MANAJEMEN DAN KENDALI BEBAN PERANGKAT ELEKTRONIK BERBASIS WEB DENGAN ALGORITMA GENETIKA

WEB-BASED MANAGEMENT AND CONTROLLING FOR ELECTRONIC DEVICE LOADS WITH *GENETIC ALGORITHM*

Willy Mochamad Ilham[1], Randy Erfa Saputra, S.T., M.T.[2], Casi Setianingsih, S.T., M.T.[3]
[1,2,3]Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]**willymilham@telkomuniversity.ac.id**, [2]**resaputra@telkomuniversity.ac.id**,
[3]**setiacasie@telkomuniversity.ac.id**

Abstrak
Dewasa ini, energi listrik menjadi salah satu kebutuhan primer dalam kehidupan sehari-hari. Setiap perangkat yang digunakan hampir seluruhnya memerlukan energi listrik untuk bisa digunakan. Hanya saja, jika kurangnya kesadaran pengguna dalam menggunakan perangkat elektronik dapat menyebabkan tagihan listrik bulanan membengkak terutama pada wilayah perkantoran. Untuk meingkatkan efisiensi dalam penggunaan energi listrik pada perkantoran, maka dibuatlah sebuah sistem manajemen dan mengatur beban elektronik sehari-hari dengan berbasis web yang dapat menentukan target tagihan listrik setiap bulannya. Dalam sistem ini, prioritas perangkat elektronik juga dapat ditentukan dan di teruskan ke dalam sebuah algoritma optimasi menggunakan metode algoritma genetika untuk sistem efisiensinya dan *database* untuk menyimpan data pengguna. Nilai *fitness* optimal berada di generasi ke-60 yang didapatkan dari pengujian nilai *fitness* optimal. Dari hasil pengujian waktu eksekusi, yang menunjukan rata-rata waktu eksekusi 0,18 detik, serta pengujian *rules*, yang mendapatkan tingkat akurasi 100%, ini menunjukan bahwa sistem berjalan dengan sesuai dengan *rules* yang dirancang.

Kata kunci : *Web, Database, Manajemen Energi, Algoritma Genetika*.

Abstract
Nowadays, electrical energy is one of the primary needs in everyday life. Every device that is used almost entirely requires electrical energy to be used. However, the lack of awareness in using electronic devices can cause monthly electrical bills uncontrolled, especially in office areas. To increase the electrical energy efficiencies on the offices, a web-based management system and control of daily electronic expenses is created that can determine the monthly electricity bill target. In this system, the priority of electronic devices can also be determined and forwarded into an optimization algorithm using the Genetetic Algorithm method for system efficiency and keep user data in the database. The optimal fitness value is on 60th generation, based on optimal fitness value. Based on the results of execution time testing that indicates the average time for execution is 0,18 seconds, and from rules validation testing, the application get a 100% accuracy rate, this indicates that the application is running properly.

Keywords: *Web, Database, Power Management, Genetetic Algorithm.*

1. Pendahuluan

Energi listrik sudah menjadi salah satu kebutuhan bagi masyarakat di zaman modern ini, setiap perangkat yang digunakan dalam kehidupan sehari-hari hampir seluruhnya memerlukan energi listrik untuk bisa digunakan. Umumnya, penggunaan perangkat-perangkat elekteronik ini masih dikendalikan secara manual. Untuk menyalakan dan mematikan alat elektronik, pengguna harus berinteraksi langsung dengan alat tersebut, seperti menekan tombol *power* maupun memasangkan atau melepaskan kabel ke stop kontak. Namun, karena kelalaian pengguna, terkadang beberapa alat elektronik yang sedang tidak digunakan lupa untuk tidak dimatikan. Hal ini, tanpa disadari dapat menyebabkan menaiknya tagihan listrik bulanan ataupun terbuangnya kuota listrik secara percuma[1][2].

Berdasarkan itu, suatu sistem dapat dibuat untuk memanajemen, memonitoring, serta mengendalikan penggunaan energi listrik dengan menggunakan suatu prioritas pada alat elektronik. Sehingga, penggunaan energi listrik lebih efisien dan target tagihan listrik bulanan dapat terpenuhi. Dalam penelitian ini, penulis membuat sistem tersebut dari sisi perangkat lunak bebasis web yang

memungkinkan penggunanya untuk memanajemen, memonitoring, serta mengendalikan penggunaan energi listrik secara otomatis dengan menggunakan algoritma genetika. Hasil dari metode ini akan dikirimkan ke antares.

Di Indonesia metode pembayaran tagihan listrik terdapat dua jenis, yaitu pra-bayar dan pasca-bayar. Pada metode tagihan pasca-bayar konsumen diharuskan membayar tagihan listrik setiap bulannya. Untuk melihat berapa kilowatt listrik yang telah digunakan konsumen dapat melihatnya pada KWh meter.

