian berisi 12 pertanyaan yang masing-masing bernilai 10 poin, peserta tes hanya perlu menjawab 10 pertanyaan untuk mencapai skor maksimum 100 poin. Namun, pada tes dengan distribusi nilai poin yang heterogen, akan lebih sulit untuk memperkirakan berapa pertanyaan yang harus dijawab dengan benar. Belum lagi jika jumlah pertanyaan kita perbesar yang mana hal ini akan semakin menambah kompleksitas perkiraan kita (Martin Feuerman, 1973).

*Knapsack Problem* dapat juga kita dapati saat kita berusaha melakukan *Cloud Provisioning* karena kita membutuhkan sebuah sistem yang dapat menjadwalkan tugas dan mengalokasikan mesin virtual (VM) yang tidak dapat dibagi di bawah anggaran tetap dan batasan waktu secara efisien. Variasi dari *Knapsack Problem* yang akan ditemukan pada saat berusaha melakukan *Cloud Provisioning* adalah *Bounded Knapsack Problem* dimana terdapat sejumlah salinan sebanyak  $X_i$  untuk setiap barang yang ada.  $X_i$  dibatasi hingga sebuah angka integer positif. Berikut adalah penulisan dari *Bounded Knapsack Problem* jika diberikan sebuah kumpulan  $n$  benda bernomor 1 hingga  $n$ , masing-masing dengan nilai  $V_i$  dan berat  $w_i$ , dengan kapasitas maksimum  $W$ :

Tabel 2.2 Penulisan *Bounded Knapsack Problem*

No	Masalah	Penulisan
1	<i>Bounded Knapsack Problem</i>	Maksimalkan $\sum_{i=1}^n V_i X_i$ dengan memperhatikan $\sum_{i=1}^n w_i X_i \leq W$ dan $X_i \in (0, 1, 2, 3, \dots, C)$



Hal ini bisa kita lihat pada saat kita ingin melakukan *Cloud Provisioning* dimana sumber daya *Cloud* yang dimiliki oleh sebuah *Cloud Service Provider* pasti lebih dari satu, masing-masing dengan nilai dan biaya mereka sendiri, tetapi tidak mungkin jumlahnya tidak terbatas dan tidak mungkin jumlahnya negatif. Sumber daya *Cloud* ini juga pasti akan dibatasi oleh kemampuan sewa dari pengguna dimana mereka tidak memiliki keuangan yang tidak terbatas untuk menyewa semua Sumber daya *Cloud* yang dimiliki sebuah *Cloud Service Provider*. Oleh karena itu jika kita ingin mencari algoritma yang bisa memaksimalkan penggunaan sumber daya *Cloud*, kita harus berusaha menyelesaikan *Bounded Knapsack Problem* juga.

### 2.2.5 Genetic Algorithm

*Genetic Algorithm* adalah sebuah algoritma yang digunakan untuk menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya *Cloud* dengan mengandalkan operator yang terinspirasi secara biologis seperti seleksi, mutasi, dan penyalangan (Mitchell, 1996). Dalam *Genetic Algorithm*, sebuah populasi dari kandidat solusi (bisa disebut individu, makhluk, organisme, atau fenotipe) untuk sebuah masalah optimasi akan dikembangkan ke arah solusi yang lebih baik. Setiap kandidat solusi memiliki seperangkat sifat (kromosom atau genotipenya) yang dapat dimutasi dan diubah. Sifat ini direpresentasikan dalam biner sebagai string 0 dan 1 (Whitley, 1994).

Proses evolusi ini akan dimulai dari populasi individu yang dihasilkan secara acak, dan merupakan proses berulang, dengan populasi di setiap iterasi yang dihasilkan disebut sebagai generasi. Di setiap generasi, kecocokan setiap individu dalam populasi akan dievaluasi. Kecocokan ini merupakan sebuah nilai dari fungsi tujuan dalam masalah optimasi yang ingin dipecahkan yang disebut sebagai *Fitness Function*. Individu yang lebih cocok akan dipilih secara stokastik dari populasi saat ini, dan genom dari setiap individu akan dimodifikasi (dikombinasikan kembali dan mungkin bermutasi secara acak) untuk membentuk generasi baru. Generasi baru dari kandidat solusi akan digunakan dalam iterasi *Genetic Algorithm* berikutnya. Umumnya, *Genetic Algorithm* akan berakhir ketika jumlah generasi maksimum telah dihasilkan, atau tingkat kecocokan yang memuaskan telah tercapai (Whitley, 1994).

Sebuah *Genetic Algorithm* membutuhkan:

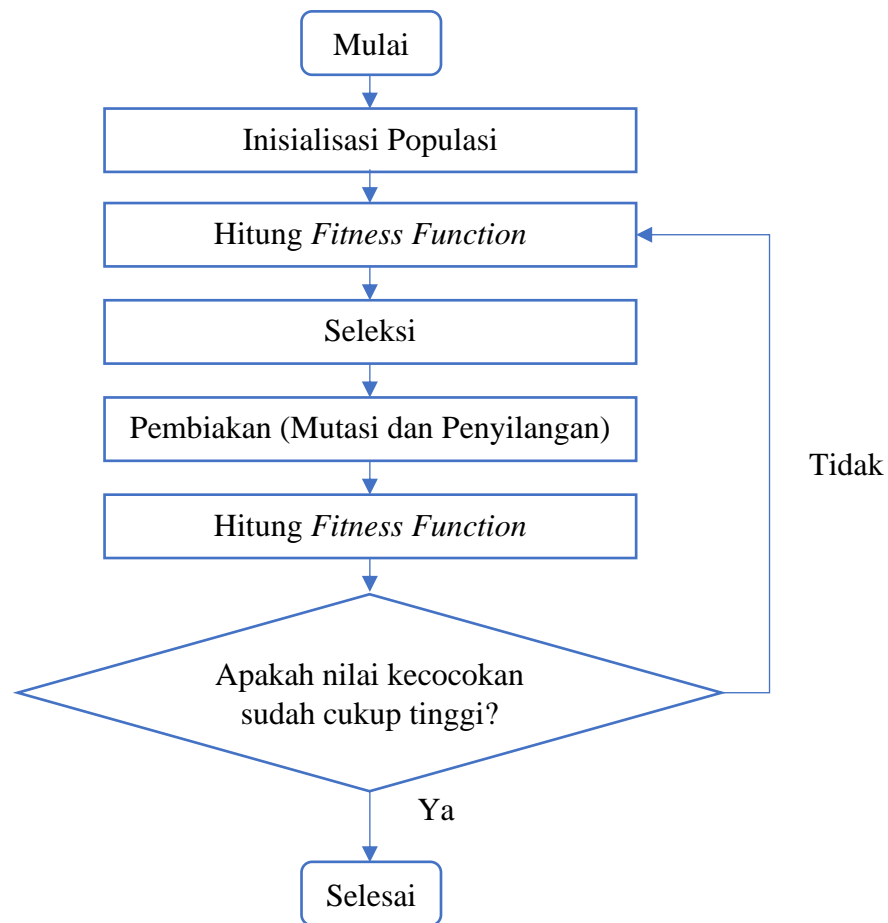
1. Representasi genetik dari kandidat solusi. Representasi ini biasanya dituliskan dalam bentuk *Array of Bits* dengan panjang yang tetap. *Array of Bits* ini akan berisikan bilangan biner 1 dan 0 yang merepresentasikan genetika dari sebuah kandidat solusi (Whitley, 1994).
2. *Fitness Function* untuk menilai kecocokan dari kandidat solusi. Semakin cocok sebuah kandidat solusi dalam masalah optimasi yang ingin dipecahkan maka akan semakin tinggi nilai dari *Fitness Function*-nya (Whitley, 1994).

Proses kerja dari *Genetic Algorithm* terdiri dari beberapa langkah yakni:

1. Inisialisasi dimana populasi awal dari kandidat solusi akan dihasilkan secara acak. Jumlah populasi dari kandidat solusi ini biasanya berjumlah ratusan hingga ribuan pada awalnya namun bisa disesuaikan dengan masalah optimasi yang ingin dipecahkan.
2. Seleksi dimana populasi awal dari kandidat solusi akan dilakukan pemilihan untuk dilakukan pembiakan dan menciptakan generasi baru dari kandidat solusi. Setiap kandidat solusi akan dilakukan seleksi sesuai dengan kecocokan mereka, dinilai dari

*Fitness Function*, dimana semakin cocok mereka dengan masalah optimasi yang ingin dipecahkan maka akan semakin tinggi kemungkinan mereka terpilih untuk dilakukan pembiakan.

3. Pembiakan dimana *Genetic Algorithm* akan menghasilkan generasi berikutnya dari kandidat solusi menggunakan kombinasi dari operasi genetika, seperti mutasi dan penyilangan. Dua kandidat solusi “orangtua” (generasi sebelumnya) akan dipilih dan dilakukan pembiakan untuk menghasilkan kandidat solusi “anak” (generasi berikutnya). Pasangan orangtua-orangtua yang baru akan dilakukan pemilihan untuk menghasilkan kombinasi anak yang baru hingga tercapai populasi generasi yang baru.
4. Iterasi yang berarti akan dilakukan proses Seleksi dan Pembiakan generasi yang baru lagi. Dengan melakukan hal ini, kandidat solusi “anak” yang dihasilkan pada setiap generasi baru akan memiliki karakteristik dari orangtua mereka. Berkat dilakukan proses Seleksi, maka dipastikan kandidat solusi “orangtua” yang terpilih akan memiliki nilai kecocokan yang tinggi sesuai dengan *Fitness Function* yang diinginkan. Hal ini akan berakibat meningkatnya nilai kecocokan rata-rata dari setiap generasi baru yang dihasilkan melalui proses Seleksi dan Pembiakan di tiap iterasi.
5. Terminasi dimana ketika sebuah nilai kecocokan dari kandidat solusi dinilai sudah cukup tinggi, maka proses iterasi *Genetic Algorithm* akan dilakukan penghentian. Kandidat solusi yang memiliki nilai kecocokan tinggi tersebut akan dijadikan solusi dari masalah optimasi yang ingin dipecahkan.



