# USULAN TUGAS AKHIR

## 1. IDENTITAS PENGUSUL

**NAMA** : **Bryan Yehuda Mannuel**
**NRP** : **05311940000021**
**DOSEN WALI** : **Ir. Muchammad Husni, M.Kom**
**DOSEN PEMBIMBING** : **1. Henning Titi Ciptaningtyas, S.Kom, M.Kom.**
**2. Ridho Rahman Hariadi, S.Kom, M.Sc**

## 2. JUDUL TUGAS AKHIR

"CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK"

## 3. LATAR BELAKANG

Di era modern dimana penggunaan teknologi semakin pesat dan meningkat secara cepat, penggunaan *Cloud Computing* [1] semakin banyak diminati. *Cloud Computing* adalah ketersediaan sumber daya sistem komputer sesuai permintaan, seperti penyimpanan data dan daya komputasi, tanpa pengelolaan langsung oleh pengguna [2]. *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 [3]. Oleh karena itu, diperlukan sebuah sistem manajemen sumber daya yang baik bagi sebuah *Cloud Service Provider* agar pengguna dapat memaksimalkan penggunaan sumber daya *Cloud Computing.*

*Cloud provisioning* adalah fitur utama dari model *Cloud Computing*, yang berkaitan dengan cara pelanggan mendapatkan sumber daya *Cloud* dari *Cloud Service Provider* [4]. 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) [5].

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. [6]". 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 [7].

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 [8]. Ditambahkan dengan implementasi *Artificial Neural Network* untuk mempelajari, memproses, dan memprediksi hasil dari sebuah data [9].

## 4. 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 untuk mempelajari, memproses, dan memperkirakan model dari data penggunaan sumber daya Cloud?

## 5. BATASAN MASALAH

Batasan masalah untuk pengerjaan tugas akhir ini adalah sebagai berikut:

1. Proses penelitian akan berupa simulasi Cloud Environment dengan menggunakan CloudSIM dan Eclipse IDE.

2. Dataset yang digunakan untuk penelitian ini akan menggunakan dataset yang dibuat sendiri dan juga mengambil dari dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [10].
3. Proses penelitian akan ditulis menggunakan Bahasa Java menggunakan IDE Eclipse.

# 6. TUJUAN PEMBUATAN TUGAS AKHIR

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.

# 7. MANFAAT TUGAS AKHIR

Manfaat dari pengerjaan tugas akhir ini adalah sebagai berikut:
1. Memaksimalkan penggunaan sumber daya Cloud.
2. Mencegah penurunan performa sumber daya Cloud.
3. Membantu mengurangi biaya modal bagi pengguna sumber daya Cloud.
4. Mencegah pengeluaran biaya operasional yang tidak terduga bagi pengguna sumber daya Cloud.

# 8. TINJAUAN PUSTAKA

## 8.1 Cloud Computing

*Cloud Computing* adalah ketersediaan sumber daya sistem komputer sesuai permintaan, seperti penyimpanan data dan daya komputasi, tanpa pengelolaan langsung oleh pengguna [2]. *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 [3].

*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 [11].

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 [12].

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 [12].

"Lima Karakteristik Penting" tentang *Cloud Computing* dari Definisi National Institute of Standards and Technology adalah [13]:

1. *On-demand self-service* dimana konsumen dapat secara sepihak menyediakan kemampuan komputasi sesuai kebutuhan secara otomatis tanpa memerlukan interaksi manusia dengan setiap *Cloud Service Provider*.
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 sumber daya komputasi *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.

### 8.2 Cloud Provisioning

*Cloud provisioning* adalah fitur utama dari model *Cloud Computing*, yang berkaitan dengan cara pelanggan mendapatkan sumber daya *Cloud* dari *Cloud Service Provider* [4]. Penjadwalan tugas dan alokasi mesin virtual (VM) memainkan peran penting dalam *Cloud provisioning.* Terdapat 3 macam model *Cloud Computing* utama yaitu:

1. Infrastructure-as-a-Service (IaaS) dimana perangkat keras komputer disediakan dan dikelola oleh *Cloud Service Provider* eksternal. Model ini pada dasarnya ditawarkan kepada mereka yang membutuhkan infrastruktur komputer tetapi tidak ingin berurusan dengan kesulitan mengelolanya [14].

