Cloud Provisioning Menggunakan *Genetic Algorithm* dan *Artificial Neural Network*

Bryan Yehuda Mannuel, Henning Titi Ciptaningtyas, dan Ridho Rahman Hariadi
Departemen Teknologi Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS)
Gedung Perpustakaan ITS, lt.6 Kampus ITS Sukolilo, Surabaya 60111 Indonesia
e-mail: bryanyehuda@gmail.com, henning@if.its.ac.id, ridho@if.its.ac.id

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

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

I. PENDAHULUAN

Di era modern dimana penggunaan teknologi semakin pesat dan meningkat secara cepat, penggunaan *Cloud Computing* semakin banyak diminati [1]. *Cloud Computing* adalah ketersediaan sumber daya sistem komputer sesuai permintaan, seperti penyimpanan data dan daya komputasi, tanpa pengelolaan langsung oleh pengguna [2]. 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 [3]. 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* [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 solusi optimasi [9].

II. TINJAUAN PUSTAKA

Farouk dkk. [5] membahas tantangan membangun sistem penjadwalan tugas dan alokasi mesin virtual (VM) dalam sistem *Cloud Computing* karena heterogenitas yang ada di dalamnya. Jurnal ini menyarankan penggunaan *Genetic Algorithm* yang telah dimodifikasi untuk mengatasi beberapa target pencapaian, termasuk memaksimalkan *Resource Utilization, Load Balancing,* dan *Power Management*. *Genetic Algorithm* yang telah dimodifikasi ini menggunakan struktur matriks untuk mewakili kromosom dan ditemukan mengungguli algoritma lain dalam hal *Makespan, Scheduling Length, Throughput, Resource Utilization, Energy Consumption,* dan *Imbalance Degree*. Penulis mereferensikan penelitian ini dan mengusulkan penggunaan *Genetic Algorithm* yang dibantu oleh *Artificial Neural Network* untuk melihat apakah hasil yang lebih baik dapat

dicapai. Penggunaan *ID Tasks, ID VM,* dan *ID Data Center* sebagai representasi kromosom juga dirujuk.

Pradeep dkk. [10] mempelajari efisiensi berbagai algoritma penjadwalan tugas dalam sistem *Cloud Computing*, seperti *BB-BC, GA-Cost, GA-Exe,* dan *GA-ANN*. Jurnal ini membandingkan algoritma tersebut dengan menggunakan berbagai parameter, seperti *Average Start Time, Average Finish Time, Average Execution Time,* dan *Total Wait Time*. Jurnal ini juga mencakup variabel kontrol yang bisa direferensikan dalam penelitian ini, seperti jumlah *Task*, mesin virtual (VM), dan *Data Center*, serta *bandwidth*, *CPU*, dan *RAM*. Jurnal ini menggunakan dua sumber data, data yang dihasilkan sendiri dan dataset *Blue Horizon Log* dari *San Diego Supercomputer Center*. Hasil penelitian menunjukkan bahwa algoritma *GA-ANN* adalah yang paling efisien, dengan pengurangan tingkat kesalahan sebesar 82,63%, peningkatan tugas yang berhasil diselesaikan sebesar 26,81%, pengurangan waktu eksekusi sebesar 10,66%, dan pengurangan waktu tunggu sebesar 69,94%. Penulis berencana untuk mempelajari lebih lanjut keefektifan algoritma *GA-ANN* dibandingkan dengan *Genetic Algorithm* saja.

Michael dkk. [11] membahas efisiensi energi dari sistem *Cloud Computing* saat ini. Ditemukan bahwa *Resource Utilization*, yang mengukur persentase efisiensi penggunaan sistem, hanya berada di kisaran 30-42% saja. *Resource Utilization* yang rendah ini dikaitkan dengan kurangnya pemanfaatan kemampuan virtualisasi yang tepat oleh Penyedia Layanan *Cloud*, yang menyebabkan pemborosan energi dan tenaga kerja. Jurnal ini meminta tindakan segera diambil oleh Penyedia Layanan *Cloud* untuk mengatasi masalah ini dan menyarankan pembuatan sistem penjadwalan tugas dan alokasi VM untuk meningkatkan *Resource Utilization*.

Henning dkk. [12] menyajikan metode untuk mengevaluasi kinerja penjadwalan tugas dan sistem alokasi mesin virtual, termasuk definisi dan rumusnya seperti *Makespan*, *Throughput*, dan *Resource Utilization*. Karena alasan tersebut penulis menggunakan jurnal ini sebagai referensi dalam penelitian ini.

Penulis ingin membangun sistem yang dapat menjadwalkan tugas secara efektif dan mengalokasikan mesin virtual untuk memaksimalkan *Resource Utilization* dengan menggunakan *Genetic Algorithm* dan *Artificial Neural Network* untuk memecahkan "*Knapsack Problem*" yang ditemui dalam *Cloud Provisioning*. Dengan demikian tidak hanya penelitian ini bisa mencegah penurunan kinerja, tetapi juga dapat meningkatkan tingkat penggunaan sumber daya *Cloud*. Penelitian ini juga dapat mengurangi waktu eksekusi serta mengurangi pembuangan energi secara sia-sia. Secara keseluruhan, penelitian ini bertujuan untuk menciptakan cara yang lebih efisien dan efektif dalam memanfaatkan sumber daya *Cloud*.

III. METODOLOGI PENELITIAN

A. Dataset yang Digunakan

Pada penelitian ini sendiri akan dijalankan dua skenario dimana untuk skenario pertama akan dilakukan penjadwalan tugas dan alokasi mesin virtual (VM) menggunakan *Genetic Algorithm* saja dan untuk skenario kedua akan dilakukan dilakukan menggunakan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network.* Kedua skenario ini akan menggunakan dua dataset seperti yang tertera pada Tabel 1 sebagai sumber data simulasi skenario penjadwalan tugas dan alokasi mesin virtual (VM). Nantinya kedua skenario ini akan dilakukan perbandingan untuk mencari tahu skenario mana yang lebih efisien.