2. Dasar Teori

2.1 Algortima Genetika

Algoritma genetika merupakan salah satu algoritma optimasi yang digunakan untuk mendapatkan sebuah solusi. Algoritma ini dapat diimplementasikan dalam berbagai masalah dan dapat memberikan hasil yang lebih baik untuk setiap iterasinya hingga mendekati optimal[3].

2.1.1 Inisialisasi Populasi

Implementasi algoritma genetika dimulai dengan menginisialisasi populasi dari kromosom yang biasanya acak. Inisialisasi populasi yaitu proses menentukan jumlah dan panjang kromosom dalam populasi yang akan digunakan[4][5]. Untuk penelitian ini dimana pada satu kromosom memiliki jumlah gen yang tergantung dari banyaknya jumlah data perangkat elektronik, bisa dirumuskan dengan rumus berikut:

$$P = K \times jD \quad (1)$$

Keterangan :
P = Populasi
K = Kromosom dengan panjang sama dengan jumlah data perangkat elektronik
jD = Jumlah data perangkat elektronik

Kemudian untuk mendapatkan nilai *threshold* algoritma genetika, digunkan jumlah tagihan yang telah diubah menjadi bentuk kWh dan hari yang diinginkan untuk dapat menghabiskan jumlah kWh tersebut yang telah di*input*kan, dengan rumus sebagai berikut:

$$Th = Pc \div H \quad (2)$$

Keterangan :
Th = *Threshold*
Pc = Jumlah tagihan yang telah diubah kedalam kWh
H = Jumlah hari

2.1.2 Evaluasi Kromosom

Evaluasi kromosom merupakan proses menghitung nilai *fitness* yang didapatkan dari menjumlahkan *value* (nilai) dari setiap kromosom dengan rumus berikut[4][5]:

$$Fitness = \sum_{i=1}^{n} GiVi \quad (3)$$

Keterangan :
n = panjang kromosom
Gi = gen ke-i
Vi = *value* ke-i

2.1.3 Seleksi Kromosom

Setelah itu, dilakukan seleksi kromosom untuk mendapatkan kromosom yang berpotensi untuk menjadi *parents* untuk proses berikutnya. Seleksi kromosom dapat dilakukan dalam beberapa cara, yaitu[4][5]:

1. *Roulette Wheel Selection,* metode seleksi ini berdasarkan probabilitas setiap kromosom. Gen dalam kromosom akan sangat bervariasi namun tergantung dari nilai fitnessnya. Seleksi dilakukan dengan membangkitkan nilai acak dari jarak semua nilai *fitness* dari populasi yang telah dievaluasi.
2. *Tournament Selection,* metode seleksi ini mendapatkan *parents* yang terbaik secara acak yang sangat bergantung pada nilai *fitness*. Seleksi dimulai dengan memunculkan nilai *fitness* dari populasi yang telah dievaluasi, kemudian dipilih kromosom dengan nilai *fitness* terbesar.

Pada penelitian ini, proses seleksi dilakukan menggunakan *tournament selection*. Untuk mendapatkan *parents* tersebut, populasi awal akan dibagi menjadi 2 dan dilakukan pengecekan nilai *fitness*, kromosom-kromosom dengan nilai *fitness* yang lebih besar dari kromosom lainnya akan dipilih sebagai *parents.*

2.1.4 Crossover

Hasil dari seleksi kromosom dijadikan *parents* untuk proses kawin silang (*crossover*). Proses ini akan menghasilkan dua kromosom baru dari setiap pasangan orang tua (*parents*). *Crossover* juga memiliki berbagai metode, yaitu[4][5]:

1. *One Point Crossover*, menukar gen dari setiap pasangan *parents* dengan satu titik persimpangan.
2. *Multi-point Crossover*, menukar gen dari setiap pasangan *parents* dengan beberapa titik persimpangan.

Crossover dilakukan antara dua kromosom dan hasilnya adalah dua kromosom baru. Dalam tahap ini terdapat probabilitas *crossover* yang digunakan untuk menentukan apakah *crossover* perlu dilakukan atau tidak. Nilai dari probabilitas ini bebas karena tidak ada kepastian mengenai probabilitas terbaik. Terdapat nilai acak yang akan menjadi acuan untuk iterasi *crossover*, proses ini akan berhenti jika nilai acak yang dibangkitkan lebih kecil dari nilai probabilitas *crossover*.