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 [15].
3. Platform-as-a-Service (PaaS) dimana produknya adalah platform pengembangan aplikasi berbasis *Cloud*. 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 [16].

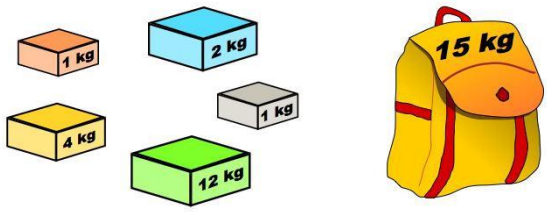
*Cloud Provisioning* memiliki banyak sekali manfaat bagi mereka yang menggunakannya. Manfaat pertama adalah skalabilitas. Dalam model penyediaan kemampuan komputasi tradisional, organisasi memerlukan investasi yang besar dalam menyediakan infrastruktur lokalnya. Hal ini memerlukan persiapan dan prakiraan 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 konsumsi [4].

### 8.3 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. [6]".




*Gambar 1. Ilustrasi Klasik Knapsack Problem* [17]

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 [18].

*Knapsack Problem* dapat juga kita dapati saat kita berusaha melakukan *Cloud Provisioning* karena kita membutuhkan sebuah 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 $Xi$ untuk setiap barang yang ada. $Xi$ dibatasi hingga sebuah angka integer positif. Berikut adalah penulisan dari *Bounded Knapsack Problem* jika diberikan sebuah kumpulan $n$ benda bernomorkan 1 hingga $n$, masing-masing dengan nilai $Vi$ dan berat $wi$, dengan kapasitas maksimum $W$:

$$Maksimalkan \sum_{i=i}^{n} ViXi \; dengan$$

$$memperhatikan \sum_{i=i}^{n} wiXi \; \leq W \; dan \; Xi \; \in (0,1,2,3,\dots,C)$$

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.

### 8.4 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 mutasi, penyilangan dan seleksi [8]. 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 [19].

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 [19].

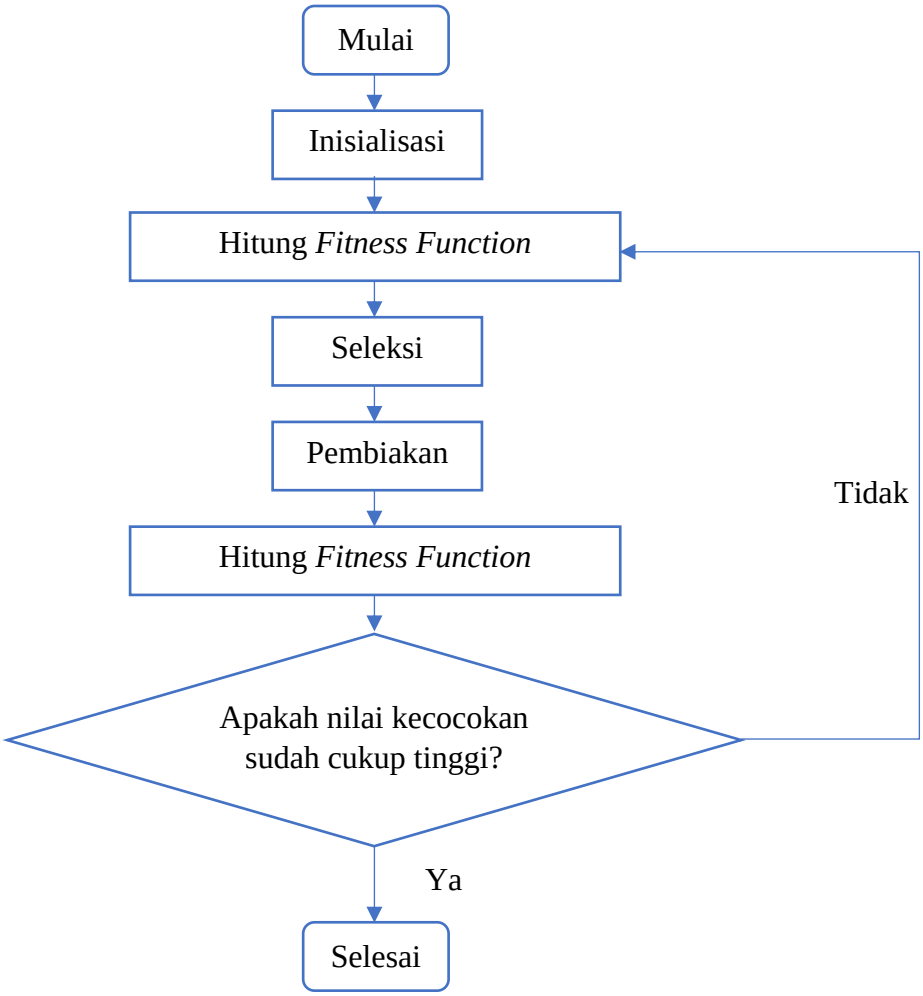
Sebuah *Genetic Algorithm* membutuhkan:

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 merepresentasian genetika dari sebuah kandidat solusi [19].
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 [19].

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. Flowchart Cara Kerja Genetic Algorithm*

### 8.5 Alasan Pemilihan Genetic Algorithm

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

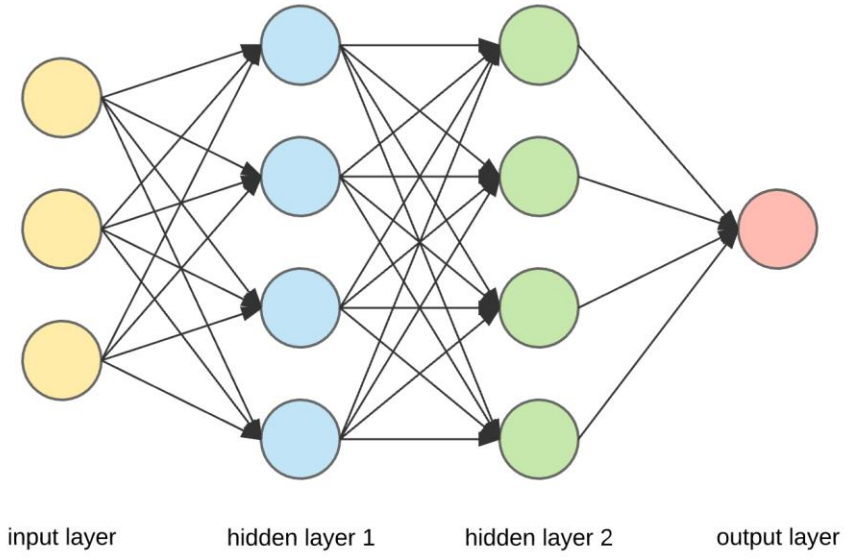
1. *Genetic Algorithm* sering digunakan untuk menghasilkan solusi berkualitas tinggi untuk optimasi penggunaan sumber daya di mana pengambil keputusan harus memilih dari serangkaian tugas yang tidak dapat dibagi di bawah anggaran tetap atau batasan waktu [8].
2. Pengkodean genetik pada *Genetic Algorithm* 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).

### 8.6 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 [20].

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 [20].