Gambar 2.3 Flowchart Cara Kerja *Genetic Algorithm*

### 2.2.6 Alasan Pemilihan Genetic Algorithm

*Genetic Algorithm* dipilih untuk bisa menyelesaikan *Knapsack Problem* yang ditemui pada saat ingin melakukan *Cloud Provisioning* karena beberapa alasan, yaitu:

1. *Genetic Algorithm* sering digunakan untuk menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya di mana pengambil keputusan harus memilih dari serangkaian tugas yang tidak dapat dibagi di bawah anggaran tetap atau batasan waktu (Mitchell, 1996).
2. Pengkodean genetik pada *Genetic Algorithm* berbasis biner 1 dan 0 yang sangat cocok digunakan untuk merepresentasikan jumlah barang pada *Knapsack Problem* yang juga dikodekan dengan basis biner 1 dan 0.
3. *Genetic Algorithm* memiliki prosedur penilaian kecocokan kandidat solusi menggunakan *Fitness Function* yang bisa dimodifikasi sesuai dengan keperluan secara fleksibel dan bisa disesuaikan dengan permintaan pengguna saat melakukan *Cloud Provisioning*.
4. Banyak iterasi dari *Genetic Algorithm* bisa disesuaikan dengan kebutuhan untuk mencapai nilai kecocokan yang diinginkan untuk optimasi penggunaan sumber daya *Cloud*.
5. Proses pembiakan dari *Genetic Algorithm* bisa menggunakan berbagai macam operator biologis yang bisa disesuaikan dengan data dan kebutuhan kita (mutasi, penyilangan, kolonisasi, kepunahan, dll).

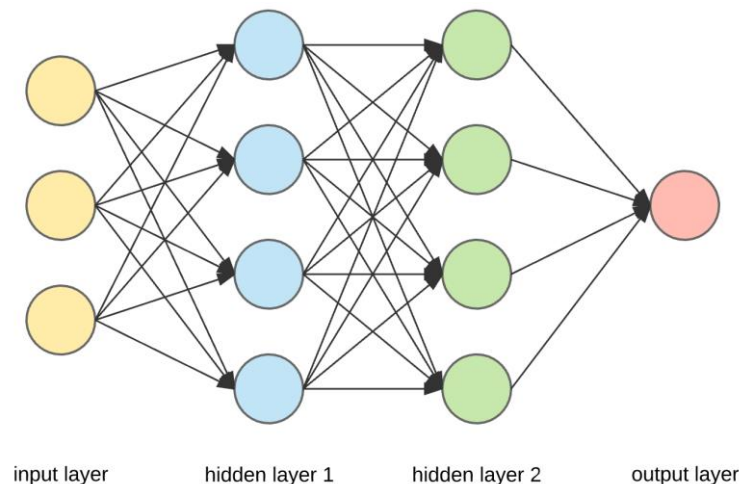
### 2.2.7 Artificial Neural Network

*Artificial Neural Network* merupakan kumpulan unit atau simpul yang saling terhubung dan didasarkan pada model neuron-neuron yang ada di dalam otak biologis. Kumpulan unit atau simpul ini disebut sebagai neuron buatan. Neuron-neuron buatan ini dapat menerima, memproses dan meneruskan sinyal pada neuron buatan lain yang terhubung dengannya. "Sinyal" ini adalah bilangan real, dan *output* dari setiap neuron buatan dihitung menggunakan berbagai fungsi non-linier dari jumlah *input* yang diterimanya (Hardesty, 2017).

Sambungan antar setiap neuron buatan disebut juga sebagai *edge*. Setiap neuron dan *edge* memiliki bobot yang dapat menyesuaikan diri pada saat proses pembelajaran berlangsung. Bobot ini dapat menambah atau mengurangi kekuatan sinyal pada setiap sambungan antar neuron buatan. Neuron-neuron buatan ini dikumpulkan menjadi beberapa lapisan yang terbagi menjadi *Input Layer* (lapisan masukan), *Hidden Layer* (lapisan tersembunyi karena berada di tengah), dan *Output Layer* (lapisan keluaran). Lapisan yang berbeda dapat melakukan perubahan yang berbeda pada *input*-nya tergantung dari fungsi non-linier yang diterapkan (Hardesty, 2017).

*Artificial Neural Network* belajar (atau dilatih) dengan memproses data, yang masing-masing berisi *input* dan *output* yang diketahui, sehingga membentuk asosiasi bobot dan probabilitas diantara keduanya. Asosiasi ini kemudian disimpan dalam struktur data *Artificial Neural Network* itu sendiri. Pelatihan *Artificial Neural Network* dilakukan dengan menentukan perbedaan antara *output* yang diprediksi dan *output target*. Perbedaan ini disebut sebagai kesalahan yang mana ingin diminimalisir melalui proses pembelajaran ini. *Artificial Neural*

*Network* kemudian menyesuaikan bobotnya berdasarkan nilai kesalahan ini. Penyesuaian berturut-turut akan menyebabkan *Artificial Neural Network* menghasilkan *output* yang semakin mirip dengan *output target*. Setelah penyesuaian dalam jumlah yang dirasa cukup, pelatihanpun dapat dihentikan dan *Artificial Neural Network* dianggap bisa memberikan prediksi yang tepat (Hardesty, 2017).

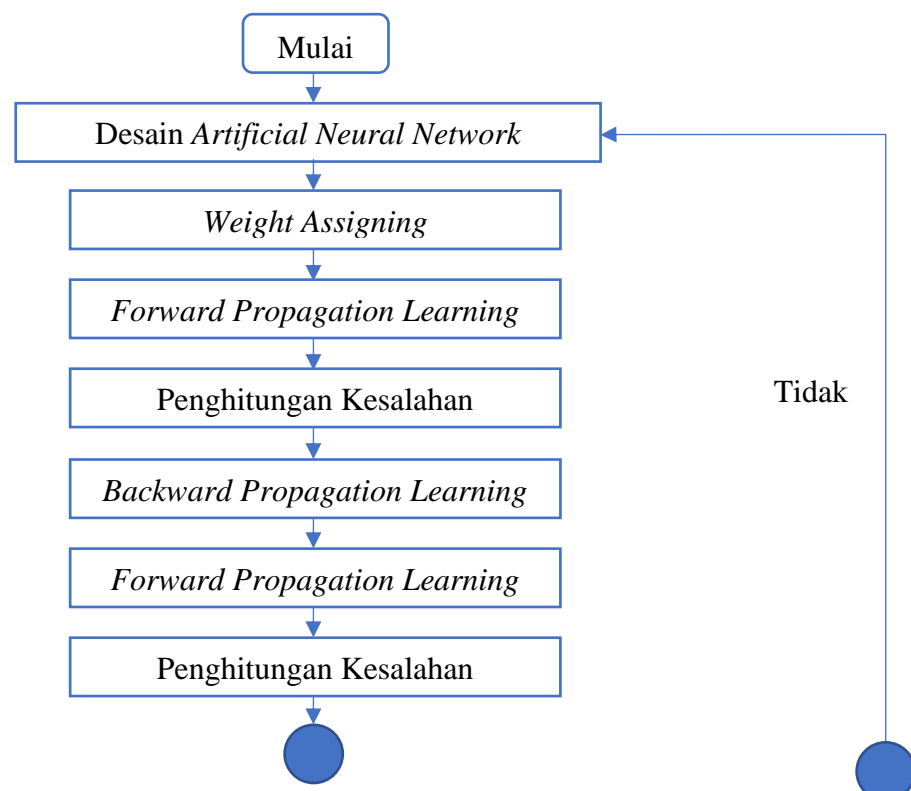


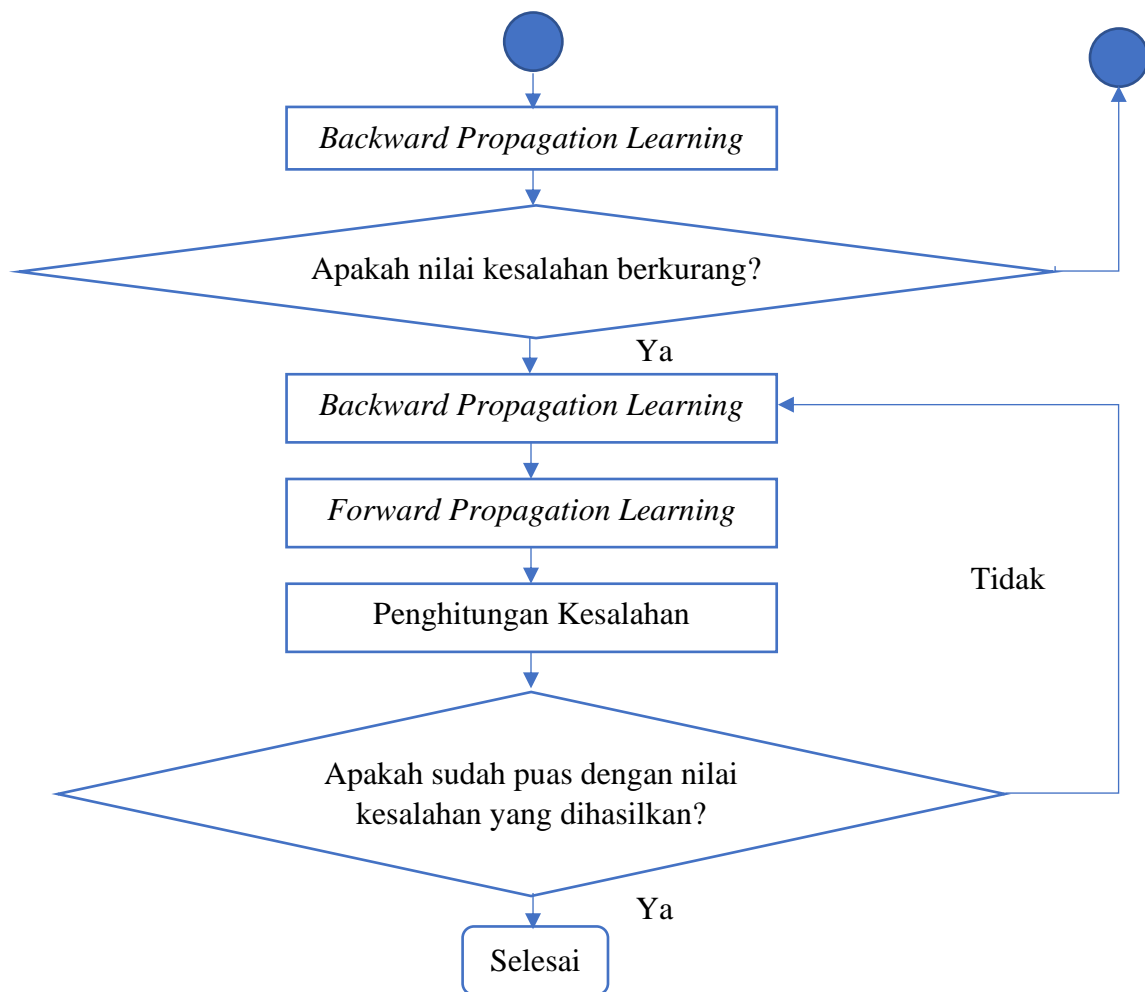
Gambar 2.4 *Artificial Neural Network* dengan 4 Lapisan (Larasati, 2019)

