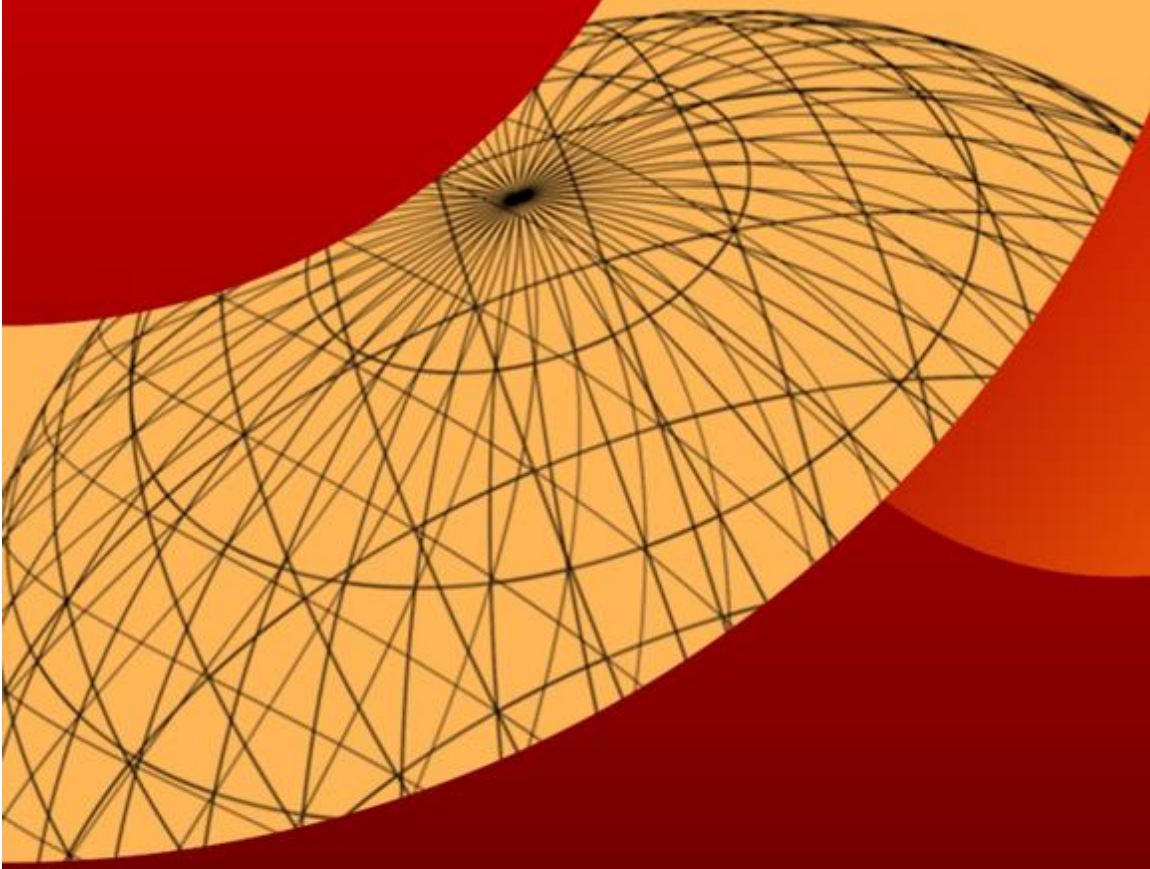
September 2024
Volume 5 Nomor 2
p-ISSN 2723-0325 e-ISSN 2723-0333



TENSOR

Pure and Applied Mathematics Journal

PROGRAM STUDI MATEMATIKA
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PATTIMURA

TENSOR

Pure and Applied Mathematics Journal

is an international academic open-access journal that gains a foothold in mathematics, and its applications are issued twice a year. The focus is to publish original research and review articles on all aspects of pure and applied Mathematics. Editorial board members of the Journal and reviewers will review submitted papers. All submitted articles should report original, previously unpublished research results, experimental or theoretical, and will be peer-reviewed. Articles submitted to the journal should meet these criteria and must not be under consideration for publication elsewhere. Manuscripts should follow the journal template and are subject to both review and editing.