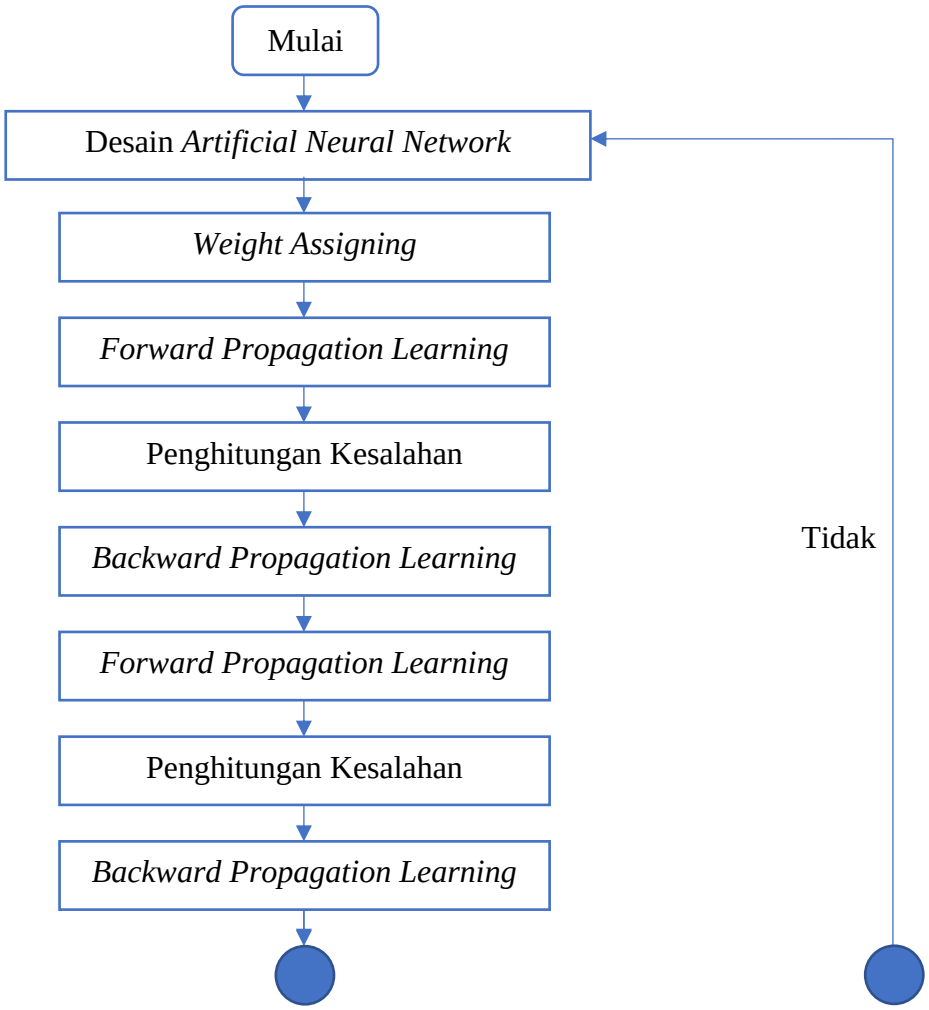

*Gambar 3. Artificial Neural Network dengan 4 Lapisan* [21]

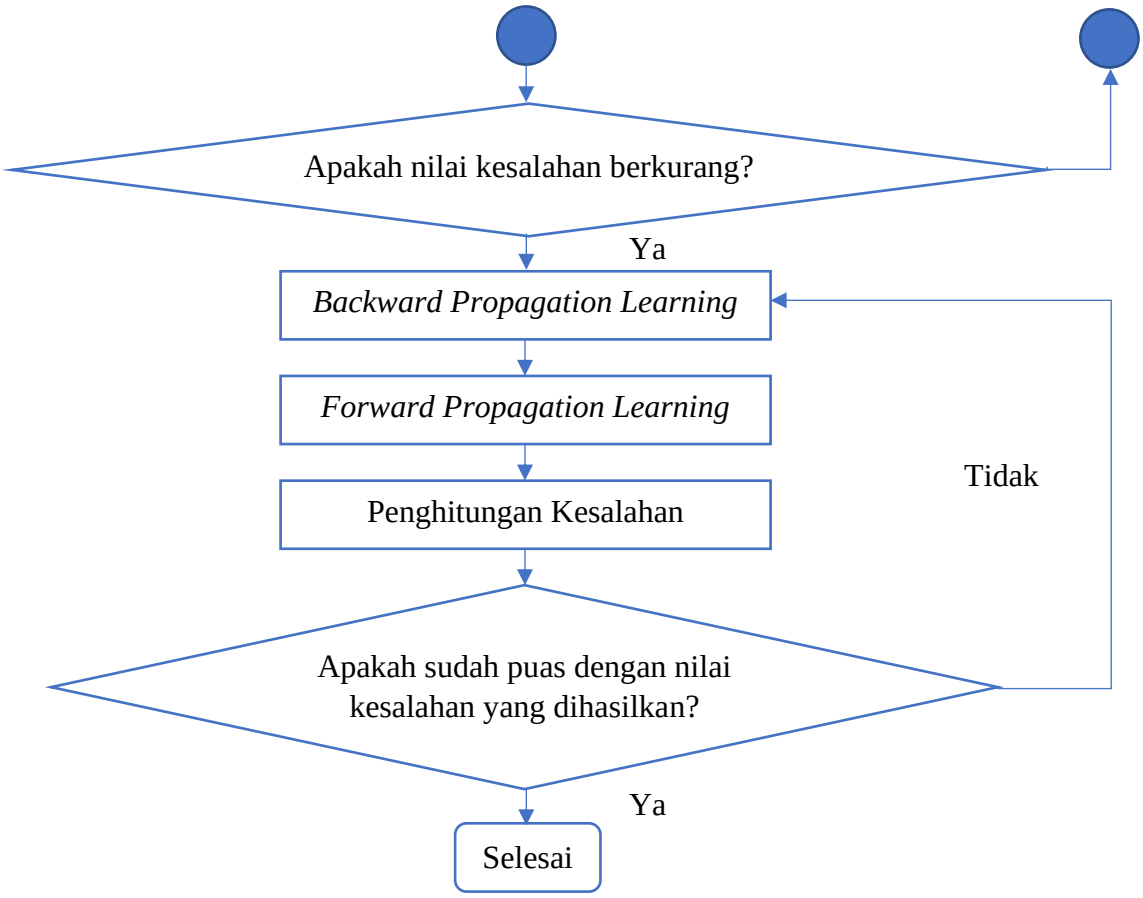
*Artificial Neural Network* belajar (atau dilatih) dengan memproses data, yang masing-masing berisi *input* dan *output* yang diketahui, sehingga membentuk asosiasi bobot dan probabilitas diantara keduanya. Asosiasi ini kemudian disimpan dalam struktur data *Artificial Neural Network* itu sendiri. Pelatihan *Artificial Neural Network* dilakukan dengan menentukan perbedaan antara *output* yang diprediksi dan *output target*. Perbedaan ini disebut sebagai kesalahan yang mana ingin diminalisir 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 [20].