TABEL 1
SUMBER DATASET.

No	Nama Algoritma	Dataset 1	Dataset 2
1	Genetic Algorithm	Dataset Acak	Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [13]
2	Genetic Algorithm + Artificial Neural Network	Dataset Acak	Dataset The San Diego Supercomputer Center (SDSC) Blue Horizon logs [13]

B. Skenario Pertama: Genetic Algorithm Saja

Untuk skenario pertama, akan dilakukan percobaan menggunakan *Genetic* Algorithm saja. Pertama-tama akan dilakukan inisialisasi populasi awal dari jadwal sebagai pedoman pengalokasian *Task* kepada mesin virtual (VM) yang sesuai menggunakan *Genetic Algorithm* saja berdasarkan dataset yang digunakan. Untuk spesifikasi dari *Genetic Algorithm* yang akan digunakan pada percobaan ini bisa dilihat pada Tabel 2.

Spesifikasi ini akan mempengaruhi bagaimana *Genetic Algorithm* akan menghasilkan solusi jadwal yang efisien. Besar populasi adalah banyaknya jumlah individu yang berada di dalam satu populasi dalam satu generasi di dalam *Genetic Algorithm* [8]. *Rate* mutasi adalah kemungkinan sebuah gen di dalam kromosom bisa secara acak berubah menjadi gen lain [8]. *Rate* penyilangan adalah kemungkinan diambilnya dua orangtua dari generasi sebelumnya untuk disilangkan gen-nya menjadi satu individu baru di generasi selanjutnya yang memiliki gabungan gen dari kedua orangtuanya [8]. Sedangkan Jumlah elite adalah banyaknya individu yang memiliki nilai *Fitness Function* tinggi dari tiap generasi yang akan dimasukkan secara langsung sebagai anggota di generasi selanjutnya [8].

TABEL 2
SPESIFIKASI GENETIC ALGORITHM.

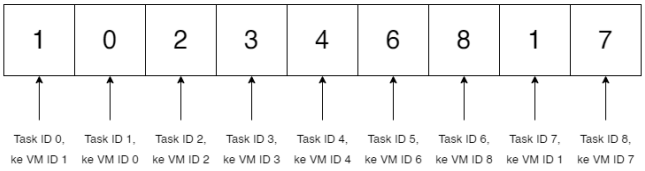
Dataset	Besar Populasi	Rate Mutasi	Rate Penyilangan	Jumlah Elite	Banyak Loop
Dataset Acak	20	30%	95%	2	20
Dataset SDSC	20	30%	95%	2	20

Jadwal ini sendiri akan berbentuk representasi kromosom dari individu di dalam *Genetic Algorithm* sepanjang 9 yang berisikan ID dari mesin virtual (VM) yang dituju oleh *Task.* Representasi kromosom ini bisa dilihat pada Gambar 1.

Kromosom dari tiap individu di dalam populasi akan berisikan angka 0 (nol) hingga 8 (delapan) yang diisikan secara acak dengan duplikasi diperbolehkan. Angka-angka ini merepresentasikan ID Mesin Virtual yang akan menjadi tempat alokasi dari Task.

Jadwal ini akan dilakukan perhitungan *Fitness Function* untuk mencari tahu apakah mereka sudah cukup efisien dalam melakukan penjadwalan tugas dan alokasi mesin

virtual (VM) atau belum berdasarkan *Fitness Function* yang bisa dilihat pada Persamaan 1 hingga Persamaan 3.



Gambar 1. Representasi Kromosom

Disini *Total Execution Time* berarti total waktu yang dihabiskan oleh *Task* di dalam eksekusi dan semakin kecil nilainya maka akan semakin baik sebuah solusi. *Failure Rate* berarti banyaknya *Task* yang gagal diproses per satu satuan waktu dan semakin kecil nilainya maka akan semakin baik sebuah solusi. α dan β adalah persentase seberapa penting nilai *Total Execution Time* dan nilai *Failure Rate* dalam mempengaruhi kualitas solusi dan total nilai keduanya harus sama dengan 1.

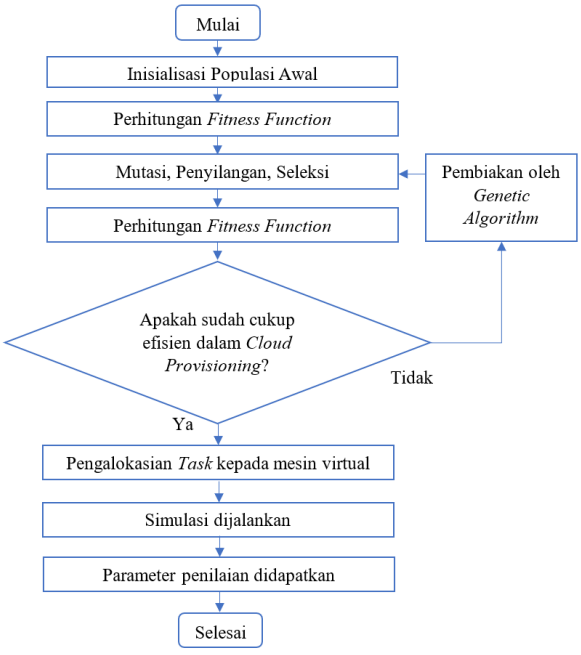
Rumus *Total Execution Time* adalah jumlah total semua panjang *Task* dibagi dengan *MIPS (Million Instruction Per Second)* dari mesin virtual (VM) tempat dia akan diproses. Sedangkan rumus *Failure Rate* adalah banyaknya *Task* yang gagal diproses dibagi dengan lama waktu simulasi.