Published by:

Department of Mathematics,
Faculty of Mathematics and Natural Sciences,
Pattimura University.
Ambon
2024

TENSOR
Pure and Applied Mathematics Journal
Volume 5 Number 2 | September 2024

Publisher
Department of Mathematics,
Faculty of Mathematics and Natural Sciences,
Pattimura University, Ambon, Indonesia

Editorial Address
Program Studi Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Unversitas Pattimura
Jln. Ir. M. Putuhena, Kampus Unpatti, Poka - Ambon 97233, Provinsi Maluku, Indonesia
Contact : +62 82397854220
Email : tensormathematics@gmail.com

Volume 5 Number 2 | September 2024

TENSOR
Pure and Applied Mathematics Journal

TENSOR Pure and Applied Mathematics Journal
September 2024 | Vol. 5 | No. 2 pp. 93-104
p-ISSN: 2723-0325 e-ISSN: 2723-0333

Penyelesaian *Unit Commitment Problem* (UCP) Menggunakan Algoritma Genetika

Aisyah Fadhilah Whardhana[1], Asri Bekti Pratiwi[1*], Edi Winarko[1]
[1] Departemen Matematika, Fakultas Sains dan Teknologi, Universitas Airlangga, Surabaya, Indonesia.
[*]Email: asri.bekti@fst.unair.ac.id

Manuscript submitted : September 2024;
Accepted for publication : November 2024.
doi: https://doi.org/10.30598/tensorvol5iss2pp93-104

Abstract: The purpose of this research is to solve the Unit Commitment Problem (UCP), which is a critical task in power system optimization. The UCP involves determining the optimal scheduling of power generating units over a specified time horizon to meet the electricity demand while minimizing costs and satisfying operational constraints. In this study, a Genetic Algorithm (GA) method is proposed to solve the UCP efficiently. GA is inspired by the process of natural selection and evolution and is often used to solve complex optimization problems where traditional methods may be inefficient. The algorithm proceeds through several steps, namely parameters initialization, generating population, modification, calculating fitness function, parent selection, crossover, and mutation. The implementation of GA to solve UCP using C++ includes four different scenarios: a system with 4 units, 5 units, 10 units, and 26 units. The results obtained from the implementation of the GA on the different data sets indicate that the more iterations and the bigger initial population, the smaller the solution in the form of the total cost incurred.

2010 Mathematical Subject Classification: 68W50.
Keywords: Unit Commitment Problem, Genetic Algorithm, Power System.

1. Pendahuluan

Listrik yang digunakan masyarakat merupakan hasil pasokan dari pembangkit listrik. Pembangkit listrik adalah bagian dari alat industri pada sistem tenaga listrik yang berfungsi untuk memproduksi dan membangkitkan tenaga listrik dari berbagai sumber tenaga yang akan didistribusikan kepada masyarakat. Kebutuhan listrik selalu mengalami perubahan sesuai dengan situasi dan kondisi. Oleh karena itu, diperlukan penjadwalan pembangkit listrik yang baik karena suplai pembangkit harus menyesuaikan kebutuhan yang berbeda tiap waktunya. Biaya bahan bakar adalah biaya yang paling besar dalam sistem operasi pembangkitan. Keluaran dari masing-masing unit pembangkit perlu dijadwalkan secara optimal agar didapatkan biaya bahan bakar yang minimum [1]. Suplai listrik pada sistem tenaga listrik diperoleh dari kombinasi beberapa generator yang bekerja dalam periode waktu tertentu. Sistem kerja tenaga listrik dipengaruhi oleh pemeliharaan, biaya pembangkitan, dan kapasitas pembagkit. Oleh karena itu, diperlukan penjadwalan pembangkit listrik agar kinerja sistem tenaga listrik optimal.

Unit Commitment Problem (UCP) adalah permasalahan optimasi untuk penjadwalan unit generator listrik

dengan biaya operasi minimum untuk memenuhi permintaan energi dengan memperhatikan batasan kendalanya [2]. UCP dapat menentukan koordinasi pengoperasian unit-unit pembangkit listrik dalam memasok aliran listrik, yaitu nyala dan matinya unit serta berapa banyak daya yang dapat dihasilkan oleh unit dalam waktu tertentu. Sistem tenaga listrik semestinya tetap bekerja saat terjadi pemadaman pada sistem. Sehingga, dalam sistem tenaga listrik disediakan kapasitas cadangan (*spinning reserve)* dengan menyediakan lebih banyak daya dari yang dibutuhkan [3], yang juga dibahas dalam UCP.