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.






*Gambar 5. Flowchart Cara Kerja Artificial Neural Network*

### 8.7 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 [22].

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.

### 8.8 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* [23].

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 [23].

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 6. Melbourne CLOUDS Lab Pengembang CloudSIM* [23]

### 8.9 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.

### 8.10 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* [24].

*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 [24].

## 9. RINGKASAN ISI TUGAS AKHIR

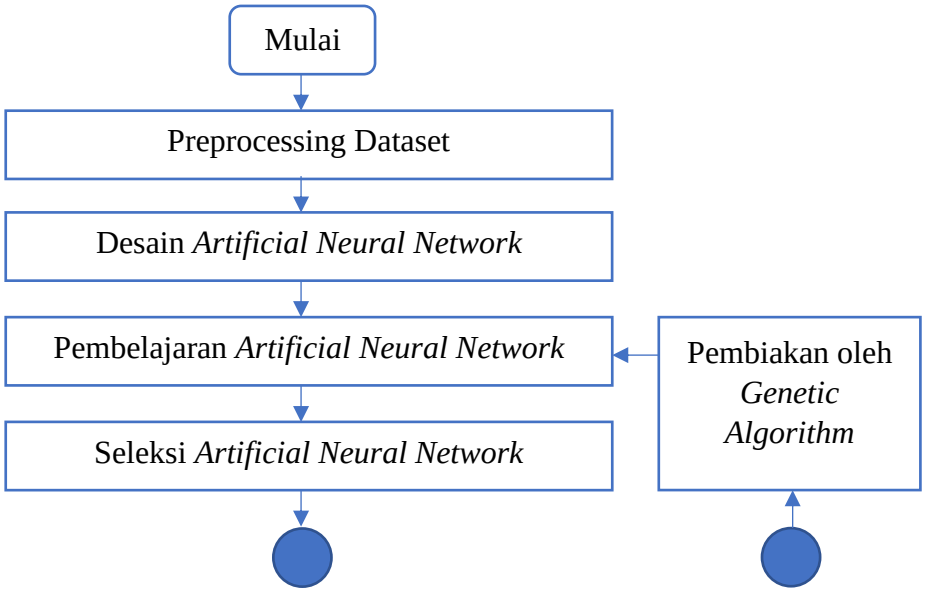
Pada tugas akhir ini akan dibangun sebuah sistem *Cloud Provisioning* yang bisa memaksimalkan penggunaan sumber daya *Cloud*. Untuk bisa melakukan hal tersebut maka dibutuhkan sebuah algoritma yang bisa melakukan penjadwalan tugas dan alokasi mesin virtual (VM) dalam *Cloud Computing* secara efisien. Penulis menyarankan penggunaan *Genetic Algorithm* sebagai algoritma seleksi natural untuk menseleksi sistem penjadwalan tugas dan alokasi mesin virtual (VM) dan *Artificial Neural Network* sebagai algoritma untuk mempelajari, memproses, dan memprediksi hasil dari dataset yang digunakan.