Proses kerja dari *Artificial Neural Network* terdiri dari beberapa langkah yakni:

1. Desain dimana kita mendesain *Artificial Neural Network* yang sesuai untuk digunakan dalam pemecahan masalah kita. Kita menentukan ada berapa banyak lapisan *Input Layer* *Hidden Layer* dan *Output Layer* yang dibutuhkan serta ada berapa banyak neuron buatan untuk tiap lapisan tersebut. Kita bisa mencari tahu berapa banyak neuron buatan yang diperlukan untuk *Input Layer* dengan cara mencari tahu berapa banyak masukan data yang ingin dipelajari oleh *Artificial Neural Network*. Untuk lapisan *Output Layer*, dengan konsep yang sama, perlu kita cari tahu berapa banyak keluaran yang ingin dihasilkan oleh *Artificial Neural Network*. Untuk *Hidden Layer*, jumlahnya kita disesuaikan sesuai dengan perjalanan sinyal dari *Input Layer* menuju *Output Layer*.
2. *Weight Assigning* dimana kita memberikan bobot pada setiap Setiap neuron dan *edge* yang ada di dalam *Artificial Neural Network*. Hal ini kita lakukan secara acak terlebih dahulu karena kita pasti tidak akan tahu seberapa besar bobot yang harus kita masukkan pada awalnya tanpa melakukan proses pembelajaran terlebih dahulu. Jika kita sudah tahu harus memasukkan berapa bobot pada tiap neuron dan *edge* yang ada, maka sejatinya kita tidak perlu menggunakan *Artificial Neural Network*.
3. *Forward Propagation Learning* dimana ini adalah proses adaptasi *Artificial Neural Network* untuk bisa menghasilkan prediksi yang lebih baik dengan mempertimbangkan sampel data yang sudah ada. Disebut sebagai *Forward Propagation* karena pada proses ini sinyal akan berjalan ke “depan” dari *Input Layer* menuju *Output Layer*. Bobot dari tiap neuron dan *edge* akan dilakukan kalkulasi menggunakan sebuah fungsi non-linier mulai dari *Input Layer* menuju *Output Layer*.
4. Penghitungan Kesalahan dimana pada saat nilai terakhir sampai kepada *Output Layer*, nilai tersebut akan dilakukan kalkulasi perbedaan antara nilai tersebut dengan *output*

- target* yang benar sesuai dengan data yang dipelajari. Perbedaan ini disebut dengan kesalahan dan merupakan bagian penting dari pembelajaran *Artificial Neural Network*.
5. *Backward Propagation Learning* dimana nilai kesalahan di tarik ke “belakang” (dari *Output Layer* menuju *Input Layer*) untuk menyesuaikan bobot yang ada pada tiap neuron dan *edge*. Hal ini dilakukan agar *Artificial Neural Network* bisa belajar dari kesalahan tersebut dan bisa menyesuaikan bobot di tiap neuron dan *edge* sehingga bisa memberikan prediksi yang lebih baik.
  6. Iterasi dimana dilakukan kembali *Forward Propagation Learning*, Penghitungan Kesalahan, dan *Backward Propagation Learning*. Hal ini dilakukan agar pada tiap iterasi, *Artificial Neural Network* dapat semakin baik menyesuaikan bobot dari tiap neuron dan *edge*-nya sehingga bisa semakin memperkecil perbedaan antara *output* yang diprediksi dan *output target* (memperkecil nilai kesalahan).
  7. Evaluasi dimana kita melihat apakah nilai kesalahan tiap iterasi dari *Forward Propagation Learning*, Penghitungan Kesalahan, dan *Backward Propagation Learning* memang memperkecil nilai kesalahan atau tidak. Jika nilai kesalahan tidak diperkecil dari tiap iterasi, maka kita harus melakukan desain ulang dari *Artificial Neural Network* dan mengulang kembali proses belajar.
  8. Terminasi dimana ketika kita mendapati bahwa nilai kesalahan memang semakin mengecil setiap kali iterasi dari *Forward Propagation Learning*, Penghitungan Kesalahan, dan *Backward Propagation Learning* dan kita sudah puas dengan perbedaan nilai kesalahan yang dihasilkan. Perlu diperhatikan bahwa nilai kesalahan dari *Artificial Neural Network* hampir tidak mungkin bernilai 0 sehingga kita harus melakukan proses terminasi secara manual ketika dirasa pengurangan nilai kesalahan sudah tidak terlalu signifikan jika dibandingkan dengan usaha yang harus kita lakukan saat melakukan pembelajaran. Nantinya *Artificial Neural Network* akan menghasilkan model yang bisa kita gunakan secara langsung untuk pemrosesan data.





Gambar 2.5 Flowchart Cara Kerja Artificial Neural Network

### 2.2.8 Alasan Pemilihan Artificial Neural Network

*Artificial Neural Network* dipilih untuk bisa menyelesaikan *Knapsack Problem* yang ditemui pada saat ingin melakukan *Cloud Provisioning* karena beberapa alasan, yaitu:

1. *Artificial Neural Network* dapat mempelajari, memproses, dan memprediksi hasil dari sebuah data. Dikarenakan penelitian ini akan menggunakan dataset, *Artificial Neural Network* merupakan pilihan yang cocok untuk digunakan.
2. *Artificial Neural Network* dapat menyesuaikan bobotnya berdasarkan nilai kesalahan yang dihasilkan dari tiap iterasi prosesnya. Hal ini akan sangat berguna ketika kita ingin menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya *Cloud*.
3. *Artificial Neural Network* akan dipadukan bersama dengan *Genetic Algorithm*. Dimana *Artificial Neural Network* akan menjadi otak yang mempelajari, memproses, dan memprediksi hasil dari dataset yang ada dan melakukan *Cloud Provisioning* berdasarkan data tersebut. Sedangkan *Genetic Algorithm* akan digunakan untuk mengembangkan otak tersebut dan memastikan hanya *Artificial Neural Network* yang memiliki nilai *Fitness Function* yang tinggi yang akan berkembang menjadi generasi berikutnya. Hal ini akan berakibat semakin bertambah tingginya solusi untuk optimasi penggunaan sumber daya *Cloud* pada tiap generasi baru (Suryansh, 2018).

4. Banyak iterasi dari *Artificial Neural Network* bisa disesuaikan dengan kebutuhan untuk mencapai nilai kesalahan minimal yang diinginkan untuk optimasi penggunaan sumber daya *Cloud*.
5. Karena kemampuannya untuk mereproduksi dan memodelkan banyak data, *Artificial Neural Network* telah diaplikasikan di banyak disiplin ilmu, termasuk pada optimasi penggunaan sumber daya.

### 2.2.9 CloudSim

*CloudSim* adalah kerangka kerja *Open Source*, yang digunakan untuk mensimulasikan infrastruktur dan layanan *Cloud Computing*. *CloudSim* dikembangkan oleh organisasi bernama CLOUDS Lab dan ditulis seluruhnya dalam bahasa *Java*. *CloudSim* digunakan sebagai sarana untuk mengevaluasi hipotesis sebelum pengembangan perangkat lunak sesungguhnya dengan mereproduksi tes dan hasilnya dalam sebuah simulasi lingkungan *Cloud Computing* (Laboratory, 2018).

Ambil contoh jika kita ingin menerapkan aplikasi atau sebuah situs web di dalam *Cloud*. Kita pasti ingin menguji layanan dan muatan yang dapat ditangani oleh produk tersebut. Kita juga ingin menyesuaikan kinerjanya agar dapat mengatasi kemacetan sebelum penerapan sesungguhnya. Kita dapat melakukan evaluasi tersebut melalui pengkodean simulasi lingkungan *Cloud Computing* dengan bantuan berbagai kelas fleksibel dan dapat diskalakan yang disediakan oleh *CloudSim* secara gratis (Laboratory, 2018).

Beberapa fitur penting yang dimiliki oleh *CloudSim* adalah:

1. Pusat data, server, dan host yang tervirtualisasi dalam skala besar.
2. Kebijakan yang dapat disesuaikan untuk melakukan *Cloud Provisioning*.
3. Sumber daya *Cloud Computing* yang dapat menghitung penggunaan energi.
4. Dilengkapi dengan topologi jaringan pusat data dan aplikasi pengiriman pesan.
5. Bisa memasukkan Sumber daya *Cloud* secara dinamis ke dalam simulasi.
6. Kebijakan *Cloud Provisioning* yang dapat ditentukan oleh pengguna.



Gambar 2.6 Melbourne CLOUDS Lab Pengembang *CloudSIM* (Laboratory, 2018)

### 2.2.10 Library CloudSim

*CloudSim* adalah kerangka kerja *Open Source* yang ditulis menggunakan bahasa pemrograman *Java* dan kelasnya terstruktur dengan cara yang sangat spesifik. Sehingga sangat penting bagi kita untuk bisa memahami bagaimana arsitektur *Cloudsim* dibagi menjadi beberapa paket dan kelas yang memfasilitasi simulasi *Cloud* menggunakan *CloudSim* (SinghAnupinder, 2019).