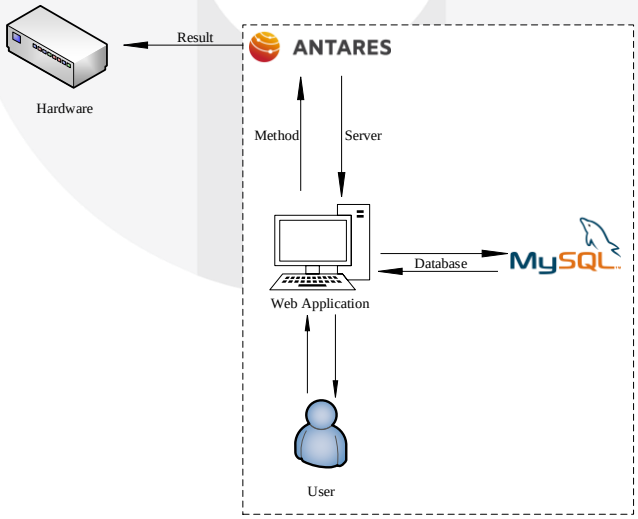
2.1.5 Mutasi

Setelah proses *crossover*, dilakukan proses mutasi dengan menukar gen didalam kromosom dengan gen sebelahnya jika tidak memenuhi syarat[4][5]. Mutasi mirip dengan *crossover*, mutasi tidak selalu dilakukan dan terdapat juga probabilitas mutasi yang tidak ada ketentuan yang pasti mengenai nilainya. Pada penelitian ini, proses yang dilakukan adalah mengubah nilai gen, untuk memastikan tidak ada gen yang berisi nilai 0 , dilakukan penggantian gen yang memiliki nilai 0 dengan nilai acak dengan jarak 1 sampai dengan 24.

3. Metodologi Penelitian

3.1 Gambaran Umum Sistem

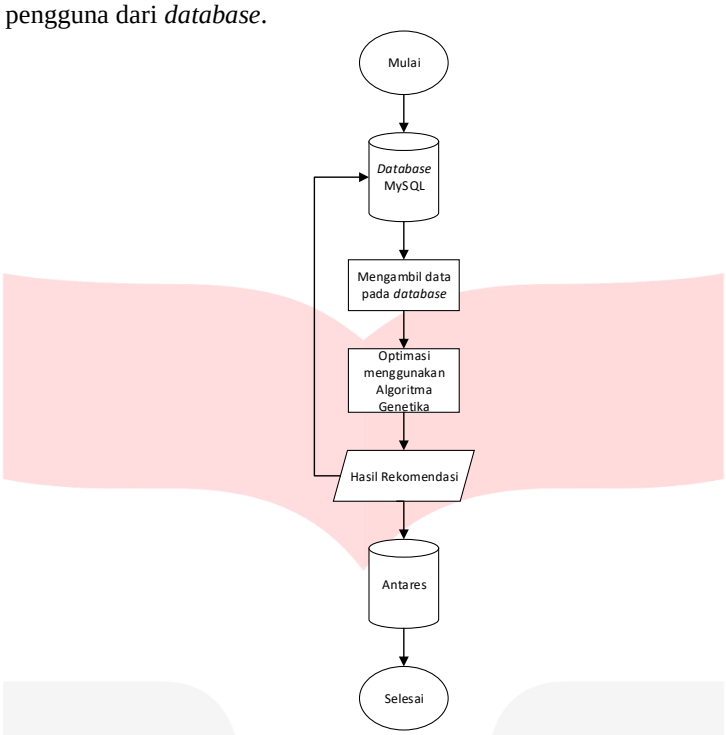
Sistem yang dibuat berupa *website* untuk kendali terhadap perangkat elektronik yang digunakan. Metode yang digunakan yaitu Algoritma Genetika untuk memperhitungkan dan akan mengeluarkan solusi dari parameter yang telah di*input*kan oleh pengguna dalam *website*. Metode ini akan memberikan rekomendasi penggunaan jam untuk setiap harinya untuk setiap perangkat elektronik yang telah di*input*kan untuk memenuhi jumlah tagihan listrik yang diinginkan oleh pengguna seperti yang terlihat pada gambar 1. Berdasarkan itu, penulis hanya merancang sistem sampai mengirim data ke Antares dan tidak diteruskan ke perangkat keras:



Gambar 1 Gambaran umum sistem

3.2 Perancangan Sistem

Dalam diagram alir sistem pada gambar 2 dibawah, menunjukan alur kerja sistem. Sistem ini bermula dengan mengambil data perangkat dan data jumlah tagihan serta untuk berapa harinya yang diinginkan oleh pengguna dari *database*.