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

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

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

Jadwal-jadwal ini kemudian akan dilakukan seleksi, mutasi, dan penyilangan menggunakan *Genetic Algorithm* untuk menghasilkan generasi baru yang lebih efisien dari generasi sebelumnya dihitung dari *Fitness Function*-nya. Ketika dirasa tidak ada kenaikan yang signifikan dari *Fitness Function*-nya atau kondisi terminasi sudah tercapai, maka *Genetic Algorithm* akan diberhentikan dan jadwal yang memiliki *Fitness Function* tertinggi akan dijadikan keluaran dari skenario pertama ini.



Gambar 2. *Flowchart* Skenario Pertama

Task bisa dialokasikan kepada mesin virtual (VM) berdasarkan jadwal tersebut dan parameter penilaian bisa didapatkan dari hasil simulasinya. Untuk lebih jelasnya mengenai skenario implementasi pertama bisa melihat pada Gambar 2.

C. Skenario Kedua: Genetic Algorithm dan Artificial Neural Network

Untuk skenario kedua, akan dilakukan percobaan kombinasi dari *Genetic Algorithm* dengan *Artificial Neural Network*. Pertama-tama hasil keluaran jadwal dari skenario pertama akan dilakukan pembagian menjadi 2 yaitu dataset pembelajaran sebanyak 80% dan dataset pengujian sebanyak 20% [5]. Dataset pembelajaran akan dilakukan analisa dan dipelajari terlebih dahulu oleh *Artificial Neural Network* untuk menghasilkan model jadwal yang memiliki akurasi yang sesuai berdasarkan keluaran jadwal dari *Genetic Algorithm*. Spesifikasi dari *Artificial Neural Network* untuk penelitian ini bisa dilihat pada Tabel 3.

TABEL 3
SPESIFIKASI ARTIFICIAL NEURAL NETWORK

Dataset	Input Node	Hidden Node	Output Node	Banyak Epoch	Akurasi
Dataset Acak	9	18	9	100000	88%
Dataset *SDSC*	9	18	9	100000	88%

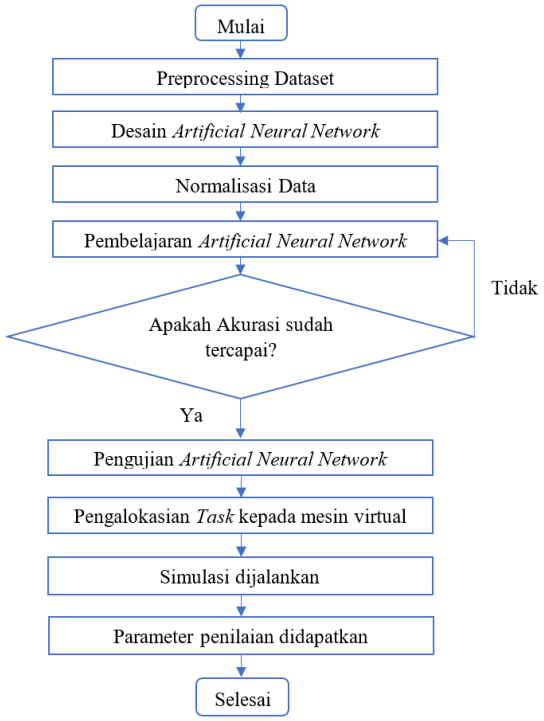
Spesifikasi 9 *Input Node* berguna untuk menerima masukkan sebanyak 9 panjang *Task* yang nantinya akan menjadi informasi awal bagi *Artificial Neural Network* untuk mengeluarkan keluaran sebanyak 9 ID VM pada *Output Node*-nya. Di antara *Input Node* dan *Output Node* akan ada 1 *Hidden Node* berisi 18 *Node* sebagai penengah antara keduanya. *Hidden Node* ini akan menggunakan *Activation Function ReLu* dan *Output Node* akan menggunakan *Activation Function Sigmoid* [14]. Lebih jelasnya bisa melihat pada Tabel 4.

TABEL 4
SPESIFIKASI NODE ARTIFICIAL NEURAL NETWORK

Dataset	Input	Output	Activation Function Pada Hidden Node	Activation Function Pada Output Node
Dataset Acak	9 Panjang *Task*	9 ID VM	*ReLu*	*Sigmoid.*
Dataset *SDSC*	9 Panjang *Task*	9 ID VM	*ReLu*	*Sigmoid.*

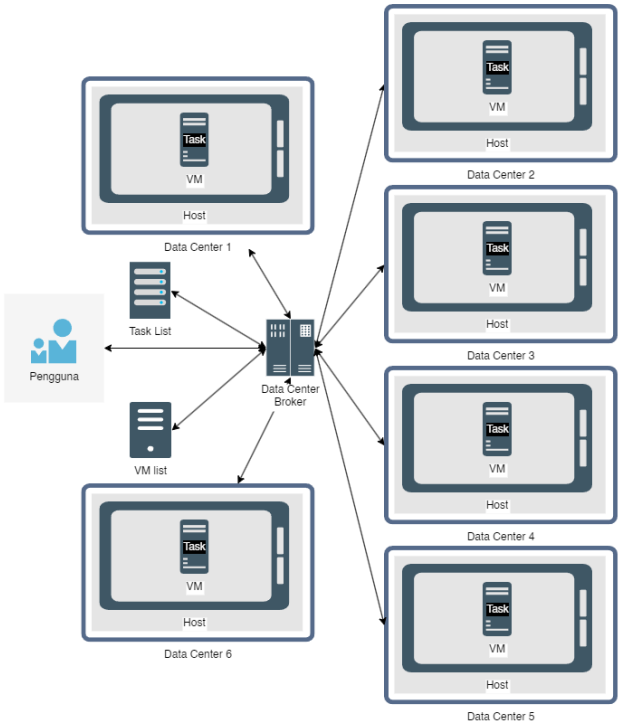
Propagasi *Manhattan* digunakan pada *Artificial Neural Network* ini agar penulis bisa mengkontrol secara langsung seberapa besar pengurangan *weight* pada *Artificial Neural Network* sehingga penulis bisa mendapatkan tingkat akurasi yang sesuai. Hal ini dikarenakan Propagasi *Manhattan* menggunakan sebuah nilai yang sangat kecil untuk mengurangi ataupun menambahkan *weight*, tidak seperti metode propagasi biasa yang menggunakan *Gradient* sehingga perubahan nilai pada *weight* terlalu besar dan akurasi yang dihasilkan tidak bisa penulis kontrol [14]. Akurasi yang penulis tuju pada percobaan ini adalah 88%,

