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* 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, 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 sebuah data (Kalita, 2022).

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* bersamaan dengan *Genetic Algorithm* 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* (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.*

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* sehingga meningkatkan tingkat penggunaan sumber daya *Cloud* dan mengurangi energi dan tenaga yang terbuang sia-sia.

7. MANFAAT TUGAS AKHIR

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.*

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 (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):

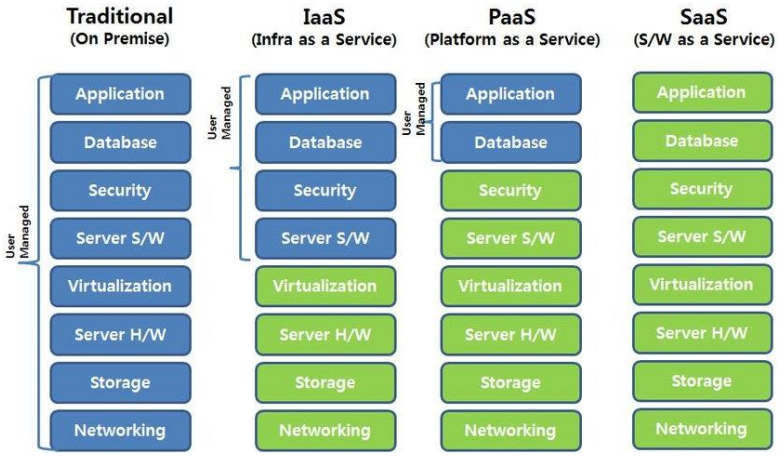
1. *Private Cloud* - Infrastruktur *Cloud* yang disediakan secara khusus untuk digunakan oleh satu organisasi yang terdiri dari beberapa unit bisnis. *Private Cloud* dimiliki, dikelola dan dioperasikan oleh satu organisasi, pihak ketiga, atau kombinasi keduanya, dan dapat berada pada satu tempat yang 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 di operasikan 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 1 Model Layanan Cloud Computing (Kwangwoog Jung, 2017)

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 (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).

8.3 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).

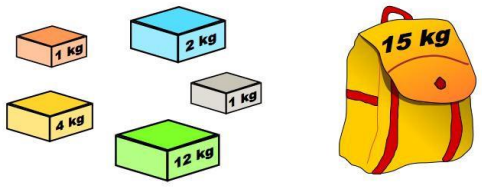
Beberapa manfaat virtualisasi yang sangat penting agar sistem *Cloud Computing* dapat berjalan dengan lancar antara lain (GeeksForGeeks, 2022):

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.

8.4 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. Ilustrasi Klasik 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 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.

8.5 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 mutasi, penyilangan dan seleksi (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:

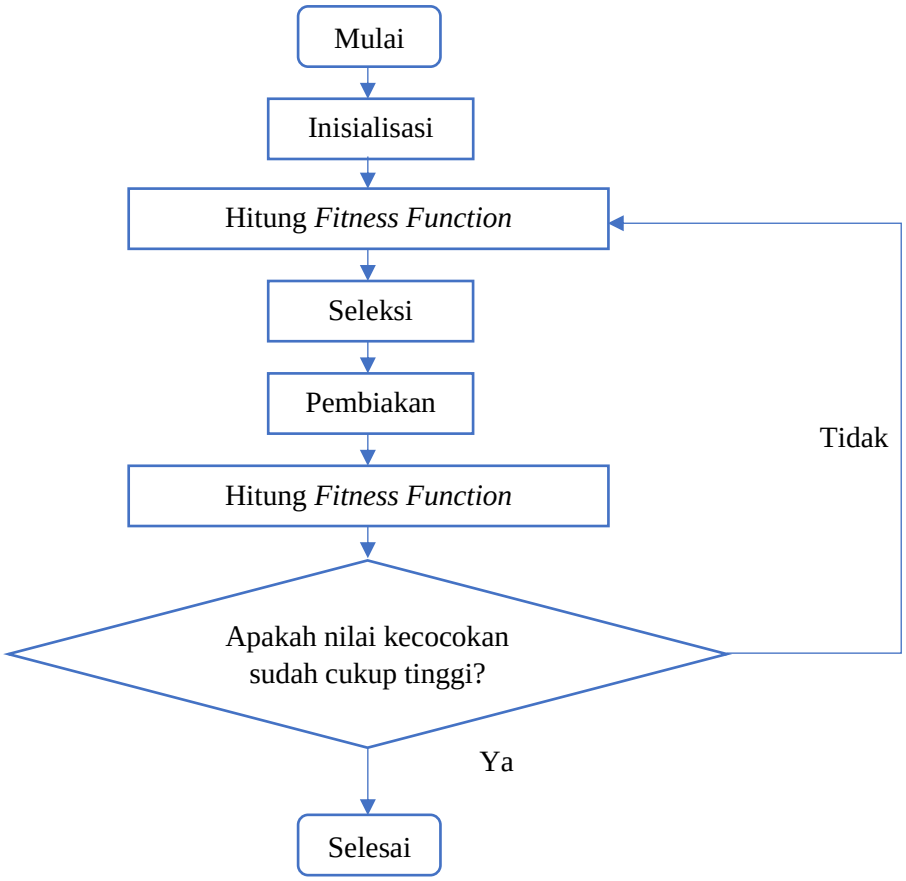
1. Representasi genetik dari kandidat solusi. Representasi ini biasanya dituliskan dalam bentuk *Array of Bits* dengan panjang yang tetap. *Array of Bits* ini akan berisikan bilangan biner 1 dan 0 yang merepresentasian genetika dari sebuah kandidat solusi (Whitley, 1994).
2. *Fitness Function* untuk 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 3. Flowchart Cara Kerja Genetic Algorithm