Gambar 2 Flowchart sistem

Kemudian akan di optimasi menggunakan Algoritma Genetika sehingga mengeluarkan rekomendasi penggunaan jam untuk setiap harinya untuk setiap perangkat elektronik. Hasil optimasi ini disimpan kedalam *database*, dan data ini akan di tampilkan pada halaman *website* dan di unggah ke dalam *server* Antares.

Tabel 1 Tabel rules jarak nilai gen

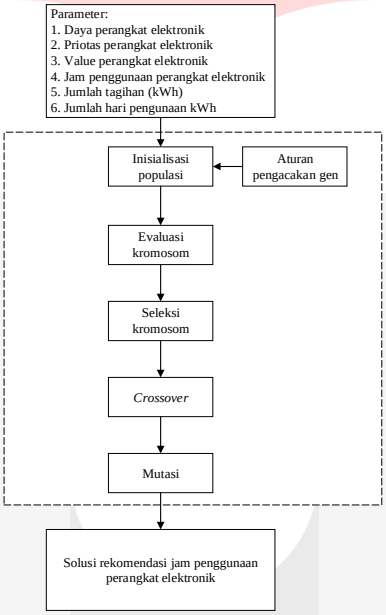
Indikator	Varibel Indikator	Rule	Variabel Rule
Device P1	A	IF A AND K THEN M	A1
Device P2	B	IF B AND K THEN M	A2
Device P3	C	IF C AND K THEN M	A3
Device P4	D	IF D AND K THEN M	A4
Device P5	E	IF E AND K THEN M	A5
Device P6	F	IF F AND K THEN M	A6
Device P7	G	IF G AND K THEN M	A7
Device P8	H	IF H AND K THEN M	A8
Device P9	I	IF I AND K THEN M	A9
Device P10	J	IF J AND K THEN M	A10
Jam 24	K	IF A AND L THEN N	A11
Jam Auto	L	IF B AND L THEN O	A12
Nilai gen 24	M	IF C AND L THEN P	A13
Jarak nilai gen acak 18 - 24	N	IF D AND L THEN Q	A14
Jarak nilai gen acak 16 - 22	O	IF E AND L THEN R	A15
Jarak nilai gen acak 14 - 20	P	IF F AND L THEN S	A16

Indikator	Varibel Indikator	Rule	Variabel Rule
Jarak nilai gen acak 12 - 18	Q	IF G AND L THEN T	A17
Jarak nilai gen acak 10 - 16	R	IF H AND L THEN U	A18
Jarak nilai gen acak 8 - 14	S	IF I AND L THEN V	A19
Jarak nilai gen acak 6 - 12	T	IF J AND L THEN W	A20
Jarak nilai gen acak 4 - 10	U		
Jarak nilai gen acak 2 - 8	V		
Jarak nilai gen acak 1 - 6	W		

Aturan (*rules*) pada tabel 1 ini digunakan untuk menentukan jarak nilai acak yang nantinya akan menjadi isi dari gen, yang digunakan dalam inisialisasi populasi untuk Algoritma Genetika. Berdasarkan tabel diatas, terdapat dua puluh rules yang dihasilkan dari dua indikator yang digunakan untuk menentukan jarak nilai acak isi gen.

4. Hasil Pengujian Sistem

4.1 Algoritma Gentika



Gambar 3 Tahapan membangun algoritma genetika

Tahapan dalam membangun Algoritma Genetika dapat dilihat pada gambar 3, yang menunjukan bahwa Algoritma Genetika memiliki lima tahapan utama, yaitu inisialisasi populasi, evaluasi kromosom, seleksi kromosom, *crossover*, dan mutasi.

Sistem ini bermula dengan mengambil data perangkat berupa nama perangkat, daya (watt), jumlah perangkat, jam penggunaan, dan prioritasnya. Selain itu, sistem juga akan mengambil data pascabayar berupa data jumlah tagihan serta untuk berapa harinya yang diinginkan oleh pengguna dari database. Untuk inisialisasi populasi, kromosom diisi oleh bilangan acak, namun menggunakan ketentuan berdasarkan jam dan prioritas yang telah diinputkan untuk setiap perangkat elektronik. Jika jam untuk perangkat elektronik di*input*kan 24 (jam) maka gen untuk perangkat tersebut berisi 24, tapi jika jam perangkat elektronik di*input*kan auto maka gen untuk device tersebut akan berisi nilai acak dengan ketentuan seperti berikut :

1. *Device* prioritas 1 = nilai acak mulai dari 18 sampai dengan 24
2. *Device* prioritas 2 = nilai acak mulai dari 16 sampai dengan 22
3. *Device* prioritas 3 = nilai acak mulai dari 14 sampai dengan 20
4. *Device* prioritas 4 = nilai acak mulai dari 12 sampai dengan 18
5. *Device* prioritas 5 = nilai acak mulai dari 10 sampai dengan 16
6. *Device* prioritas 6 = nilai acak mulai dari 8 sampai dengan 14