karena jika terlalu tinggi akan mengakibatkan *Overfit* dan jika terlalu kecil akan mengakibatkan *Underfit*. Untuk lebih jelasnya mengenai skenario implementasi kedua bisa melihat pada Gambar 3.



Gambar 3. *Flowchart* Skenario Kedua

D. *Lingkungan Implementasi*



Gambar 4. Skema Implementasi Perangkat Lunak

Implementasi dari perangkat lunak akan menggunakan *CloudSIM*. Sebuah kerangka kerja *Open Source*, yang digunakan untuk mensimulasikan layanan *Cloud Computing*. Perangkat lunak ini akan dilakukan implementasi kepada 54 *Virtual Machine* yang berada di dalam 18 *Host*, yang berada di dalam 6 *Data Center*. Masing-masing *Data Center* ini akan terhubung kepada sebuah *Data Center Broker* yang berguna sebagai otak penjadwalan tugas dan alokasi mesin virtual (VM). *Data Center Broker* ini akan terhubung ke *VM List* sebagai daftar *Virtual Machine* yang ada dan status mereka, ke *Task List* sebagai daftar *Task* yang ada dan status mereka, dan terakhir ke pengguna sebagai entitas yang memberikan *Task* dan menerima *Output* hasil pengerjaan. Skema implementasi ini bisa dilihat pada Gambar 4 dan Gambar 5.

Gambar 5. Skema Implementasi Perangkat Lunak Pembagian VM

Setiap *Virtual Machine* dan *Data Center* akan memiliki spesifikasi mereka masing-masing agar penulis bisa melihat hasil yang heterogen. Spesifikasi dari *Virtual Machine*, *Host*, dan *Data Center* ini bisa dilihat secara detail pada Tabel 5, Tabel 6, dan Tabel 7.

TABEL 5
SPESIFIKASI VIRTUAL MACHINE

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

TABEL 6
SPESIFIKASI HOST

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

TABEL 7
SPESIFIKASI DATA CENTER

ID	Jumlah Prosesor	Jumlah Host	Bandwith (MBps)	Latency (ms)	ID
D1	6	3	10000	6	D1
D2	6	3	10000	6	D2
D3	6	3	10000	8	D3

E. *Parameter Penilaian Yang Digunakan*

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

perbandingan untuk mencari tahu mana sistem *Cloud Provisioning* yang lebih efisien. Perbandingan tersebut akan didasarkan pada beberapa parameter sesuai dengan Tabel 8.

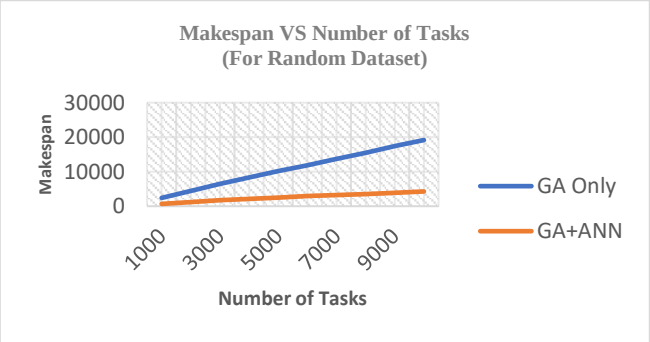
TABEL 8
PARAMETER PENILAIAN.

No	Parameter	Rumus
1	Makespan	$max_{i \epsilon tasks}\{F_i\}$
2	Average Start Time	$\frac{\sum_{i=1}^{n} Waktu\ Ri\ Mulai}{nR}$
3	Average Finish Time	$\frac{\sum_{i=1}^{n} Waktu\ Ri\ Selesai}{nR}$
4	Average Execution Time	$\frac{\sum_{i=1}^{n} Waktu\ Ti}{nT}$
5	Total Wait Time	$Total\ Delay$
6	Scheduling Length	$Scheduling\ Time + Makespan$
7	Throughput	$\frac{nT}{Makespan}$
8	Resource Utilization	$\frac{\sum_{i=1}^{n} Waktu\ Ri\ Selesai}{Makespan * nR}$
9	Energy Consumption	$Total\ Energy\ Consumption$
10	Imbalance Degree	$\frac{ETmax - ETmin}{ETavg}$

IV. HASIL PENGUJIAN

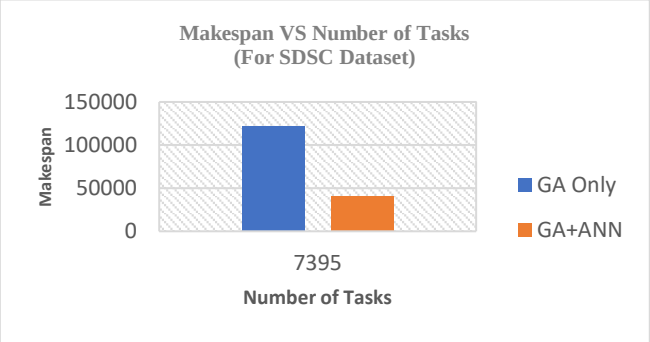
A. *Makespan*

Dari Gambar 6 bisa dilihat bahwa skenario kedua, yaitu implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* memiliki nilai yang lebih rendah secara konsisten dengan adanya pertambahan jumlah *Task*. Hal ini berarti *Task* terakhir yang ada di dalam simulasi skenario kedua lebih cepat diselesaikan oleh implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* dibandingkan dengan implementasi *Genetic Algorithm* saja.