8.6 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).

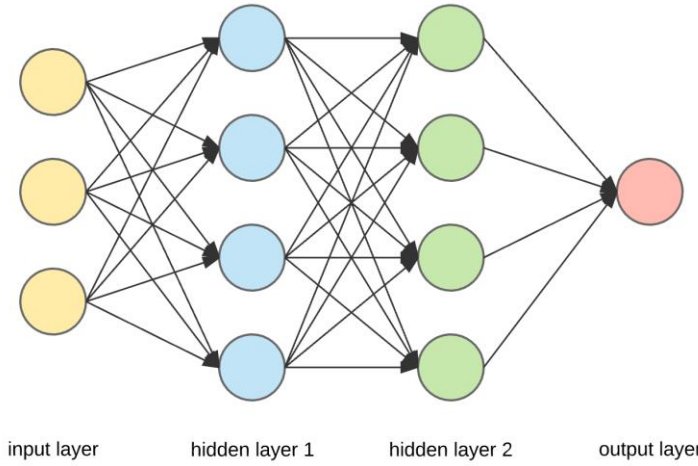
8.7 Artificial Neural Network

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 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 (Hardesty, 2017).



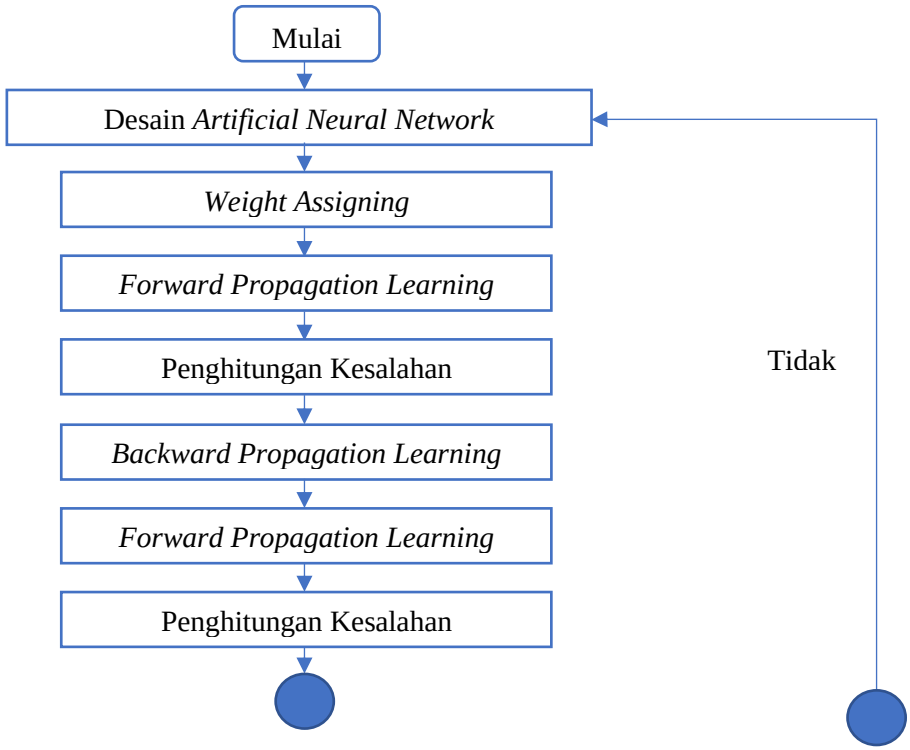
Gambar 4. Artificial Neural Network dengan 4 Lapisan (Larasati, 2019)