Ada 12 *Namespace* dalam arsitektur *CloudSim* dimana setiap *Namespace* memiliki satu set kelas dengan fungsi spesifik yang berkorelasi dengan cara simulasi *Cloud* dijalankan. Ke-12 *Namespace* tersebut adalah (SinghAnupinder, 2019):

Tabel 2.3 *Namespace CloudSim*

No	Namespace	Fungsi
1	<i>Org.cloudbus.cloudsim</i>	<i>Namespace</i> ini berisi kelas model dari berbagai komponen perangkat keras dasar, kelompoknya, dan metode alokasi atau penggunaannya di <i>CloudSim</i> . Di sini, model berarti bahwa kelas ini berisi implementasi berbagai atribut dan perilaku komponen perangkat keras sistem <i>Cloud Computing</i> di dalam kehidupan nyata sebagai kumpulan metode <i>Java</i> . Setelah kelas-kelas ini dimulai selama simulasi, mereka akan mensimulasikan perilaku komponen sistem <i>Cloud Computing</i> nyata.
2	<i>Org.cloudbus.cloudsim.core</i>	<i>Namespace</i> ini berisi implementasi mesin simulasi, di mana kelas <i>cloudsim.java</i> yang ada di dalamnya adalah kelas utama dan bertanggung jawab untuk memulai dan menghentikan proses simulasi.  Selain itu, terdapat kelas <i>simentity.java</i> yang berfungsi untuk mempertahankan status komponen <i>Cloud</i> yang disimulasikan.  Terakhir, terdapat juga kelas <i>Simevent.java</i> , <i>futurequeue.java</i> , dan <i>defferedqueue.java</i> yang bertanggung jawab untuk memelihara panggilan peristiwa terkait komponen <i>Cloud</i> selama proses simulasi.
3	<i>Org.cloudbus.cloudsim.core.predicates</i>	<i>Namespace</i> ini berisi kelas-kelas yang terkait dengan pemanggilan peristiwa dari <i>defferedqueue.java</i> agar bisa diproses selama pemrosesan event simulasi.
4	<i>Org.cloudbus.cloudsim.lists</i>	<i>Namespace</i> ini berisi implementasi daftar yang akan digunakan secara global selama proses simulasi. <i>Namespace</i> ini adalah kumpulan kelas khusus, di mana operasi penyortiran dan perbandingan diimplementasikan. Ada kelas daftar



No	Namespace	Fungsi
		untuk host, elemen pemrosesan, tugas, mesin virtual, dan sumber daya.
5	<i>Org.cloudbus.cloudsim.power</i>	<i>Namespace</i> ini berisi implementasi komponen sistem <i>Cloud Computing</i> yang diperluas, yang dapat mensimulasikan skenario komputasi hijau ( <i>Green Computing</i> ) atau sadar daya ( <i>Power Aware</i> ).
6	<i>Org.cloudbus.cloudsim.power.lists</i>	<i>Namespace</i> ini hanya berisi implementasi <i>powervmList.java</i> , yang merupakan versi lanjutan dari class <i>VmList.java</i> dari <i>namespace org.cloudbus.cloudsim.list</i>
7	<i>Org.cloudbus.cloudsim.power.models</i>	<i>Namespace</i> ini berisi kumpulan kelas yang menentukan konfigurasi daya server sebagai kelas model. Kelas-kelas ini meniru cara kerja sebenarnya dari berbagai merek mesin server yang tersedia di pasar dan membantu dalam menentukan konsumsi daya oleh mesin tersebut selama proses simulasi.
8	<i>Org.cloudbus.cloudsim.provisioners</i>	<i>Namespace</i> ini berisi implementasi perilaku tentang bagaimana komponen sistem <i>Cloud Computing</i> dapat disediakan untuk memenuhi permintaan sumber daya virtual.
9	<i>Org.cloudbus.cloudsim.util</i>	<i>Namespace</i> ini berisi implementasi fungsi perhitungan penting dan kumpulan utilitas umum yang digunakan oleh <i>Namespace</i> lain.

### 2.2.11 Alasan Pemilihan CloudSim

*CloudSim* dipilih untuk bisa menyediakan simulasi lingkungan *Cloud Computing* untuk dilakukan penelitian *Cloud Provisioning* karena beberapa alasan, yaitu:

1. Open source dan bebas biaya, sehingga menguntungkan pengguna.
2. Mudah untuk diunduh dan diatur.
3. Lebih ramah ke pengguna dan dapat diperluas untuk mendukung pemodelan dan eksperimen.
4. Tidak memerlukan komputer dengan spesifikasi tinggi untuk bekerja.
5. Menyediakan kebijakan *Cloud Provisioning* yang telah ditentukan sebelumnya untuk mengelola sumber daya, dan juga memungkinkan penerapan algoritma yang ditentukan oleh pengguna.
6. Dokumentasi yang lengkap memberikan contoh yang telah dikodekan sebelumnya agar pengguna baru bisa terbiasa dengan berbagai fungsi dasar.

### 2.2.12 Library Encog

*Encog* adalah *framework* pembelajaran mesin yang tersedia untuk bahasa *Java* dan *.Net*. *Encog* mendukung berbagai algoritma pembelajaran mesin seperti *Bayesian Networks*, *Hidden Markov Models*, dan *Support Vector Machines*. Namun, kekuatan utamanya terletak pada algoritma *Artificial Neural Network*-nya. *Encog* berisi berbagai macam kelas untuk membuat berbagai macam *Artificial Neural Network*, serta memiliki kelas untuk normalisasi dan pemrosesan data untuk *Artificial Neural Network* ini (Heaton, *Encog: Library of Interchangeable Machine Learning Models*, 2015).

*Encog* memungkinkan pengguna *Java* untuk bereksperimen dengan berbagai macam model bahasa mesin menggunakan antarmuka yang sederhana dan konsisten untuk pengelompokan, regresi, dan klasifikasi. Hal ini memungkinkan pengguna untuk menemukan model mana yang paling cocok untuk data yang mereka gunakan. *Encog* menyediakan alat dasar untuk pembuatan model dan pengguna bisa langsung membuat model yang mereka inginkan berdasarkan spesifikasi yang sudah ditentukan sebelumnya (Heaton, *Encog: Library of Interchangeable Machine Learning Models*, 2015).



Gambar 2.7 Logo *Encog* (Heaton, *Encog Machine Learning Framework*, 2022)

### 2.2.13 Alasan Pemilihan Encog

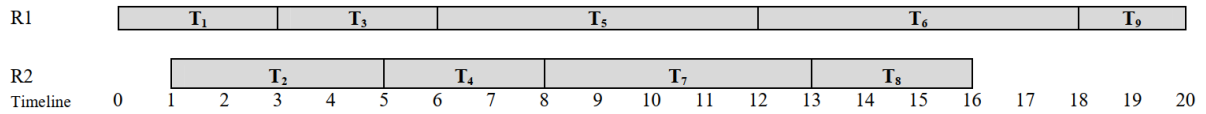
*Encog* dipilih untuk melakukan pengkodean *Artificial Neural Network* untuk dilakukan penelitian dalam *Cloud Provisioning* ini karena beberapa alasan, yaitu:

1. Semua model *Encog* diimplementasikan menggunakan algoritma *Multithreading* yang efisien, sehingga bisa mengurangi waktu pemrosesan data (Heaton, *Encog: Library of Interchangeable Machine Learning Models*, 2015).
2. *Encog* memiliki banyak fungsi pelatihan yang bisa dipilih untuk memaksimalkan akurasi yang dimiliki oleh model sehingga tidak terjadi *Overfitting* dan *Underfitting*.
3. *Encog* memiliki kelas bawaan untuk normalisasi data sehingga tidak perlu menggunakan *Library* lain untuk melakukan normalisasi.
4. Dokumentasi yang lengkap memberikan contoh yang telah dikodekan sebelumnya agar pengguna baru bisa terbiasa dengan berbagai fungsi dasar.

### 2.2.14 Definisi Parameter Penggunaan Sumber Daya Cloud

Untuk bisa mengetahui perbandingan efisiensi antara satu algoritma penjadwalan tugas dan alokasi mesin virtual (VM) dengan yang lainnya dalam meningkatkan tingkat penggunaan sumber daya *Cloud*, maka kita perlu terlebih dahulu tahu parameter apa saja yang akan digunakan untuk bisa mengukurnya serta definisi dan cara menghitung parameter tersebut.

Berikut adalah parameter yang akan digunakan dalam mencari tahu tingkat penggunaan sumber daya *Cloud* (contoh akan dihitung berdasarkan dari gambar 2.7) (Henning Titi Ciptaningtyas, 2022):



Gambar 2.8 Contoh Penjadwalan Tugas (Henning Titi Ciptaningtyas, 2022)

Tabel 2.4 Parameter Tingkat Penggunaan Sumber Daya *Cloud*

Nama	Definisi	Cara Menghitung
<i>Makespan</i>	Waktu penyelesaian tugas terakhir (Henning Titi Ciptaningtyas, 2022).	$\max_{i \in tasks} \{F_i\}$ Contoh : 20 unit waktu karena T9 yang merupakan tugas terakhir selesai di unit ke-20
<i>Average Start Time</i>	Waktu rata-rata tugas mulai dieksekusi (Pradeep Singh Rawat, 2020).	$\frac{\sum_{i=1}^n Waktu Ri Mulai}{nR}$ Contoh: $(0+1)/2 = 0.5$
<i>Average Finish Time</i>	Waktu rata-rata tugas selesai dieksekusi (Pradeep Singh Rawat, 2020).	$\frac{\sum_{i=1}^n Waktu Ri Selesai}{nR}$ Contoh: $(20+16)/2 = 18$
<i>Average Execution Time</i>	Rata-rata lama waktu yang dibutuhkan untuk tugas selesai dieksekusi (Pradeep Singh Rawat, 2020).	$\frac{\sum_{i=1}^n Waktu Ti}{nT}$ Contoh: $(3+3+6+6+2+4+3+5+3)/9 = 32.3$
<i>Scheduling Time</i>	Lama waktu yang dibutuhkan untuk tugas pertama mulai dieksekusi (Pradeep Singh Rawat, 2020).	Contoh : Dari gambar kita ambil bahwa <i>Scheduling Time</i> adalah 0, karena tidak ada delay untuk penjadwalan tugas
<i>Scheduling Length</i>	Jumlah total waktu yang dibutuhkan dari awal simulasi dimulai hingga simulasi selesai (Farouk A. Emara, 2021).	$Scheduling Time + Makespan$ Contoh: $0+20 = 20$
<i>Throughput</i>	Jumlah total tugas yang menyelesaikan eksekusi per unit waktu (Henning Titi Ciptaningtyas, 2022).	$\frac{nT}{Makespan}$ Contoh: $9/20 = 0.45$
<i>Resource Utilization</i>	Persentase penggunaan sumber daya saat pengerjaan tugas (Henning Titi Ciptaningtyas, 2022).	$\frac{\sum_{i=1}^n Waktu Ri Selesai}{Makespan * nR}$ Contoh: $(20+15) / (20*2) = 0.875$