Gambar 6. Grafik Perbandingan *Makespan* Pada Dataset Acak

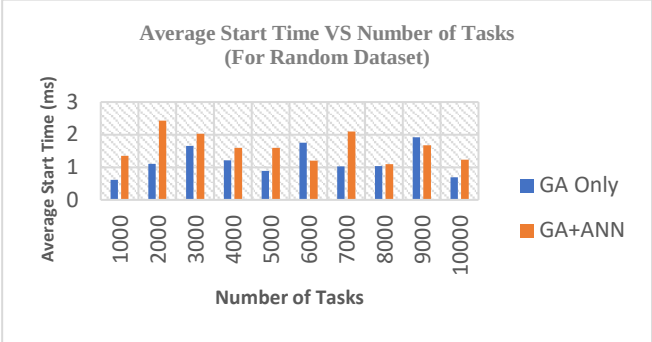
Pada dataset SDSC di Gambar 7, hasil yang sama bisa dilihat dimana implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai *Makespan* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* saja.



Gambar 7. Grafik Perbandingan *Makespan* Pada Dataset SDSC

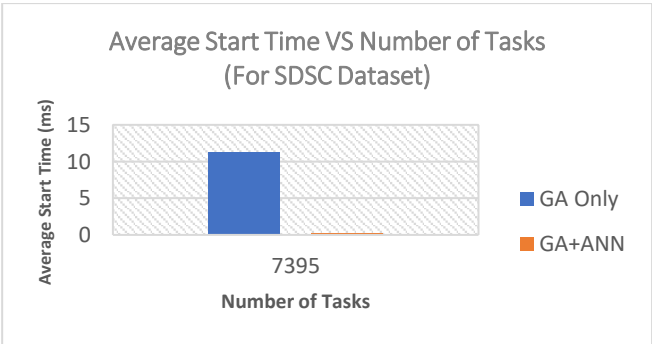
B. *Average Start Time*

Pada Gambar 8, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak. Hal ini berarti implementasi skenario pertama bisa memulai pemrosesan *Task* lebih cepat daripada implementasi skenario kedua.



Gambar 8. Grafik Perbandingan *Average Start Time* Pada Dataset Acak

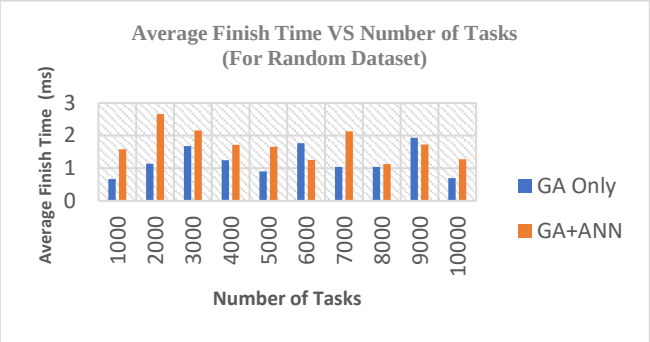
Namun ada hal yang menarik ketika melihat Gambar 9 Disini terlihat bahwa *Average Start Time* implementasi skenario kedua, yaitu *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai yang lebih rendah. Disini berarti bahwa implementasi skenario kedua bisa memulai pemrosesan *Task* lebih cepat daripada implementasi skenario pertama. Hal ini diakibatkan lebih besar dan beragamnya rentang data dari dataset SDSC dibandingkan dengan dataset acak. Dari sini bisa disimpulkan bahwa *Genetic Algorithm* lebih baik saat menghadapi dataset yang rentang dan ragam datanya tidak terlalu besar dan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* bisa digunakan untuk menghadapi dataset yang yang rentang dan ragam datanya besar.



Gambar 9. Grafik Perbandingan *Average Start Time* Pada Dataset SDSC

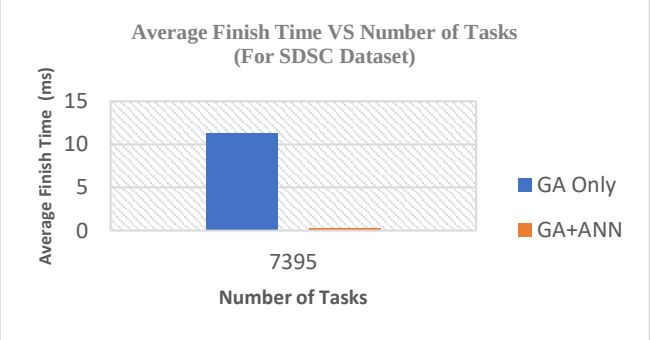
C. *Average Finish Time*

Pada Gambar 10, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak. Hal ini berarti implementasi skenario pertama bisa menyelesaikan pemrosesan *Task* lebih cepat daripada implementasi skenario kedua.