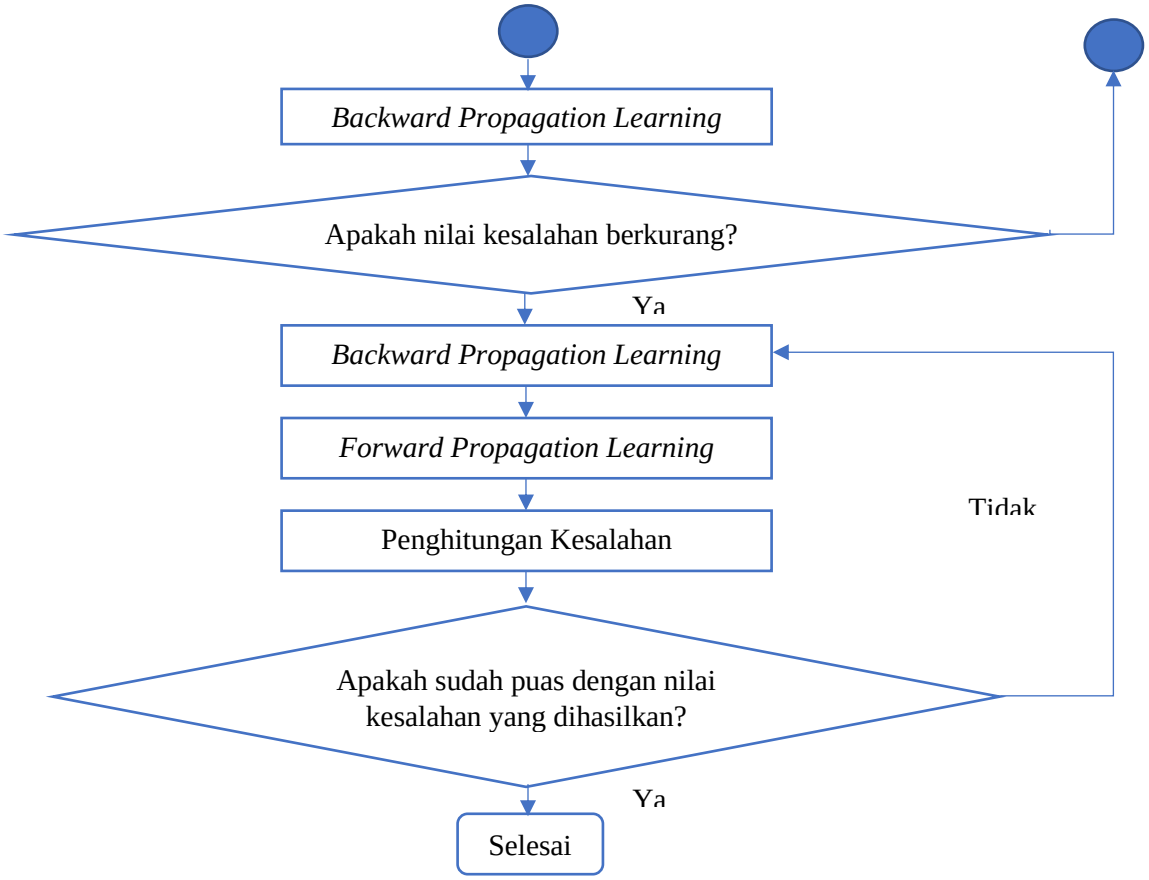
Proses kerja dari *Artificial Neural Network* terdiri dari beberapa langkah yakni:

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.





Gambar 5. Flowchart Cara Kerja Artificial Neural Network

8.8 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.

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* (Laboratory, n.d.).

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, n.d.).

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 (Laboratory, n.d.)