Nama	Definisi	Cara Menghitung
<i>Energy Consumption</i>	Jumlah total penggunaan energi saat pengerjaan tugas (Farouk A. Emara, 2021).	
<i>Imbalance Degree</i>	Pengukuran ketidakseimbangan diantara semua pusat data (Farouk A. Emara, 2021).	$\frac{ET_{max} - ET_{min}}{ET_{avg}}$ <p>Dimana <math>ET_{max}</math>, <math>ET_{min}</math>, <math>ET_{avg}</math> adalah <i>Execution Time</i> maksimum, minimum, dan rata-rata secara urut dari semua pusat data yang ada.</p> <p>Contoh: <math>(6-2)/32.3 = 0.12</math></p>

### 2.2.15 Tingkatan Penggunaan Sumber Daya

Tingkatan pemanfaatan sumber daya membantu kita untuk memahami bagaimana sistem kita menghabiskan waktu dan tenaga untuk bisa menyelesaikan tugas yang kita minta, sehingga kita dapat membuat keputusan manajemen sumber daya yang lebih efisien yang dapat memaksimalkan produktivitas dan profitabilitas (Meier, 2020). Hal ini dikarenakan pemanfaatan yang berlebihan (misalnya, bekerja lebih dari waktu dan tenaga yang tersedia) dapat menyebabkan penurunan performa sistem kita. Sedangkan pemanfaatan yang kurang (misalnya, bekerja kurang dari waktu dan tenaga yang tersedia) dapat menyebabkan penundaan penyelesaian tugas (Education, 2021). Lalu berapakah persentase tingkatan pemanfaatan sumber daya yang direkomendasikan dan dianggap paling efisien? Menurut analisis *Gartner*, jawabannya adalah 70% hingga 80% (Moore, 2019).

### 2.2.16 Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon Log

Dataset *San Diego Supercomputer Center (SDSC) Blue Horizon Log* adalah dataset yang akan digunakan untuk mensimulasikan permintaan sumber daya ke dalam simulasi sistem *Cloud Computing* dalam penelitian ini. *San Diego Supercomputer Center (SDSC) Blue Horizon Log* adalah sebuah catatan ekstensif yang dimulai saat mesin *Cloud Computing* baru saja dipasang di lab *San Diego Supercomputer Center (SDSC)*, dan kemudian mencatat catatan penggunaan mesin tersebut selama lebih dari dua tahun penggunaan produksi. Catatan ini berisi informasi tentang waktu saat *Node* diminta, waktu saat *Node* digunakan, waktu *CPU*, waktu pengiriman, waktu tunggu, waktu jalan, dan informasi tentang penggunaannya (log, 2003).

Ada 144 *Node* di dalam mesin *Cloud Computing* lab *San Diego Supercomputer Center (SDSC)*. Masing-masing adalah *SMP* 8 arah dengan palang yang menghubungkan prosesor ke memori bersama. Catatan ini mencatat penggunaan mesin tersebut mulai dari April 2000 hingga Januari 2003. Sistem penjadwalan tugas yang digunakan pada mesin ini disebut *Catalina*. Sistem ini dikembangkan di lab *San Diego Supercomputer Center (SDSC)*. Sistem ini menggunakan antrian prioritas, melakukan *Backfilling*, dan mendukung penggunaan reservasi (log, 2003).

### 2.2.17 Eclipse IDE

*Eclipse* adalah lingkungan pengembangan terintegrasi (IDE) yang digunakan dalam pemrograman komputer. *Eclipse* berisi ruang kerja dasar dan memiliki sistem *plug-in* yang

dapat diperluas untuk menyesuaikan lingkungan pemrograman komputer. *Eclipse* adalah IDE paling populer kedua untuk pengembangan *Java*, dan, hingga 2016, adalah yang paling populer. *Eclipse* sebagian besar ditulis dalam bahasa *Java* dan penggunaan utamanya adalah untuk mengembangkan aplikasi berbasis *Java* (Foundation, 2001).

*Eclipse* memiliki peralatan pengembangan perangkat lunak (SDK) yang dimaksudkan untuk para pengembang *Java* namun tidak terbatas pada Bahasa *Java* saja. Pengguna dapat memperluas kemampuan *Eclipse* dengan memasang berbagai macam *plug-in* yang ditulis untuk *Eclipse Platform*, seperti peralatan pengembangan lunak untuk bahasa pemrograman lain. Pengguna bahkan dapat menulis dan menyumbangkan modul *plug-in* mereka sendiri. Peralatan pengembangan perangkat lunak *Eclipse* ini adalah perangkat lunak *open source* dan gratis sehingga bebas untuk digunakan tanpa perlu mengeluarkan biaya apapun (Foundation, 2001).



Gambar 2.9 Logo *Eclipse IDE* (Foundation, 2001)

## **BAB III METODOLOGI**

### **3.1 Tahapan Metodologi Penelitian**

Pada sub bab ini akan dibahas mengenai metodologi yang akan digunakan dalam pengerjaan tugas akhir. Metodologi yang digunakan dalam penelitian ini terdiri dari enam bagian antara lain:

1. Identifikasi Permasalahan

Pada bagian ini akan dilakukan pemahaman empatik atau pengidentifikasian masalah yang ingin dipecahkan.

2. Studi Literatur

Pada bagian ini akan dilakukan penentuan masalah inti dengan cara mencari referensi dari beberapa jurnal penelitian ataupun konferensi yang pernah dilakukan sebelumnya.

3. Analisis dan Desain Perangkat Lunak

Pada bagian ini akan ditetapkan ide yang diusulkan dalam bentuk desain purwarupa.

4. Implementasi Perangkat Lunak

Pada bagian ini akan dilakukan visualisasi dari ide yang diusulkan atau contoh dari purwarupa sistem penjadwalan tugas dan alokasi mesin virtual (VM).

5. Pengujian dan Evaluasi

Pada bagian ini dilakukan pengujian purwarupa berupa pengukuran parameter efisiensi sistem penjadwalan tugas dan alokasi mesin virtual (VM).

6. Penyusunan Buku Tugas Akhir

Pada Bagian ini akan dilakukan penyusunan buku tugas akhir yang merupakan laporan secara lengkap mengenai tugas akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut

### **3.2 Deskripsi Metodologi Penelitian**

Berikut ini merupakan tahapan metode yang akan digunakan oleh penulis dalam penelitian tugas akhir ini. Bagian ini penting untuk memahami apa saja tahapan dan proses yang akan dilakukan oleh peneliti dalam menjalankan penelitian ini.

#### **3.2.1 Identifikasi Permasalahan**

Pada tahap ini akan dilakukan kegiatan mencari permasalahan yang ada di sekitar penulis ataupun terinspirasi dari jurnal penelitian. Setelah permasalahan ditemukan, langkah selanjutnya adalah mencari solusi yang dapat digunakan untuk menyelesaikan permasalahan tersebut. Hasil dari tahap ini merupakan permasalahan dan usulan solusi yang dapat diangkat menjadi topik tugas akhir.

### 3.2.2 Studi Literatur

Tugas akhir ini menggunakan beberapa literatur yang sudah pernah dibuat sebelumnya seperti “*Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment*” (Farouk A. Emara, 2021), “*Resource provisioning in scalable cloud using bio-inspired artificial neural network model*” (Pradeep Singh RawatPriti, 2020), “*A Call for Energy Efficiency in Data Centers*” (Michael Pawlish A. S., 2014), dan “*Survey on Task Scheduling Methods in Cloud RPS System*” (Henning Titi Ciptaningtyas, 2022).

### 3.2.3 Analisis dan Desain Perangkat Lunak

Langkah-langkah dari analisis dan desain perangkat lunak yang akan dibuat adalah, pertama-tama melakukan analisa dan *preprocessing* pada dataset yang akan digunakan. Hasil dari *preprocessing* pada dataset ini akan menghasilkan dataset yang bersih dan memiliki format data yang sesuai sehingga siap untuk digunakan dalam penelitian.

Pada penelitian ini sendiri akan dijalankan dua skenario dimana untuk skenario pertama akan dilakukan penjadwalan tugas dan alokasi mesin virtual (VM) menggunakan *Genetic Algorithm* saja dan untuk skenario kedua akan dilakukan menggunakan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Kedua skenario ini akan menggunakan dua dataset seperti yang tertera pada Tabel 3. sebagai sumber data simulasi skenario penjadwalan tugas dan alokasi mesin virtual (VM). Nantinya kedua skenario ini akan dilakukan perbandingan untuk mencari tahu skenario mana yang lebih efisien.

Tabel 3.1 Dataset yang Digunakan

No	Nama Algoritma	Dataset 1	Dataset 2
1	<i>Genetic Algorithm</i>	Dataset yang dibuat sendiri	Dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i> (log, 2003).
2	<i>Genetic Algorithm + Artificial Neural Network</i>	Dataset yang dibuat sendiri	Dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i> (log, 2003).