7. *Device* prioritas 7 = nilai acak mulai dari 6 sampai dengan 12
8. *Device* prioritas 8 = nilai acak mulai dari 4 sampai dengan 10
9. *Device* prioritas 9 = nilai acak mulai dari 2 sampai dengan 8
10. *Device* prioritas 10 = nilai acak mulai dari 1 sampai dengan 6

Device yang dimaksudkan bersifat fleksibel, dapat berupa alat elektronik apa saja yang perlu terhubung dengan listrik, namun pada penelitian ini *device* yang di*input*kan secara berurutan, yaitu server, komputer, mesin fotocopy, printer, pompa air, AC, amplifier, speaker, lampu, dan dispenser. Pada bagian evaluasi kromosom, *value* yang digunakan untuk menghitung nilai *fitness* ini berisi nilai 1000 sampai dengan 100, dengan ketentuan seperti berikut :

1. *Device* prioritas 1 = *value* 1000
2. *Device* prioritas 2 = *value* 900
3. *Device* prioritas 3 = *value* 800
4. *Device* prioritas 4 = *value* 700
5. *Device* prioritas 5 = *value* 600
6. *Device* prioritas 6 = *value* 500
7. *Device* prioritas 7 = *value* 400
8. *Device* prioritas 8 = *value* 300
9. *Device* prioritas 9 = *value* 200
10. *Device* prioritas 10 = *value* 100

Berikut adalah perhitungan manual dari Algoritma Genetika yang akan dimulai dari inisialisasi populasi, evaluasi kromosom, seleksi kromosom, *crossover*, dan mutasi. Dimana nilai *input* yang akan digunakan sebagai berikut:

1. *Device* 1 : Daya = 0.9 kWh, Prioritas = 1, *Value* = 1000, Jam = Auto
2. *Device* 2 = Daya = 1 kWh, Prioritas = 2, *Value* = 900, Jam = Auto
3. *Device* 3 = Daya = 1.2 kWh, Prioritas = 3, *Value* = 800, Jam = Auto
4. *Device* 4 = Daya = 0.05 kWh, Prioritas = 4, *Value* = 700, Jam = Auto
5. *Device* 5 = Daya = 0.125 kWh, Prioritas = 5, *Value* = 600, Jam = Auto
6. *Device* 6 = Daya = 0.6 kWh, Prioritas = 6, *Value* = 500, Jam = Auto
7. *Device* 7 = Daya = 0.23 kWh, Prioritas = 7, *Value* = 400, Jam = Auto
8. *Device* 8 = Daya = 0.09 kWh, Prioritas = 8, *Value* = 300, Jam = Auto
9. *Device* 9 = Daya = 0.15 kWh, Prioritas = 9, *Value* = 200, Jam = Auto
10. *Device* 10 = Daya = 0.3 kWh, Prioritas = 10, *Value* = 100, Jam = Auto
11. Jumlah Tagihan = Rp.1,000,000 (681.53320 kWh)
12. Hari = 7

Untuk membangkitkan populasi awal pertama dilakukan adalah menentukan jumlah kromosom yang akan digunakan sebagai berikut:

$$P = K \times jD$$

P = 10 data perangkat χ 10 data perangkat

P = 100 gen

Sesuai dengan perhitungan jumlah gen diatas, maka terdapat 100 gen yang akan digunakan. Kemudian dilakukan pengecekan terhadap jam setiap *device*, jika jam yang di*input*kan 24 artinya *device* tersebut harus menyala selama 24 jam, maka *device* tersebut akan mendapatkan gen berisi 24. Namun jika jam yang di*input*kan auto maka *device* tersebut akan mendapatkan gen bernilai acak sesuai prioritasnya. Berikut adalah gambaran untuk populasi awal :

```
Population size = (10, 10)
[[21 17 19 17 16 10 12 10  7  5]
 [23 22 20 18 12 13 10  9  8  4]
 [22 16 15 16 12 10 10  9  7  4]
 [20 16 15 17 10 12  8  9  2  1]
 [21 17 16 16 13 13  6  8  4  3]
 [20 17 14 17 12 10  6  5  2  3]
 [24 17 15 18 10 12  9  7  3  3]
 [23 17 17 18 16  9 10  6  4  6]
 [19 19 20 18 16  8  8  7  6  4]
 [18 21 19 16 15  9  8  9  7  5]]
```