Gambar 10. Grafik Perbandingan *Average Finish Time* Pada Dataset Acak

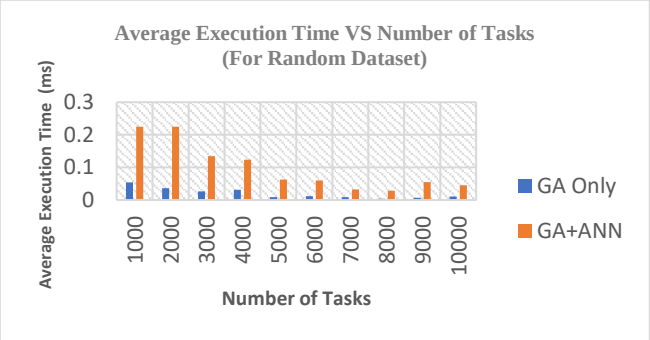
Namun ada hal yang menarik ketika melihat Gambar 11. Disini terlihat bahwa *Average Finish Time* implementasi skenario kedua, yaitu *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai yang lebih rendah. Disini berarti bahwa implementasi skenario kedua bisa menyelesaikan pemrosesan *Task* lebih cepat daripada implementasi skenario pertama. Hal ini diakibatkan lebih besar dan beragamnya rentang data dari dataset SDSC dibandingkan dengan dataset acak. Dari sini bisa disimpulkan bahwa *Genetic Algorithm* lebih baik saat menghadapi dataset yang rentang dan ragam datanya tidak terlalu besar dan *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* bisa digunakan untuk menghadapi dataset yang yang rentang dan ragam datanya besar.



Gambar 11. Grafik Perbandingan *Average Finish Time* Pada Dataset SDSC

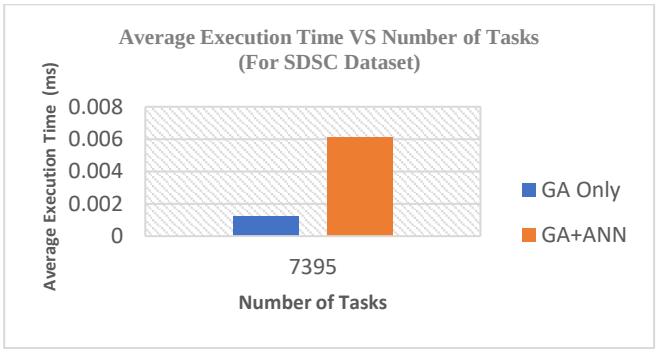
D. *Average Execution Time*

Pada Gambar 12, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak. Hal ini berarti implementasi skenario pertama lebih baik dalam menghemat waktu yang *Task* habiskan saat pemrosesan daripada implementasi skenario kedua.



Gambar 12 Grafik Perbandingan *Average Execution Time* Pada Dataset Acak

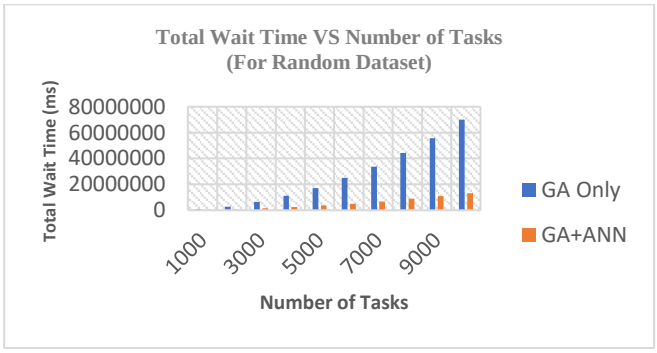
Pada dataset SDSC di Gambar 13, hasil yang sama bisa dilihat dimana implementasi *Genetic Algorithm* saja menghasilkan nilai *Average Finish Time* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Disini bisa disimpulkan bahwa implementasi *Genetic Algorithm* saja lebih baik dalam menghemat waktu yang *Task* habiskan saat pemrosesan daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*.



Gambar 13 Grafik Perbandingan *Average Execution Time* Pada Dataset SDSC

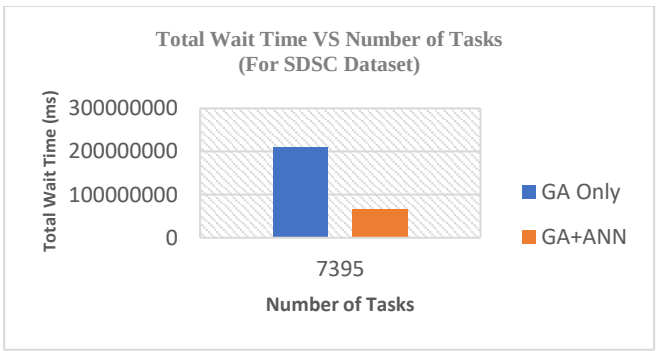
E. *Total Wait Time*

Pada Gambar 14, bisa dilihat bahwa implementasi skenario kedua, yaitu *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* saja pada dataset acak.



Gambar 14 Grafik Perbandingan *Total Wait Time* Pada Dataset Acak

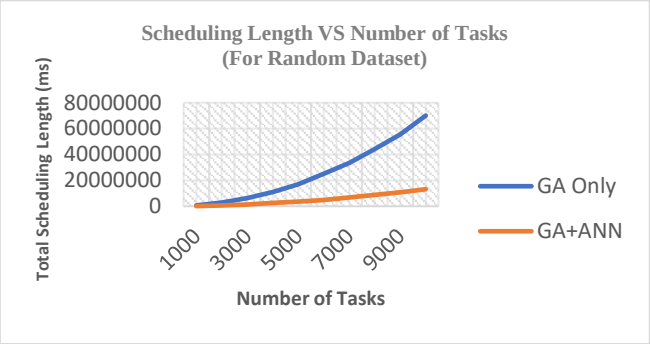
Pada dataset SDSC di Gambar 15, hasil yang sama bisa dilihat dimana implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai *Total Wait Time* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* saja. Disini bisa disimpulkan bahwa implementasi skenario kedua memang lebih baik dalam mengurangi waktu *Delay Task*.