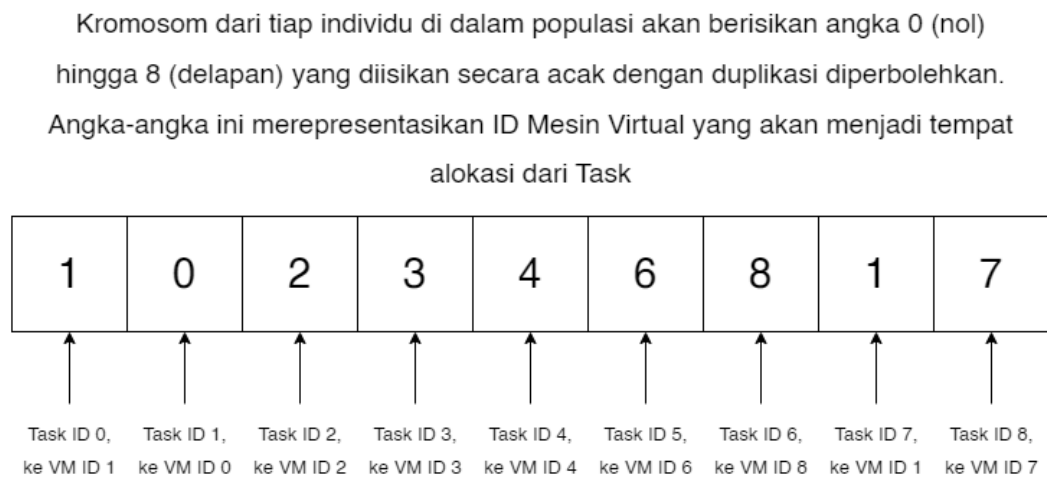
Untuk skenario pertama, akan dilakukan percobaan menggunakan *Genetic Algorithm* saja. Pertama-tama akan dilakukan inisialisasi populasi awal dari jadwal sebagai pedoman pengalokasian *Task* kepada mesin virtual (VM) yang sesuai menggunakan *Genetic Algorithm* saja berdasarkan dataset yang digunakan. Untuk spesifikasi dari *Genetic Algorithm* yang akan digunakan pada percobaan ini bisa dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi *Genetic Algorithm*

Dataset	Besar Populasi	Persentase Mutasi	Persentase Penyilangan	Jumlah Individu Elit	Banyak Loop
Dataset yang dibuat sendiri	20	30%	95%	2	20
Dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i> (log, 2003).	20	30%	95%	2	20

Spesifikasi ini akan mempengaruhi bagaimana *Genetic Algorithm* akan menghasilkan solusi jadwal yang efisien. Besar populasi adalah banyaknya jumlah individu yang berada di dalam satu populasi dalam satu generasi di dalam *Genetic Algorithm* (Mitchell, 1996). Persentase mutasi adalah kemungkinan sebuah gen di dalam kromosom bisa secara acak berubah menjadi gen lain (Mitchell, 1996). Persentase penyilangan adalah kemungkinan diambilnya dua orangtua dari generasi sebelumnya untuk disilangkan gen-nya menjadi satu individu baru di generasi selanjutnya yang memiliki gabungan gen dari kedua orangtuanya (Mitchell, 1996). Sedangkan Jumlah individu elit adalah banyaknya individu yang memiliki nilai *Fitness Function* tinggi dari tiap generasi yang akan dimasukkan secara langsung sebagai anggota di generasi selanjutnya (Mitchell, 1996).

Jadwal ini sendiri akan berbentuk representasi kromosom dari individu di dalam *Genetic Algorithm* sepanjang 9 yang berisikan ID dari mesin virtual (VM) yang dituju oleh *Task*. Representasi kromosom ini bisa dilihat pada Gambar 3.1.



Gambar 3.1 Representasi Kromosom

Jadwal ini akan dilakukan perhitungan *Fitness Function* untuk mencari tahu apakah mereka sudah cukup efisien dalam melakukan penjadwalan tugas dan alokasi mesin virtual (VM) atau belum berdasarkan *Fitness Function* yang bisa dilihat pada Tabel 3.3 hingga Tabel 3.5.

Tabel 3.3 *Fitness Function*

Penulisan
$\alpha * Total Waktu Eksekusi + \beta * \frac{1}{Persentase Kegagalan}$ <p style="text-align: center;">dimana</p> $\alpha + \beta = 1$

Tabel 3.4 Total Waktu Eksekusi

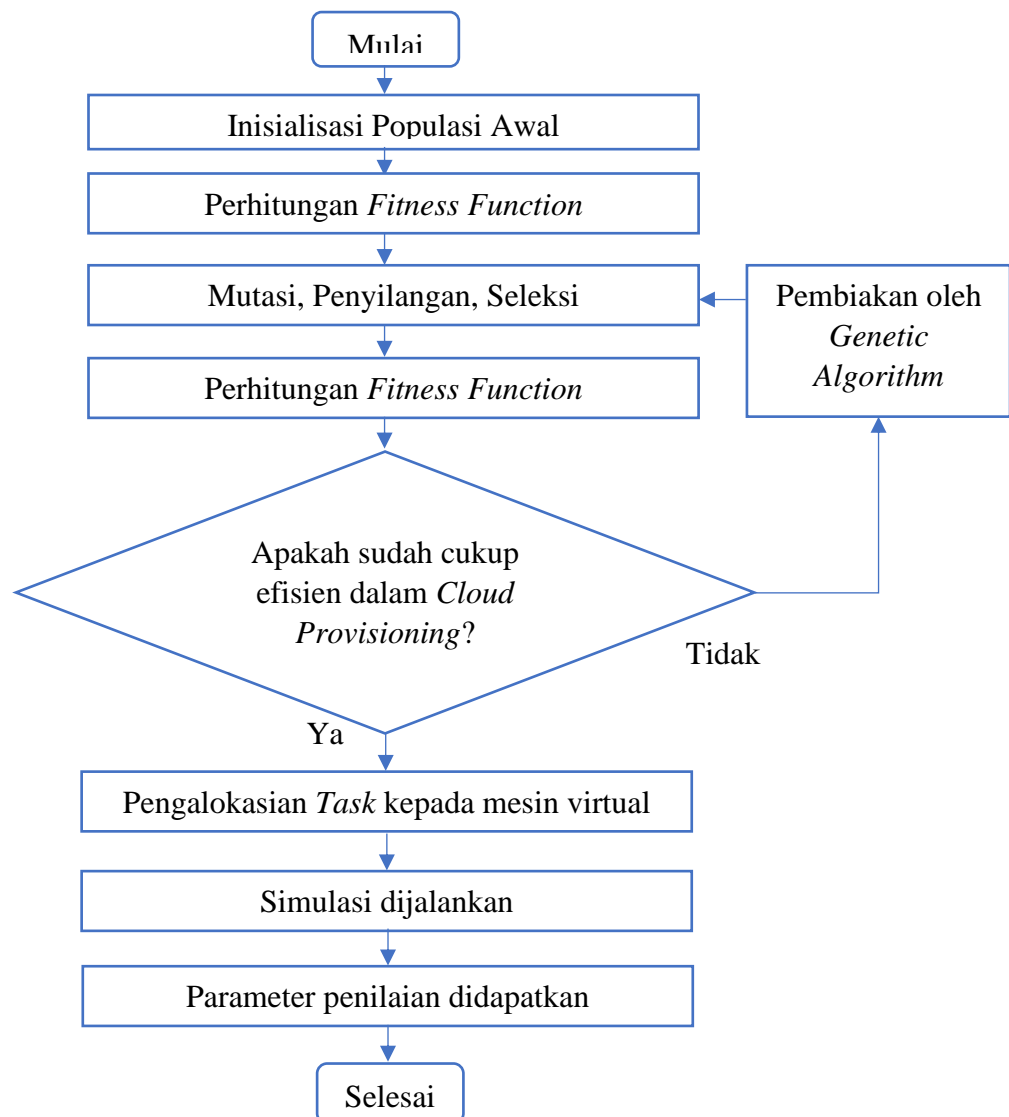
Penulisan
$Total Waktu Eksekusi = \sum_{i=1}^n \frac{Panjang Task i}{Mips VM i}$



Tabel 3.5 Persentase Kegagalan

Penulisan
$\text{Persentase Kegagalan} = \frac{\text{jumlah task gagal}}{\text{total waktu}}$

. Jadwal-jadwal ini kemudian akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* untuk menghasilkan generasi baru yang lebih efisien dari generasi sebelumnya dihitung dari *Fitness Function*-nya. Ketika dirasa tidak ada kenaikan yang signifikan dari *Fitness Function*-nya atau kondisi terminasi sudah tercapai, maka *Genetic Algorithm* akan diberhentikan dan jadwal yang memiliki *Fitness Function* tertinggi akan dijadikan keluaran dari skenario pertama ini. *Task* bisa dialokasikan kepada mesin virtual (VM) berdasarkan jadwal tersebut dan parameter penilaian bisa didapatkan dari hasil simulasinya. Skenario pertama ini bisa dilihat lebih detail pada Gambar 3.1 dibawah ini:



Gambar 3.2 Flowchart Skenario Pertama

Untuk skenario kedua, akan dilakukan percobaan kombinasi dari *Genetic Algorithm* dengan *Artificial Neural Network*. Pertama-tama hasil keluaran jadwal dari skenario pertama akan dilakukan pembagian menjadi 2 yaitu dataset pembelajaran sebanyak 80% dan dataset pengujian sebanyak 20% (Farouk A. Emara, 2021). Dataset pembelajaran akan dilakukan analisa dan dipelajari terlebih dahulu oleh *Artificial Neural Network* untuk menghasilkan model jadwal yang memiliki akurasi yang sesuai berdasarkan keluaran jadwal dari *Genetic Algorithm*. Spesifikasi dari *Artificial Neural Network* untuk penelitian ini bisa dilihat pada Tabel 3.6 dibawah ini.