8.10 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 1. Namespace CloudSim

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

No	Namespace	Fungsi
		Namespace ini mungkin akan digunakan suatu saat nanti di masa depan.
5	*Org.cloudbus.cloudsim.lists*	*Namespace* ini berisi implementasi daftar yang akan digunakan secara global selama proses simulasi. *Namespace* ini adalah kumpulan kelas khusus, di mana operasi penyortiran dan perbandingan diimplementasikan. Ada kelas daftar untuk host, elemen pemrosesan, tugas, mesin virtual, dan sumber daya.
6	*Org.cloudbus.cloudsim.network*	*Namespace* ini berisi implementasi untuk tujuan simulasi terkait jaringan dan internet.
7	*Org.cloudbus.cloudsim.network.datacenter*	*Namespace* ini berisi implementasi yang diperluas dari komponen perangkat keras sistem *Cloud Computing* dasar yang dapat mendukung simulasi fungsi C*loud* gabungan yang terdistribusi di banyak wilayah.
8	*Org.cloudbus.cloudsim.power*	*Namespace* ini berisi implementasi komponen sistem *Cloud Computing* yang diperluas, yang dapat mensimulasikan skenario komputasi hijau (*Green Computing)* atau sadar daya (Power Aware).
9	*Org.cloudbus.cloudsim.power.lists*	*Namespace* ini hanya berisi implementasi *powervmlist.java*, yang merupakan versi lanjutan dari class *VmList.java* dari *namespace org.cloudbus.cloudsim.list*
10	*Org.cloudbus.cloudsim.power.models*	*Namespace* 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.
11	*Org.cloudbus.cloudsim.provisioners*	*Namespace* ini berisi implementasi perilaku tentang bagaimana komponen sistem *Cloud Computing* dapat disediakan untuk memenuhi permintaan sumber daya virtual.
12	*Org.cloudbus.cloudsim.util*	*Namespace* ini berisi implementasi fungsi perhitungan penting dan kumpulan utilitas umum yang digunakan oleh *Namespace lain*.

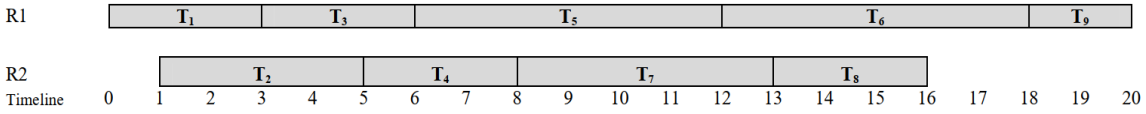
8.11 Alasan Pemilihan CloudSIM

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.

8.12 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 7) (Henning Titi Ciptaningtyas, 2022):



Gambar 7. Contoh Penjadwalan Tugas (Henning Titi Ciptaningtyas, 2022)

Tabel 2. Parameter Tingkat Penggunaan Sumber Daya Cloud

Nama	Definisi	Cara Menghitung
Makespan	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
Average Start Time	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

Nama	Definisi	Cara Menghitung
Average Finish Time	Waktu rata-rata tugas selesai dieksekusi (Pradeep Singh Rawat, 2020).	$\frac{\sum_{i=1}^{n} Waktu\ Ri\ Selesai}{nR}$ Contoh: (20+16)/2 = 18
Average Execution Time	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
Scheduling Time	Lama waktu yang dibutuhkan untuk tugas pertama mulai dieksekusi (Pradeep Singh Rawat, 2020).	Contoh : Dari gambar kita ambil bahwa *Scheduling Time* adalah 0, karena tidak ada delay untuk penjadwalan tugas
Scheduling Length	Jumlah total waktu yang dibutuhkan dari awal simulasi dimulai hingga simulasi selesai (Farouk A. Emara, 2021).	$Scheduling\ Time + Makespan$ Contoh: 0+20 = 20
Throughput	Jumlah total tugas yang menyelesaikan eksekusi per unit waktu (Henning Titi Ciptaningtyas, 2022).	$\frac{nT}{Makespan}$ Contoh: 9/20 = 0.45
Resource Utilization	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
Energy Consumption	Jumlah total penggunaan energi saat pengerjaan tugas (Farouk A. Emara, 2021).	
Imbalance Degree	Pengukuran ketidakseimbangan diantara semua pusat data (Farouk A. Emara, 2021).	$\frac{ETmax - ETmin}{ETavg}$ Dimana $ETmax, ETmin, ETavg$ adalah *Execution Time* maksimum, minimum, dan rata-rata secara urut dari semua pusat data yang ada. Contoh: (6-2)/32.3 = 0.12

8.13 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 kuran (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 analis *Gartner,* jawabannya adalah 70% hingga 80% (Moore, 2019).