Gambar 4 Populasi awal

Setelah itu dilakukan perhitungan untuk *threshold* dari jumlah tagihan yang telah diubah menjadi kWh dan hari yang telah di*input*kan, jumlah tagihan ini kemudian dibagi dengan hari agar mendapatkan kWh untuk setiap harinya yang akan dijadikan sebagai *threshold* sebagai berikut :

$$Th = Pc \div H$$

Th = 681.53320 / 7

Th = 97.3618857142857 kWh

Dibulatkan menjadi:

Th = 97 kWh

Setelah mendapatkan populasi awal, setiap kromosom akan dilakukan perhitungan nilai *fitness* sebagai berikut:

Fitness = $\sum_{i=1}^{n} GiVi$

Fitness = 21 χ 1000 + 17 χ 900 + 19 χ 800 + 17 χ 700 + 16 χ 600 + + 10 χ 500 + 12 χ 400 + 10 χ 300 + 7 χ 200 + 5 χ 100

Fitness = 87700

Perhitungan diatas untuk kromosom pertama, kemudian proses ini akan dilakukan terhadap setiap kromosom.

Pada tahap seleksi untuk mendapatkan *parents*, pemelihan seleksi ini dilakukan untuk mendapatkan *parents* yang terbaik secara random. Langkah yang dilakukan adalah membagi 2 jumlah populasi, setelah membagi 2 jumlah populasi sistem aka secara otomatis memilih kromosom dengan nilai *fitness* yang paling besar, dan kemudian akan kromosom-kromosm tersebut akan dipilih untuk menjadi *parents.*

```
Parents:
[[23. 22. 20. 18. 12. 13. 10.  9.  8.  4.]
 [21. 17. 19. 17. 16. 10. 12. 10.  7.  5.]
 [23. 17. 17. 18. 16.  9. 10.  6.  4.  6.]
 [19. 19. 20. 18. 16.  8.  8.  7.  6.  4.]
 [18. 21. 19. 16. 15.  9.  8.  9.  7.  5.]]
```

Gambar 5 *Parents* hasil seleksi

Setelah mendapatkan *parents*, dilakukan proses *crossover* yang merupakan proses persilangan anatra setiap parents yang akan menghasilakn *offsprings* atau kromsom baru.

Parents A	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
	23	22	20	18	12	13	10	9	8	4
Parents B	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
	21	17	19	17	16	10	12	10	7	5
Hasil										
Parents A	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
	23	21	24	17	13	14	9	10	7	5
Parents B	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
	22	20	24	13	16	11	12	9	8	4

Gambar 6 Ilustrasi *crossover*

Proses *crossover* tidak hanya dilakukan satu kali saja, namun dilakukan secara terus menerus hingga nilai acak yang dibangkitan lebih kecil dari probabilitas *crossover*, dan probabilitas *crossover* yang digunakan adalah 0.4. Setelah proses *crossover* selesai, akan dilakukan pengecekan terhadap setiap kromosom untuk memastikan berat kromosom tidak melebihi threshold. Jika kromosom melebihi threshold, maka nilai gen kromosom tersebut akan di nol kan mulai dari gen yang paling akhir dari setiap kromosom, hingga beratnya sama atau lebih ringan dari threshold.

```
Crossover:
[[19. 17. 16. 17. 14. 11.  6.  8.  8.  4.]
 [18. 16. 17. 16. 11. 12. 10.  8.  3.  3.]
 [18. 21. 15. 12. 11.  8.  7.  4.  4.  5.]
 [21. 22. 19. 12. 12. 13. 11.  6.  3.  5.]
 [21. 17. 20. 16. 15.  9.  8.  4.  4.  5.]]
```

Gambar 7 Kromosom hasil *crossover*

Tahapan terakhir adalah mutasi, proses mutasi ini dilakukan secara otomatis oleh sistem dengan mengoptimalkan nilai pada gen yang ada, seperti utnuk mengganti gen bernilai 0 dengan gen bernilai random dari jarak 1 sampai dengan 24. Sama seperti *crossover*, proses ini tidak hanya dilakukan satu kali saja, namun dilakukan secara terus menerus hingga nilai acak yang dibangkitan lebih kecil dari probabilitas mutasi, dan probabilitas mutasi yang digunakan adalah 0.4. Setelah proses mutasi selesai, akan dilakukan pengecekan terhadap setiap kromosom untuk memastikan berat kromosom tidak melebihi threshold. Jika kromosom melebihi threshold, maka nilai gen kromosom tersebut akan di nol kan mulai dari gen yang paling akhir dari setiap kromosom, hingga beratnya sama atau lebih ringan dari threshold.

```
Mutation:
[[19. 17. 16. 17. 14. 11.  5.  8.  8.  4.]
 [18. 16. 17. 16. 11. 12. 10.  8.  3.  3.]
 [18. 21. 15. 12. 11.  8.  7.  4.  4.  5.]
 [21. 22. 19. 12. 12. 13. 11.  6.  3.  5.]
 [21. 17. 20. 16. 15.  9.  8.  4.  4.  5.]]
```