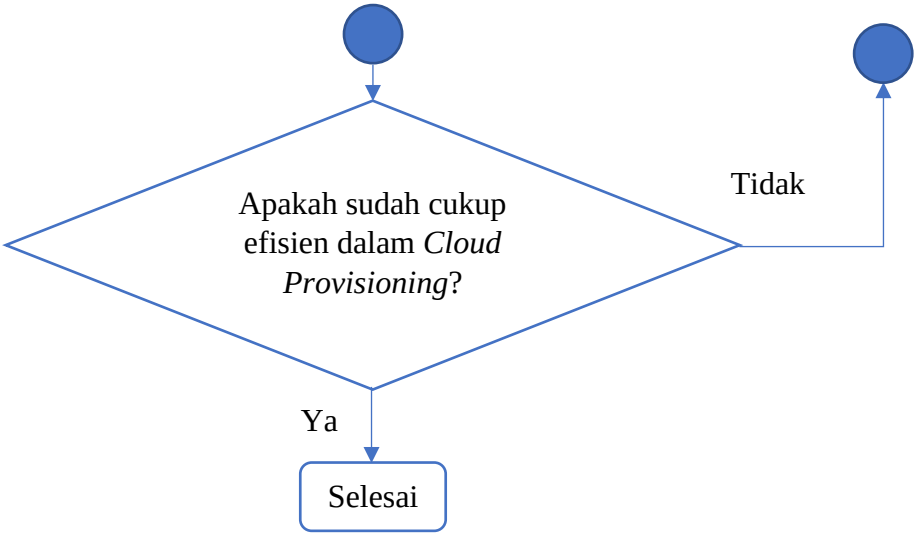
Cara kerja dari penelitian ini adalah pada awalnya, akan dilakukan proses *preprocessing* pada dataset yang akan digunakan. Penelitian ini akan menggunakan dua dataset yaitu dataset yang dibuat sendiri dan juga mengambil dari dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [10]. Guna proses *preprocessing* ini adalah untuk melakukan pembersihan pada kedua dataset tersebut agar data dan *log* yang ada di dalam dataset tersebut menjadi berkualitas tinggi dan memiliki tingkat akurasi yang tinggi.

Kemudian, akan dilakukan proses desain awal dari *Artificial Neural Network* yang akan digunakan untuk mempelajari dataset tersebut. Setelah desain awal selesai, akan dilakukan inisialisasi populasi awal dari *Artificial Neural Network* secara acak sebagai otak yang akan melakukan *Cloud Provisioning*. Pada awalnya, otak *Artificial Neural Network* ini pasti akan memiliki nilai kesalahan yang tinggi karena merupakan generasi pertama. Untuk bisa mengatasi hal tersebut maka akan dilakukan seleksi pada otak-otak *Artificial Neural Network* ini. Semakin rendah nilai kesalahan atau semakin tinggi nilai kecocokan mereka, maka akan semakin tinggi kemungkinan mereka terpilih.

Otak-otak *Artificial Neural Network* yang terpilih akan dilakukan pengembang-biakan menggunakan *Genetic Algorithm*. Dua otak "orangtua" akan dipilih secara acak untuk bisa menghasilkan satu otak "anak". Hal ini akan berakibat terbentuknya populasi untuk generasi kedua dari otak-otak *Artificial Neural Network* yang memiliki nilai kesalahan rendah atau nilai kecocokan yang tinggi. Hal ini dikarenakan otak-otak *Artificial Neural Network* yang memiliki nilai kesalahan tinggi atau nilai kecocokan yang rendah akan tersingkirkan.

Kemudian akan dilakukan iterasi ulang pembelajaran dataset oleh *Artificial Neural Network*, evaluasi nilai mereka, pengembang-biakan oleh *Genetic Algorithm*, dan kembali lagi ke pembelajaran dataset oleh *Artificial Neural Network* secara berulang. Iterasi ini akan terus-menerus diulang hingga suatu saat nanti dinilai otak *Artificial Neural Network* yang melakukan penjadwalan tugas dan alokasi mesin virtual (VM) dalam *Cloud Computing* secara efisien sudah ditemukan.






*Gambar 7. Flowchart Cara Kerja Penelitian*

# 10.METODOLOGI

## 10.1 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.

## 10.2 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 [5]” dan “Resource provisioning in scalable cloud using bio-inspired artificial neural network model [11]”.