8.14 Definisi Parameter Penggunaan Sumber Daya Cloud

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 penggunanya (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 pengunaan 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).

8.14 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, Eclipse Desktops & Web IDEs, n.d.).

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, Eclipse Desktops & Web IDEs, n.d.).

Gambar 8. Logo Eclipse IDE (Foundation, Eclipse Desktops & Web IDEs, 2001)

9. RINGKASAN ISI TUGAS AKHIR

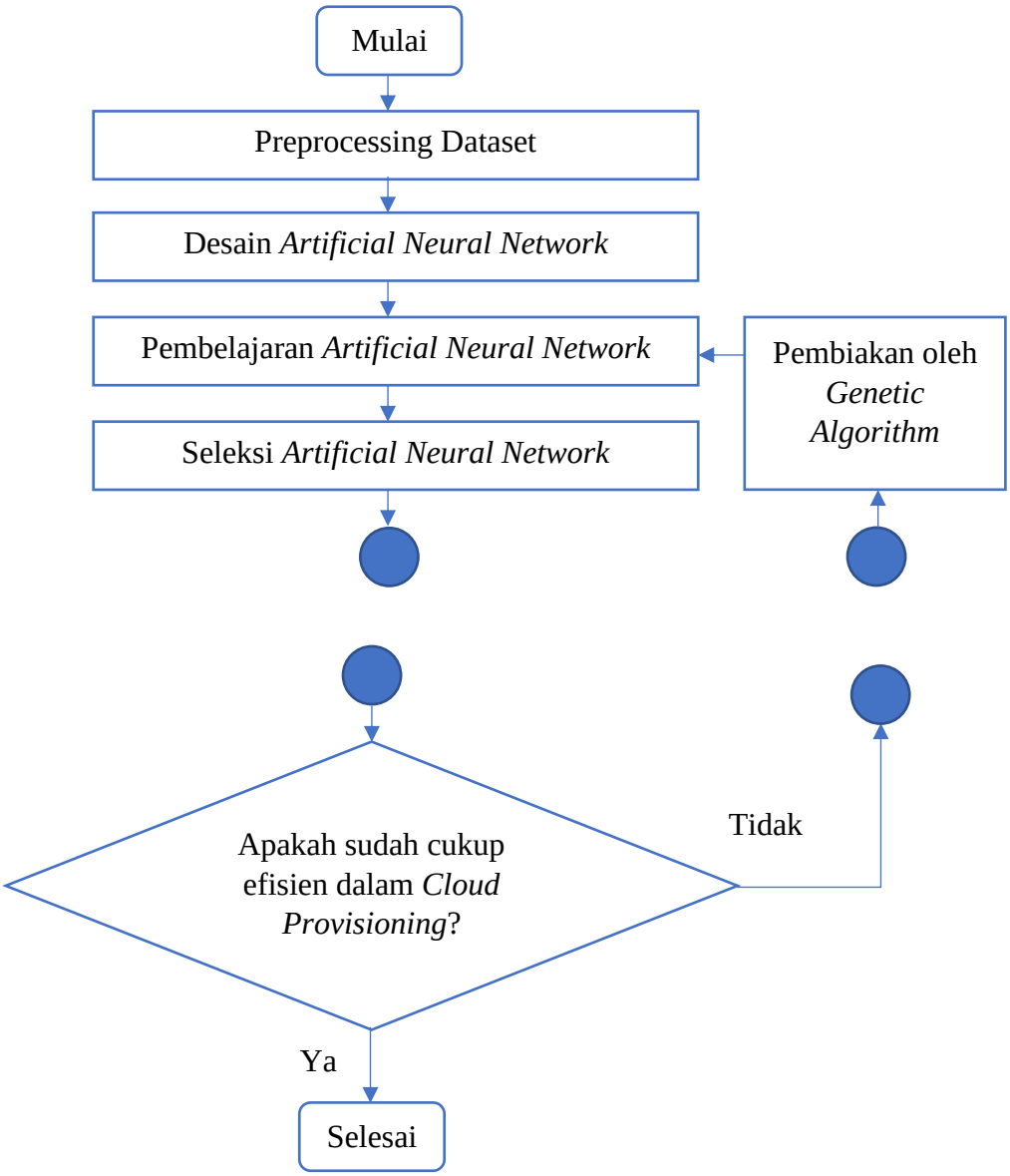
Pada tugas akhir ini akan dibangun sebuah sistem *Cloud Provisioning* yang bisa memaksimalkan penggunaan sumber daya *Cloud*. Untuk bisa melakukan hal tersebut maka dibutuhkan sebuah algoritma yang bisa melakukan penjadwalan tugas dan alokasi mesin virtual (VM) dalam *Cloud Computing* secara efisien. 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 (log, 2003). 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 9. 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"* (Farouk A. Emara, 2021) dan "*Resource provisioning in scalable cloud using bio-inspired artificial neural network model"* (Pradeep Singh RawatPriti, 2020).