Gambar 15 Grafik Perbandingan *Total Wait Time* Pada Dataset SDSC

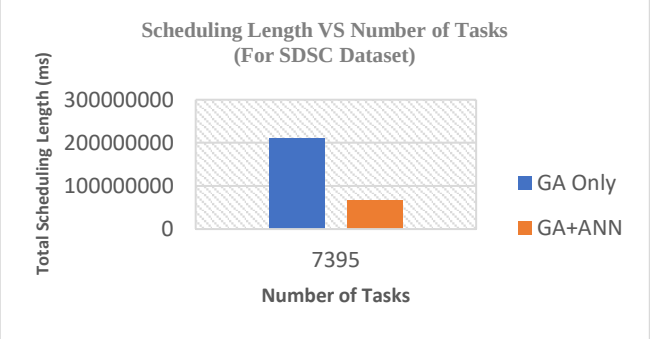
F. *Scheduling Length*

Pada Gambar 16, bisa dilihat bahwa implementasi skenario kedua, yaitu *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* saja pada dataset acak.



Gambar 16 Grafik Perbandingan *Scheduling Length* Pada Dataset Acak

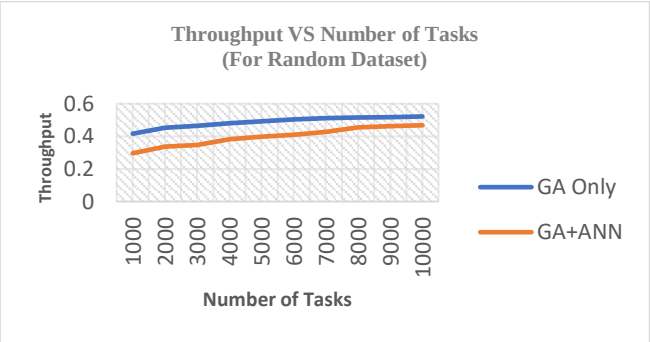
Pada dataset SDSC di Gambar 17, hasil yang sama bisa dilihat dimana implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai *Scheduling Length* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* saja. Disini bisa disimpulkan bahwa implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* memang lebih baik dalam menghemat waktu simulasi *Cloud* daripada implementasi *Genetic Algorithm* saja.



Gambar 17 Grafik Perbandingan *Scheduling Length* Pada Dataset SDSC

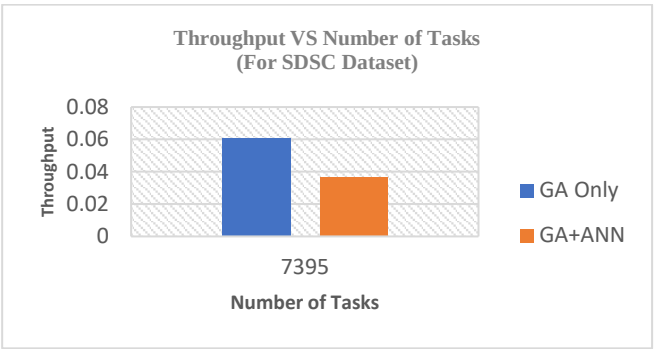
G. *Throughput*

Pada Gambar 18, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih tinggi daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak.



Gambar 18 Grafik Perbandingan *Throughput* Pada Dataset Acak

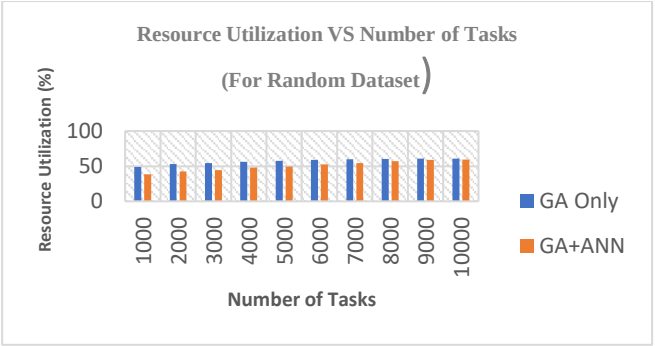
Pada dataset SDSC di Gambar 19, hasil yang sama bisa dilihat dimana implementasi *Genetic* Algorithm saja menghasilkan nilai *Throughput* yang lebih tinggi dibandingkan dengan implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*.



Gambar 19 Grafik Perbandingan *Throughput* Pada Dataset SDSC

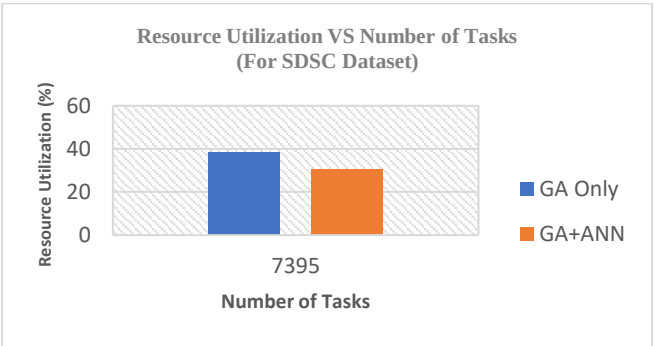
H. *Resource Utilization*

Pada Gambar 20, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih tinggi daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak.