Tabel 3.6 Spesifikasi *Artificial Neural Network*

Dataset	Input Node	Hidden Node	Output Node	Metode Propagasi	Banyak Epoch	Akurasi
Dataset yang dibuat sendiri	9	18	9	<i>Manhattan</i>	100000	88%
Dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i> (log, 2003).	9	18	9	<i>Manhattan</i>	100000	88%

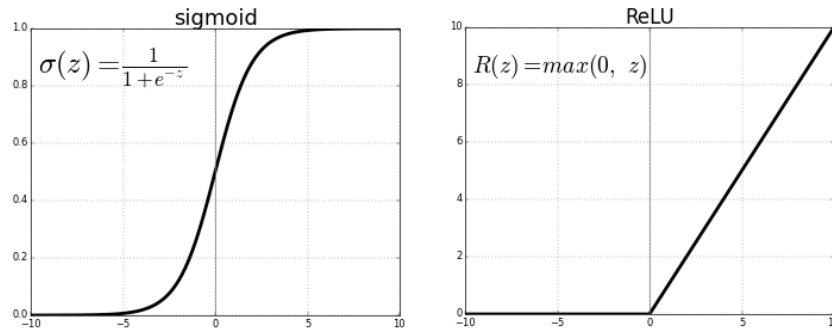
Spesifikasi 9 *Input Node* berguna untuk menerima masukan sebanyak 9 panjang *Task* yang nantinya akan menjadi informasi awal bagi *Artificial Neural Network* untuk mengeluarkan keluaran sebanyak 9 ID VM pada *Output Node*-nya. Di antara *Input Node* dan *Output Node* akan ada 1 *Hidden Node* berisi 18 *Node* sebagai penengah antara keduanya. *Hidden Node* ini akan menggunakan *Activation Function ReLu* dan *Output Node* akan menggunakan *Activation Function Sigmoid* (Heaton, Programming Neural Networks with Encog3 in Java, 2011). Lebih jelasnya bisa melihat pada Tabel 3.7 dibawah ini.

Tabel 3.7 Spesifikasi *Node Artificial Neural Network*

Dataset	Input	Output	Activation Function Pada Hidden Node	Activation Function Pada Output Node
Dataset yang dibuat sendiri	9 Panjang <i>Task</i>	9 ID VM	<i>ReLu</i>	<i>Sigmoid.</i>
Dataset <i>The San Diego Supercomputer Center (SDSC) Blue Horizon logs</i> (log, 2003).	9 Panjang <i>Task</i>	9 ID VM	<i>ReLu</i>	<i>Sigmoid.</i>

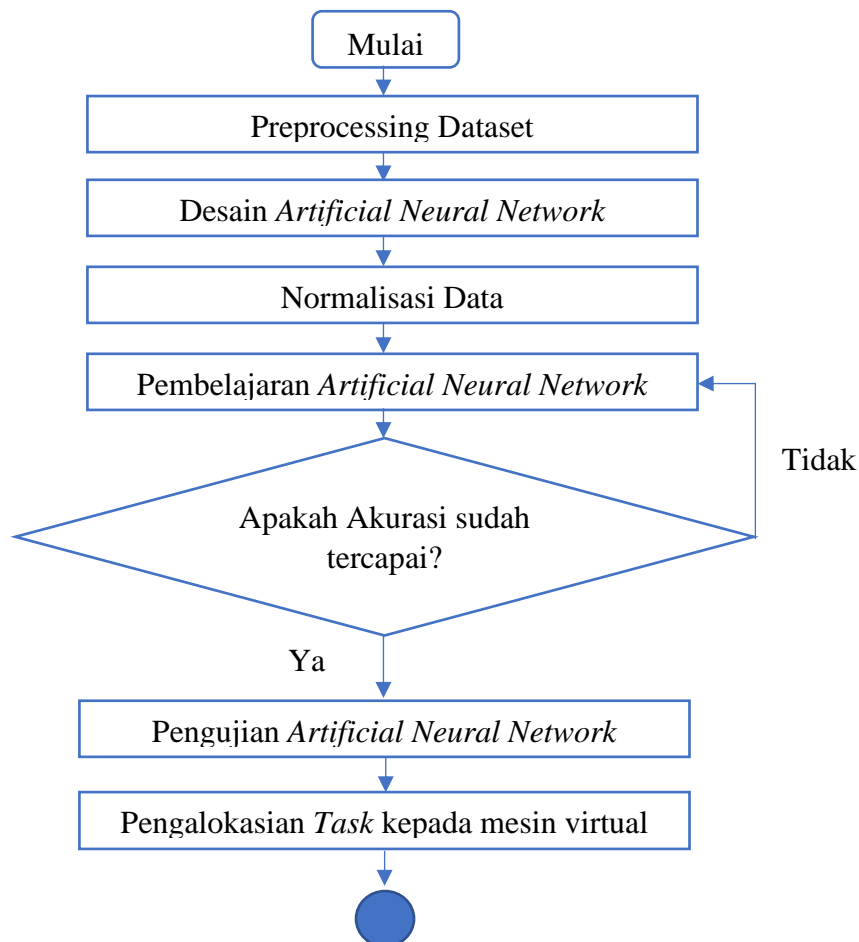
Propagasi *Manhattan* digunakan pada *Artificial Neural Network* ini agar kita bisa mengkontrol secara langsung seberapa besar pengurangan *weight* pada *Artificial Neural Network* sehingga kita bisa mendapatkan tingkat akurasi yang sesuai. Hal ini dikarenakan Propagasi *Manhattan* menggunakan sebuah nilai yang sangat kecil untuk mengurangi ataupun

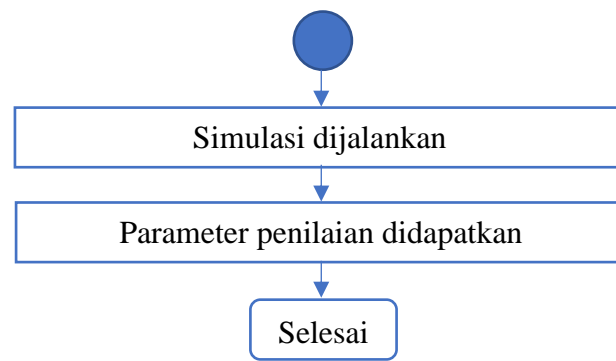
menambahkan *weight*, tidak seperti metode propagasi biasa yang menggunakan *Gradient* sehingga perubahan nilai pada *weight* terlalu besar dan akurasi yang dihasilkan tidak bisa kita kontrol (Heaton, Programming Neural Networks with Encog3 in Java, 2011). Akurasi yang kita tuju pada percobaan ini adalah 88%, karena jika terlalu tinggi akan mengakibatkan *Overfit* dan jika terlalu kecil akan mengakibatkan *Underfit*.



Gambar 3.3 Activation Function Sigmoid dan ReLu (Sharma, 2017)

Setelah dataset pembelajaran dilakukan analisa dan dipelajari oleh *Artificial Neural Network*, model dari prediksi jadwal akan dihasilkan. Model ini akan dilakukan pengujian menggunakan dataset pengujian dan hasil dari jadwal yang dihasilkan akan dimasukkan kedalam simulasi di dalam *CloudSim*. *Task* bisa dialokasikan kepada mesin virtual (VM) berdasarkan jadwal tersebut dan parameter penilaian bisa didapatkan dari hasil simulasinya. Skenario kedua ini bisa dilihat lebih detail pada Gambar 3.4 dibawah ini:

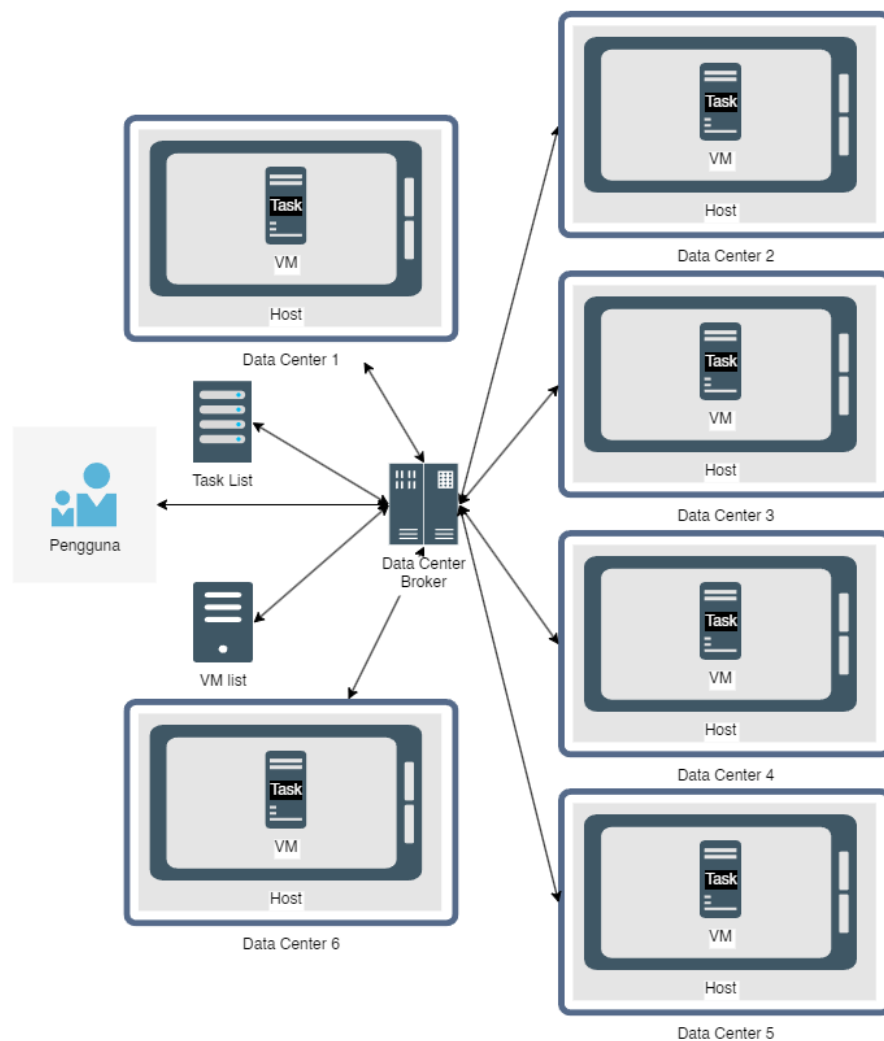




Gambar 3.4 *Flowchart Skenario Kedua*

Setelah kedua skenario diatas selesai dijalankan, maka parameter penilaian dari kedua skenario tersebut akan dilakukan perbandingan untuk mencari tahu kombinasi algoritma mana yang lebih efisien dalam melakukan penjadwalan tugas dan alokasi mesin virtual (VM).