### 10.3 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 *Clean* sehingga siap dilakukan analisa dan dipelajari oleh *Artificial Neural Network* untuk menghasilkan model yang bisa melakukan penjadwalan tugas dan alokasi mesin virtual (VM) secara efisien. Model-model ini kemudian akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* untuk menghasilkan generasi baru yang lebih efisien dari model sebelumnya. Generasi terbaru hasil penyilangan akan menganalisa dan mempelajari kembali dataset menggunakan *Artificial Neural Network* dan akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* kembali. Iterasi ini akan dilakukan terus-menerus hingga menghasilkan model *Cloud Provisioning* yang efisien.

### 10.4 Implementasi Perangkat Lunak

Implementasi dari perangkat lunak akan menggunakan CloudSIM.

### 10.5 Pengujian dan Evaluasi

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

### 10.6 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. Untuk sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    **1.1. Latar Belakang**
    **1.2. Rumusan Masalah**
    **1.3. Batasan Tugas Akhir**
    **1.4. Tujuan Tugas Akhir**
    **1.5. Metodologi Penelitian**
    **1.6. Sistematika Penulisan**
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

## 11.JADWAL KEGIATAN

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

# 12.DAFTAR PUSTAKA


[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] J. Wray, "Where's The Rub: Cloud Computing's Hidden Costs," Forbes, 27 February 2014. [Online]. Available: https://www.forbes.com/sites/centurylink/2014/02/27/wheres-the-rub-cloud-computings-hidden-costs/. [Accessed 21 July 2022].

[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] 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].

[11] AWS, "What is cloud computing?," AWS, 19 March 2013. [Online]. Available: https://aws.amazon.com/what-is-cloud-computing/. [Accessed 7 July 2022].

[12] L. T. Mohammad Hamdaqa, "Cloud Computing Uncovered: A Research Landscape," *Elsevier,* vol. 86, pp. 41-86, 2012.

[13] N. I. o. S. a. Technology, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology,* pp. 1-7, 2011.

[14] Techslang, "What is Infrastructure as a Service (IaaS)?," Techslang, [Online]. Available: https://www.techslang.com/definition/what-is-infrastructure-as-a-service-iaas/. [Accessed 7 July 2022].

[15] Techslang, "What is Software as a Service (SaaS)?," Techslang, [Online]. Available: https://www.techslang.com/definition/what-is-software-as-a-service-saas/. [Accessed 7 July 2022].

[16] Techslang, "What is Platform as a Service (PaaS)?," Techslang, [Online]. Available: https://www.techslang.com/definition/what-is-platform-as-a-service-paas/#long-answer. [Accessed 7 July 2022].

[17] F. Terh, "How to solve the Knapsack Problem with dynamic programming," Medium, 28 March 2019. [Online]. Available: https://medium.com/@fabianterh/how-to-solve-the-knapsack-problem-with-dynamic-programming-eb88c706d3cf. [Accessed 7 July 2022].

[18] H. W. Martin Feuerman, "A Mathematical Programming Model for Test Construction and Scoring," *Informs,* vol. 19, no. 8, pp. 841-971, 1973.

[19] D. Whitley, "A Genetic Algorithm Tutorial," *Statistics and Computing,* pp. 1-37, 1994.

[20] L. Hardesty, "Explained: Neural networks," MIT News Office, 4 April 2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. [Accessed 25 July 2022].

[21] K. D. Larasati, "Artificial Neural Network," Medium, 9 July 2019. [Online]. Available: https://medium.com/@dhea.larasati326/artificial-neural-network-55797915f14a. [Accessed 25 July 2022].

[22] Suryansh, "Genetic Algorithms + Neural Networks = Best of Both Worlds," Towards Data Science, 26 March 2018. [Online]. Available: https://towardsdatascience.com/gas-and-nns-6a41f1e8146d. [Accessed 25 July 2022].

[23] C. C. a. D. S. (. Laboratory, "CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services," School of Computing and Information Systems, [Online]. Available: http://www.cloudbus.org/cloudsim/. [Accessed 26 July 2022].

[24] T. E. Foundation, "Eclipse Desktops & Web IDEs," The Eclipse Foundation, [Online]. Available: https://www.eclipse.org/ide/. [Accessed 26 July 2022].

[25] 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.