Penyelesaian UCP sebelumnya sudah pernah diselesaikan menggunakan *Ant Colony* [4], *Dynamic Programming* [5], *Particle Swarm Optimization* [6], *Grey Wolf Optimization* [7], *Simulated Annealing* [8]. Dalam penelitian ini akan digunakan Algoritma Genetika (AG) untuk menyelesaikan UCP.

Algoritma Genetika (AG) pertama kali diperkenalkan oleh John Holland pada tahun 1960an. AG merupakan metode pencarian heuristik yang digunakan untuk menemukan solusi optimal. Teknik yang digunakan dalam AG terinspirasi dari proses evolusi dan seleksi alam seperti seleksi, mutasi, pewarisan, dan rekombinasi untuk menemukan solusi [9]. AG sangat baik untuk kumpulan data yang memiliki banyak variabel, memiliki ruang solusi besar, dan mampu menemukan solusi yang baik untuk masalah kompleks, termasuk masalah optimasi dengan kendala maupun tanpa kendala [10].

Sebelumnya, AG telah digunakan untuk menyelesaikan beberapa permasalahan optimasi, diantaranya *Travelling Salesman Problem* [11], *Job-Shop Scheduling Problem* [12], dan *Vehicle Routing Problem* [13]. AG telah dibuktikan memberikan hasil yang lebih baik jika dibandingkan dengan *Simulated Annealing* [14]. AG juga memiliki performa yang lebih baik dari *Ant Colony Optimization* (ACO) [15]. Operator genetika seperti *crossover,* mutasi, dan seleksi dapat membantu dalam memberikan hasil yang optimal. Seleksi yang dimiliki oleh AG dapat memilih fungsi tujuan kromosom yang terbaik. *Crossover* dalam AG dapat memberikan anak dengan kombinasi sifat terbaik dari dua induk. Selain itu, mutasi dapat memberikan keragaman dalam solusi.

Artikel ini terbagi menjadi beberapa bagian. Bagian 1 adalah pendahuluan yang berupa latar belakang, Bagian 2 membahas mengenai UCP. Selanjutnya Bagian 3 menyajikan penjelasan singkat mengenai Algoritma Genetika (AG) dan Langkah-langkah AG untuk menyelesaikan UCP pada Bagian 4. Pada Bagian 5 adalah hasil dan diskusi. Kesimpulan disajikan sebagai penutup yaitu pada Bagian 6.

2. Unit Commitment Problem (UCP)

Menurut [16], *Unit Commitment Problem* (UCP) adalah permasalahan optimasi untuk menjadwalkan unit generator listrik dengan biaya operasi minimum dengan memenuhi permintaan daya dan kendala persamaan dan pertidaksamaannya. UCP dapat menentukan koordinasi pengoperasian unit-unit pembangkit listrik dalam memasok aliran listrik, yaitu nyala dan matinya unit dalam periode waktu tertentu.

Tujuan dari UCP adalah untuk meminimalkan total biaya operasional yang dikeluarkan. Terdapat dua komponen biaya dalam UCP, yaitu biaya pembangkitan dan biaya *start-up.* Biaya pembangkitan, atau bisa juga disebut biaya bahan bakar diekspresikan dengan persamaan kuadrat.

$$F_j(Y_{t,j}) = a_j\left(Y_{t,j}\right)^2 + b_j Y_{t,j} + c_j \tag{1}$$

dengan $Y_{t,j}$ adalah jumlah daya yang diproduksi unit j pada waktu ke-t, dan a_j, b_j, c_j adalah koefisien biaya dari unit ke-j. Biaya *start-up* yang dikeluarkan unit ke-j pada waktu ke-t $(S_{t,j})$ pada waktu tertentu diasumsikan sebagai berikut.

$$S_{t,j} = \begin{cases} SH_j & if \quad T_{min,j}^{off} \le T_j^{off}(t) \le TC_j \\ SC_j & if \quad T_j^{off}(t) > TC_j \end{cases} \tag{2}$$