Gambar 8 Kromosom hasil mutasi

Untuk menggantikan setengah kromosom yang hilang kromosom hasil mutasi ini akan digabungkan kembali dengan kromosom hasil seleksi (*parents*).

```
Last generation:
[[23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 23 22  4]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 21 21 17 22 22  7]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 22 22  7]
 [23 24 20 18 22 21 17 23 22  4]]
```

Gambar 9 Populasi akhir

Proses dari tahap pertama hingga tahap terakhir akan terus dilakukan hingga jumlah generasi terpenuhi, dalam sistem ini jumlah generasi yang harus dipenuhi adalah 50. Kromosom dengan nilai *fitness* dan beratnya tidak melebihi *threshold* maka akan dipilih sebagai solusi.

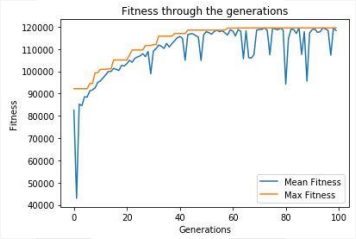
```
Last Height:
95.42999999999999

Fitness of the last generation:
[115400 115400 115400 115400 115400 115400 114800 115400 115400 115400]

The optimized parameters for the given inputs are:
[array([23, 24, 20, 18, 22, 21, 17, 22, 22,  7])]
Device :  23
Device :  24
Device :  20
Device :  18
Device :  22
Device :  21
Device :  17
Device :  22
Device :  22
Device :  7
```

Gambar 10 Kromosom terpilih menjadi solusi

Pada gambar 10 diatas, menunjukkan hasil algoritma genetika yang berupa kromosom terbaik dari generasi terakhir, kemudian isi dari kromosom tersebut dimasukkan satu per satu secara berurutan kedalam database, misalkan gen pertama dalam kromosom merupakan jam untuk device 1, dan seterusnya.

4.2 Pengujian Nilai Fitness Optimal

Untuk mengetahui nilai *fitness* pada kromosom yang optimal terdapat di generasi ke berapa, maka dilakukan pengujian nilai *fitness* optimal. Pengujian dilakukan dengan menaikan batas generasi yang asalnya dari 50 dengan interval 10 kenaikan hingga 100 jumlah generasi. Berikut adalah hasil dari pengujian yang dilakukan:



Gambar 11 Grafik nilai *fitness* optimal

Berdasarkan grafik diatas, nilai *fitness* optimal didapatkan di generasi 60, hal ini dapat dipastikan karena tidak ada kenaikan lagi nilai *fitness* hingga generasi 100.

4.3 Pengujian Rules

Untuk membuktikan *rules* yang digunakan pada inisialisasi populasi berjalan dengan seharusnya, maka dilakukan pengujian *rules*. Berikut adalah data setelah dilakukan pengujian:

Tabel 2 Tabel pengujian *rules*

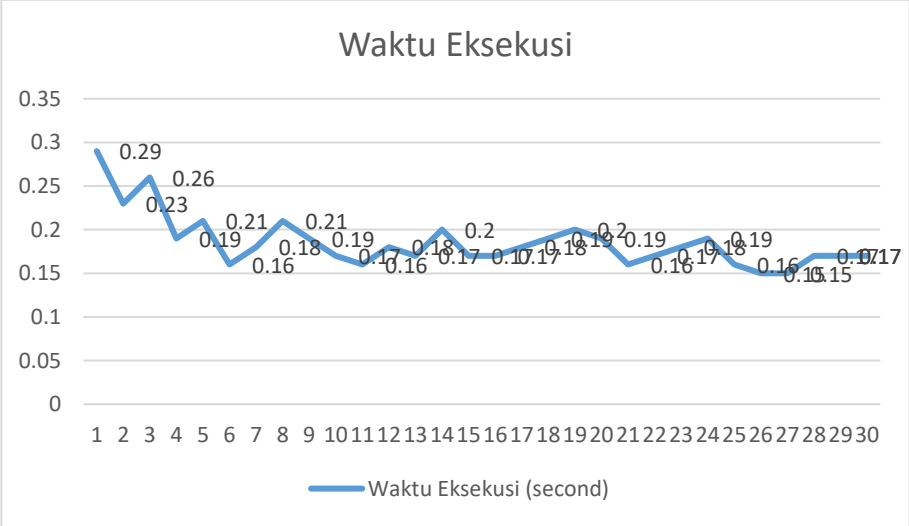
Pengujian	Rule	Variabel		Gen
		Device	Jam	
1	R1	Prioritas 1	24	24
2	R2	Prioritas 2	24	24
3	R3	Prioritas 3	24	24
4	R4	Prioritas 4	24	24
5	R5	Prioritas 5	24	24