Gambar 20 Grafik Perbandingan *Resource Utilization* Pada Dataset Acak

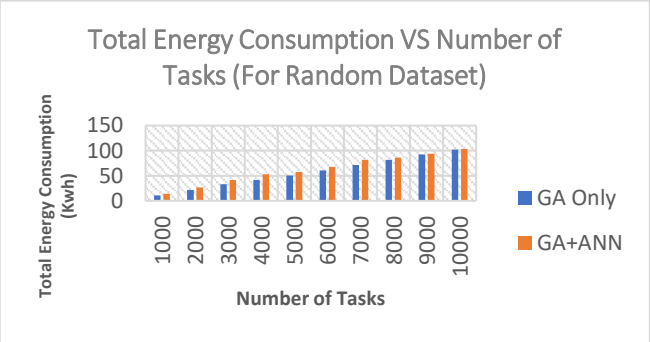
Pada dataset SDSC di Gambar 21, hasil yang sama bisa dilihat dimana implementasi *Genetic* Algorithm saja menghasilkan nilai *Resource Utilization* yang lebih tinggi dibandingkan dengan implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*.



Gambar 21 Grafik Perbandingan *Resource Utilization* Pada Dataset SDSC

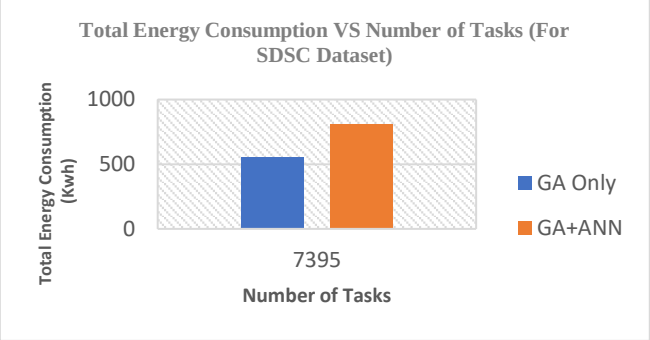
I. *Resource Utilization*

Pada Gambar 22, bisa dilihat bahwa implementasi skenario pertama, yaitu *Genetic Algorithm* saja, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* pada dataset acak.



Gambar 22 Grafik Perbandingan *Total Energy Consumption* Pada Dataset Acak

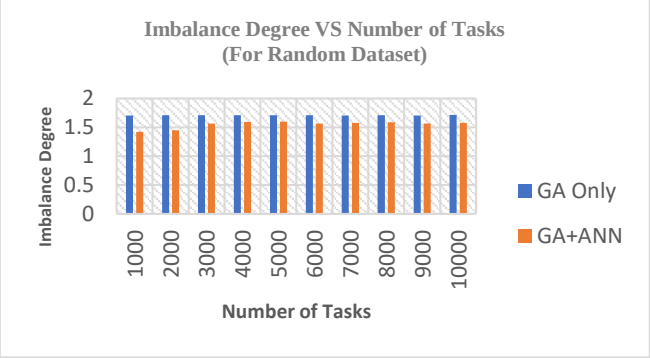
Pada dataset SDSC di Gambar 23, hasil yang sama bisa dilihat dimana implementasi *Genetic* Algorithm saja menghasilkan nilai *Total Energy Consumption* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*. Disini bisa disimpulkan bahwa implementasi *Genetic Algorithm* saja memang lebih baik dalam menghemat penggunaan energi daripada implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*.



Gambar 23 Grafik Perbandingan *Total Energy Consumption* Pada Dataset SDSC

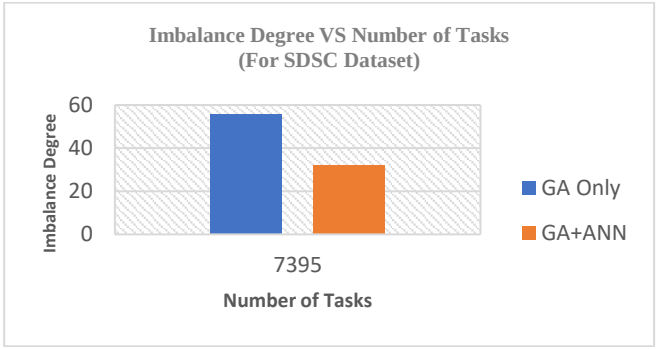
J. Imbalance Degree

Pada Gambar 24, bisa dilihat bahwa implementasi skenario kedua, yaitu *Genetic Algorithm* bersamaan dengan *Artificial Neural Network*, bisa menghasilkan nilai yang secara konsisten lebih rendah daripada implementasi *Genetic Algorithm* saja pada dataset acak.



Gambar 24 Grafik Perbandingan *Imbalance Degree* Pada Dataset Acak

Pada dataset SDSC di Gambar 25, hasil yang sama bisa dilihat dimana implementasi *Genetic Algorithm* bersamaan dengan *Artificial Neural Network* menghasilkan nilai *Imbalance Degree* yang lebih rendah dibandingkan dengan implementasi *Genetic Algorithm* saja.



Gambar 25 Grafik Perbandingan *Imbalance Degree* Pada Dataset SDSC

V. KESIMPULAN

Implementasi *Genetic Algorithm* dalam sistem penjadwalan menghasilkan tingkat pemanfaatan sumber daya yang lebih tinggi dibandingkan dengan sistem tanpanya, dengan tingkat berkisar antara 48% hingga 60%. Ketika dikombinasikan dengan *Artifical Neural Network*, tingkat pemanfaatan sumber daya berkisar antara 38% hingga 59%. *Genetic Algorithm* sendiri berperforma lebih baik dalam konservasi energi dan menyelesaikan tugas dalam jangka waktu yang lebih singkat, sedangkan kombinasi keduanya berperforma lebih baik dalam menangani ketidakseimbangan data dan menyelesaikan simulasi *Cloud* dengan cepat.

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

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

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

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

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

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

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

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

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

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

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

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