### 3.2.4 Implementasi Perangkat Lunak



Gambar 3.5 Skema Implementasi Perangkat Lunak



Gambar 3.6 Skema Implementasi Perangkat Lunak Pembagian VM

Implementasi dari perangkat lunak akan menggunakan *CloudSIM*. Sebuah kerangka kerja *Open Source*, yang digunakan untuk mensimulasikan layanan *Cloud Computing*. Perangkat lunak ini akan dilakukan implementasi kepada 54 *Virtual Machine* yang berada di dalam 18 *Host*, yang berada di dalam 6 *Data Center*. Masing-masing *Data Center* ini akan terhubung kepada sebuah *Data Center Broker* yang berguna sebagai otak penjadwalan tugas dan alokasi mesin virtual (VM). *Data Center Broker* ini akan terhubung ke *VM List* sebagai daftar *Virtual Machine* yang ada dan status mereka, ke *Task List* sebagai daftar *Task* yang ada dan status mereka, dan terakhir ke pengguna sebagai entitas yang memberikan *Task* dan menerima *Output* hasil pengerjaan. Skema implementasi ini bisa dilihat pada Gambar 3.3.

Setiap *Virtual Machine* dan *Data Center* akan memiliki spesifikasi mereka masing-masing agar penulis bisa melihat hasil yang heterogen. Spesifikasi dari *Virtual Machine*, *Host*, dan *Data Center* ini bisa dilihat secara detail pada Tabel 3.8, Tabel 3.9, dan Tabel 3.10

Tabel 3.8 Spesifikasi *Virtual Machine*

ID	Memori (MB)	RAM (MB)	MIPS tiap Prosesor	Jumlah Prosesor	Bandwith (Mbps)
VM 1	1000	512	400	1	1000
VM 2	1000	1024	500	1	1000
VM 3	1000	2048	600	1	1000

Tabel 3.9 Spesifikasi *Host*

ID	Memori (MB)	RAM (MB)	Jumlah Core	MIPS Core 1	MIPS Core 2	MIPS Core 3	MIPS Core 4
H1	1000000	128000	4	300	400	500	600
H2	1000000	128000	4	300	400	500	600
H3	1000000	128000	4	300	400	500	600

Tabel 3.10 Spesifikasi *Data Center*

ID	Jumlah Prosesor	Jumlah Host	Bandwith (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

### 3.2.5 Pengujian dan Evaluasi

Pengujian dan evaluasi akan dilaksanakan dengan uji coba menggunakan simulasi *Cloud Environment* yang dijalankan pada *CloudSIM* untuk menguji efisiensi melakukan *Cloud Provisioning* menggunakan algoritma *Genetic Algorithm* dan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Kedua skenario tersebut kemudian akan dilakukan perbandingan untuk mencari tahu mana sistem *Cloud Provisioning* yang lebih efisien. Perbandingan tersebut akan didasarkan pada beberapa parameter sesuai dengan Tabel 3.4 dibawah ini:

Tabel 3.11 Parameter yang Digunakan

No	Parameter yang Digunakan
1	<i>Makespan</i>
2	<i>Average Start Time</i>
3	<i>Average Finish Time</i>
4	<i>Average Execution Time</i>
5	<i>Scheduling Time</i>
6	<i>Scheduling Length</i>
7	<i>Throughput</i>
8	<i>Resource Utilization</i>
9	<i>Energy Consumption</i>
10	<i>Imbalance Degree</i>

### 3.2.6 Penyusunan Buku Tugas Akhir

Pada tahap ini akan dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

### 3.3 Jadwal Pengerjaan Tugas Akhir

Pada bagian ini akan dibahas mengenai rencana jadwal pengerjaan tugas akhir yang akan direpresentasikan pada Tabel 3.12 dibawah ini:

Tabel 3.12 Jadwal Pengerjaan

Tahapan	2022																							
	Juli		Agustus				September				Oktober				November				Desember					
Pembuatan Proposal Tugas Akhir																								
Belajar CloudSIM Melalui Tutorial Online																								
Studi Literatur																								
Pembuatan Perangkat Lunak																								
Pengujian Perangkat Lunak																								
Evaluasi Perangkat Lunak																								
Penyusunan Buku Tugas Akhir																								

## DAFTAR PUSTAKA

- Ahmadreza Montazerolghaem, M. H.-G. (2020). Green Cloud Multimedia Networking: NFV/SDN Based Energy-Efficient Resource Allocation. *IEEE*, 4(3), 873 - 889.
- AWS. (2013, March 19). *What is cloud computing?* (AWS) Retrieved July 7, 2022, from <https://aws.amazon.com/what-is-cloud-computing/>
- Dantzig, T. (2007). *Number : the language of science (The Masterpiece Science ed.)*. New York: Plume Book.
- Education, I. C. (2021, September 3). *What Is Resource Utilization?* Retrieved from IBM: <https://www.ibm.com/cloud/blog/what-is-resource-utilization>
- Farouk A. Emara, A. A.-E. (2021). Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing. *International Journal of Intelligent Engineering & Systems*, 1-12.
- Foundation, T. E. (2001, November 29). *Eclipse Desktops & Web IDEs*. (The Eclipse Foundation) Retrieved July 26, 2022, from Eclipse Desktops & Web IDEs: <https://www.eclipse.org/ide/>
- G. B. Mathews, M. (1896). On the Partition of Numbers. *Proceedings of the London Mathematical Society*, s1-28(1), 486–490.
- GeeksForGeeks. (2022, June 28). *Virtualization In Cloud Computing and Types*. Retrieved from GeeksForGeeks: <https://www.geeksforgeeks.org/virtualization-cloud-computing-types/>
- Hardesty, L. (2017, April 4). *Explained: Neural networks*. (MIT News Office) Retrieved July 25, 2022, from <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>
- Heaton, J. (2011). *Programming Neural Networks with Encog3 in Java*. Saint Louis: Heaton Research, Inc.
- Heaton, J. (2015). Encog: Library of Interchangeable Machine Learning Models. *Journal of Machine Learning Research*, 1-5.
- Heaton, J. (2022, 12 25). *Encog Machine Learning Framework*. Retrieved from Heaton Research: <https://www.heatonresearch.com/encog/>
- Henning Titi Ciptaningtyas, A. M. (2022). Survey on Task Scheduling Methods. *2022 International Seminar on Intelligent Technology and Its Applications (ISITIA)*. Surabaya: IEEE.
- Kalita, D. (2022, April 6). *An Overview and Applications of Artificial Neural Networks*. (Analytics Vidhya) Retrieved July 22, 2022, from <https://www.analyticsvidhya.com/blog/2022/03/an-overview-and-applications-of-artificial-neural-networks-ann>
- Kwangwoog Jung, Y.-K. C.-J. (2017). Performance evaluation of ROMS v3.6 on a commercial cloud system. *ResearchGate*.
- Laboratory, C. C. (2018). *CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services*. (School of Computing and Information Systems) Retrieved July 26, 2022, from Laboratory, Cloud Computing and Distributed Systems (CLOUDS): <http://www.cloudbus.org/cloudsim/>



- Larasati, K. D. (2019, July 9). *Artificial Neural Network*. (Medium) Retrieved July 25, 2022, from <https://medium.com/@dhea.larasati326/artificial-neural-network-55797915f14a>
- log, T. S. (2003, January). *The San Diego Supercomputer Center (SDSC) Blue Horizon log*. (The San Diego Supercomputer Center (SDSC) ) Retrieved July 22, 2022, from [https://www.cs.huji.ac.il/labs/parallel/workload/l\\_sdsc\\_blue/index.html](https://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_blue/index.html)
- Martin Feuerman, H. W. (1973). A Mathematical Programming Model for Test Construction and Scoring. *Inform*s, 19(8), 841-971.
- Meier, K. (2020, June 1). *What Resource Utilization Is and How To Calculate It*. Retrieved from Float Resources: <https://www.float.com/resources/guide-to-resource-utilization/>
- Michael Pawlish, A. S. (2012, June). Analyzing Utilization Rates in Data Centers for Optimizing Energy. *2012 International Green Computing Conference (IGCC)* (pp. 1-6). San Jose: IEEE.
- Michael Pawlish, A. S. (2014). A Call for Energy Efficiency in Data Centers. *ACM SIGMOD Record*, 45-51.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.
- Mohammad Hamdaqa, L. T. (2012). Cloud Computing Uncovered: A Research Landscape. *Elsevier*, 86, 41-86.
- Montgomery, J. (2020, October). *Cloud Provisioning*. (Tech Target) Retrieved July 22, 2021, from <https://www.techtarget.com/searchitchannel/definition/cloud-provisioning>
- Moore, S. (2019, July 5). *How to Avoid Overloading Your IT Project Team*. Retrieved from Gartner: <https://www.gartner.com/smarterwithgartner/avoid-overloading-project-team>
- Pradeep Singh Rawat, P. D. (2020). Resource provisioning in scalable cloud using bio-inspired artificial neural network model. *Elsevier*, 1-16.
- Ray, P. P. (2017). An Introduction to Dew Computing: Definition, Concept and Implications. *IEEE*, 6, 723-737.
- Sharma, S. (2017, September 6). *Activation Functions in Neural Networks*. Retrieved from Medium: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
- Singh, A. (2019, July 10). *Beginners Guide to Cloudsim Project Structure*. Retrieved from Cloudsim Tutorials: <https://www.cloudsimtutorials.online/beginners-guide-to-cloudsim-project-structure/>
- Suryansh. (2018, March 26). *Genetic Algorithms + Neural Networks = Best of Both Worlds*. (Towards Data Science) Retrieved July 25, 2022, from <https://towardsdatascience.com/gas-and-nns-6a41f1e8146d>
- Technology, N. I. (2011). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*, 1-7.
- Terh, F. (2019, March 28). *How to solve the Knapsack Problem with dynamic programming*. (Medium) Retrieved July 7, 2022, from <https://medium.com/@fabianterh/how-to-solve-the-knapsack-problem-with-dynamic-programming-eb88c706d3cf>
- Whitley, D. (1994). A Genetic Algorithm Tutorial. *Statistics and Computing*, 1-37.

Wray, J. (2014, February 27). *Where's The Rub: Cloud Computing's Hidden Costs*. (Forbes) Retrieved July 21, 2022, from <https://www.forbes.com/sites/centurylink/2014/02/27/wheres-the-rub-cloud-computings-hidden-costs/>