Pengujian	Rule	Variabel		Gen
		Device	Jam	
6	R6	Prioritas 6	24	24
7	R7	Prioritas 7	24	24
8	R8	Prioritas 8	24	24
9	R9	Prioritas 9	24	24
10	R10	Prioritas 10	24	24
11	R11	Prioritas 1	Auto	24
12	R12	Prioritas 2	Auto	20
13	R13	Prioritas 3	Auto	18
14	R14	Prioritas 4	Auto	17
15	R15	Prioritas 5	Auto	12
16	R16	Prioritas 6	Auto	10
17	R17	Prioritas 7	Auto	9
18	R18	Prioritas 8	Auto	5
19	R19	Prioritas 9	Auto	3
20	R20	Prioritas 10	Auto	4

Berdasarkan tabel pengujian validasi *rules* diatas, *rules* yang berhasil semua, hal ini menunjukkan bahwa pengujian validasi *rules* ini berhasil 100% dan menunjukan juga bahwa algoritma nantinya akan berjalan dengan semestinya.

4.4 Pengujian Waktu Eksekusi Metode

Pengujian waktu eksekusi dilakukan untuk mengetahui kecepatan sistem dalam merespon penggunanya. Dalam penelitian Tugas Akhir ini, dilakukan pengujian respon mengirim dan menerima data antara *database* dan *website power management*, dilakukan sebanyak 30 kali untuk mendapatkan nilai waktu respon yang lebih akurat.Pengujian dilakukan dengan mencatat waktu eksekusi yang ditampilkan dari hasil keluaran Visual Studio Code. Berikut ini adalah hasil dari pengujiannya:



Gambar 12 Waktu eksekusi metode

Dari hasil pengujian waktu eksekusi diatas, dapat dilihat waktu yang dibutuhkan oleh sistem untuk melakukan seluruh prosesnya yaitu diantara 0,15 detik hingga 0,3 detik, dengan rata-rata waktu eksekusi 0,18 detik.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil dari pengujian dan analisis yang telah dilakukan pada Tugas Akhir ini, maka dapat disimpulkan sebagau berikut:

1. Nilai *fitness* optimal berarda di generasi ke-60 berdasarkan pengujian *fitness* optimal, dan *rules* yang digunakan untuk memberi jarak nilai acak pada gen sudah berjalan dengan baik di pengujian *rules* dengan akurasi 100%.
2. Berdasarkan hasil pengujian waktu proses eksekusi yang dibutuhkan untuk menjalankan sistem yang dimulai dari menerima data dari *database* hingga mengirimkan kembali data ke *database* lagi, rata-rata waktu eksekusi yaitu 0,18 detik.

5.2 Saran

Setelah mendapatkan hasil pengujian yang dilakukan pada Tugas Akhir ini, penulis dapat menyarankan untuk penelitian selanjutnya dapat mencoba untuk menggunakan algoritma optimasi lain seperti *Brute Force, Greedy, Dynamic Programming*, ataupun algoritma optimasi yang lainnya.

Reference

[1] S. A. Widhiono, M. A. Murti, C. Setianingsih, F. T. Elektro, U. Telkom, and A. Studio, "PERANCANGAN APLIKASI ANDROID UNTUK MANAJEMEN DAN KENDALI BEBAN ELEKTRONIK BERBASIS ALGORITMA PRIORITY QUEUE DESIGN OF ANDROID APPLICATION FOR MANAGING AND CONTROLLING ELECTRONIC LOADS BASED ON PRIORITY QUEUE," pp. 1–8.

[2] M. F. Nur, M. A. Murti, C. Setianingsih, F. T. Elektro, and U. Telkom, "Design of Home Electrical Appliances Control and Monitoring System Based on Android," *e-Proceeding Eng. Vol.6, No.1 April 2019*, vol. 6, no. 1, pp. 125–134, 2019.

[3] S. Wei and C. P. Soon, "Genetic algorithm-based text clustering technique," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4221 LNCS, pp. 779–782, 2006, doi: 10.1007/11881070_103.

[4] D. Whitley, "A genetic algorithm tutorial," *Stat. Comput.*, vol. 4, no. 2, pp. 65–85, 1994, doi: 10.1007/BF00175354.

[5] Jyotishree, "Knowledge based operation and problems representation in genetic algorithms," *Dep. Comput. Sc. Appl.*, vol. 39, 2015, [Online]. Available: http://hdl.handle.net/10603/32680.