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 bersih sehingga siap 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 algoritma mana yang lebih efisien dalam melakukan penjadwalan tugas dan alokasi mesin virtual (VM).

Untuk skenario pertama akan dilakukan inisialisasi populasi awal dari mesin virtual (VM) menggunakan *Genetic Algorithm.* Kemudian *Task Cloud Computing* yang dianggap sebagai permintaan dari pengguna berdasarkan dataset akan dilakukan alokasi menggunakan *Genetic Algorithm* kepada setiap mesin virtual (VM) yang dianggap sesuai. Mesin-mesin virtual ini kemudian akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* kembali untuk menghasilkan generasi baru yang lebih efisien dari generasi sebelumnya. Untuk skenario kedua, dataset akan dilakukan analisa dan dipelajari terlebih dahulu oleh *Artificial Neural Network* untuk menghasilkan model mesin virtual (VM) yang bisa melakukan penjadwalan tugas secara efisien. Model-model mesin virtual ini kemudian akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* untuk menghasilkan generasi baru yang lebih efisien dari model sebelumnya.

Pada kedua skenario ini, generasi terbaru hasil penyilangan akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* kembali. Iterasi ini akan dilakukan secara

terus menerus hingga tidak didapati peningkatan efisiensi yang signifikan. Kedua skenario tersebut kemudian akan dilakukan perbandingan untuk mencari tahu mana sistem *Cloud Provisioning* yang lebih efisien.

10.4 Implementasi Perangkat Lunak

Implementasi dari perangkat lunak akan menggunakan *CloudSIM*. Sebuah kerangka kerja *Open Source*, yang digunakan untuk mensimulasikan layanan *Cloud Computing*.

10.5 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*.

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																		

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/

Foundation, T. E. (n.d.). *Eclipse Desktops & Web IDEs*. (The Eclipse Foundation) Retrieved July 26, 2022, from 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

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. (n.d.). *CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services*. (School of Computing and Information Systems) Retrieved July 26, 2022, from 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

Martin Feuerman, H. W. (1973). A Mathematical Programming Model for Test Construction and Scoring. *Informs, 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.

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.

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.

Techslang. (n.d.). *What is Infrastructure as a Service (IaaS)?* (Techslang) Retrieved July 7, 2022, from https://www.techslang.com/definition/what-is-infrastructure-as-a-service-iaas/

Techslang. (n.d.). *What is Platform as a Service (PaaS)?* (Techslang) Retrieved July 7, 2022, from https://www.techslang.com/definition/what-is-platform-as-a-service-paas/#long-answer

Techslang. (n.d.). *What is Software as a Service (SaaS)?* (Techslang) Retrieved July 7, 2022, from https://www.techslang.com/definition/what-is-software-as-a-service-saas/

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/

Lembar Pengesahan

Proposal Tugas Akhir

CLOUD PROVISIONING MENGGUNAKAN GENETIC ALGORITHM DAN ARTIFICIAL NEURAL NETWORK

Bryan Yehuda Mannuel

05311940000021

Surabaya, 28 Juli 2022

Menyetujui,

Dosen Pembimbing 1	Dosen Pembimbing 2
Henning Titi Ciptaningtyas, S.Kom, M.Kom.	**Ridho Rahman Hariadi, S.Kom, M.Sc**
NIP. 19840708 201012 2 004	**NIP. 19870213 201404 1 001**