Biaya *start-up* dipengaruhi oleh waktu mati dari unit generator (T_j^{off}) dihitung dari waktu nyala terakhir. SH_j adalah biaya *start-up* panas yang digunakan ketika waktu mati unit ke-j berada di interval waktu mati minimum unit j $\left(T_{min,j}^{off}\right)$ dan waktu *start-up* dingin $\left(TC_j\right)$. Apabila waktu mati unit generator (T_j^{off}) lebih dari waktu *start-up* dingin (TC_j), maka biaya *start-up* dingin (SC_j) akan digunakan.
Biaya operasi total adalah total dari biaya pembangkitan dan biaya *start-up* diberikan sebagai berikut.

$$minimize \sum_{t=1}^{T}\left(\sum_{j=1}^{N}\left\{F_j\left(Y_{t,j}\right)+S_{t,j}\left(1-U_{t-1,j}\right)U_{t,j}\right\}\right) \tag{3}$$

dengan U adalah matriks berisi elemen biner (0 dan 1) untuk menentukan mati nyalanya unit j pada waktu t dengan kondisi sebagai berikut.

$$U_{t,j}=\begin{cases}1, & jika\ unit\ j\ menyala\ pada\ waktu\ t\\ 0, & sebaliknya\end{cases} \tag{4}$$

Berikut adalah kendala dari UCP.
1. Batasan daya yang harus dipenuhi.

$$\sum_{j=1}^{N} Y_{t,j}U_{t,j} \geq D_t\,, \forall t \in 1,2,3,\dots,T \tag{5}$$

2. Waktu nyala (mati) minimum yang harus dipenuhi sebelum unit dapat dimatikan (dinyalakan).

$$T_j^{on}(t) \geq T_{min,j}^{on}, \forall j \in 1,2,3,\dots,N, \forall t \in 1,2,3,\dots,T \tag{6}$$

$$T_j^{off}(t) \geq T_{min,j}^{off}, \forall j \in 1,2,3,\dots,N, \forall t \in 1,2,3,\dots,T \tag{7}$$

3. Batasan kapasitas cadangan.

$$\sum_{j=1}^{N} Y_{max,j}U_{t,j} \geq Dr_t + D_t \forall t \in 1,2,3,\dots,T \tag{8}$$

dengan keterangan notasi diberikan sebagai berikut.

$Y_{t,j}$	=	daya yang dihasilkan unit j pada waktu t (dalam satuan daya)
$U_{t,j}$	=	status unit j pada waktu t (1 jika unit menyala, 0 sebaliknya)
t	=	indeks periode waktu
j	=	indeks unit
D_t	=	permintaan daya pada waktu t (dalam satuan daya)
Dr_t	=	kapasitas cadangan pada waktu t (dalam satuan daya)
$T_j^{on}(t)$	=	waktu unit j menyala hingga periode waktu t (dalam satuan waktu)
$T_j^{off}(t)$	=	waktu unit j mati hingga periode waktu t (dalam satuan waktu)
$T_{min,j}^{on}$	=	waktu nyala minimum unit j (dalam satuan waktu)
$T_{min,j}^{off}$	=	waktu mati minimum unit j (dalam satuan waktu)
SC_j	=	biaya *start-up* dingin unit j (dalam satuan biaya)
SH_j	=	biaya *start-up* panas unit j (dalam satuan biaya)
TC_j	=	waktu *start-up* dingin unit j (dalam satuan waktu)

3. Algoritma Genetika (AG)

Menurut [17], algoritma genetika (AG) adalah algoritma dengan teknik yang terinspirasi oleh teori evolusi. AG memiliki operator genetik seperti mutasi, seleksi, dan *crossover un*tuk mencari solusi dari masalah optimasi. Proses algoritma genetika dimulai dari populasi awal yang dihasilkan secara acak. Fungsi tujuan dari tiap individu dihitung dan akan dipilih beberapa individu dari populasi untuk membentuk populasi baru. Populasi baru tersebut akan digunakan pada iterasi selanjutnya. Algoritma akan berhenti jika iterasi telah terpenuhi.

Berikut adalah beberapa istilah yang digunakan dalam algoritma genetika menurut [9] dan [18].

1. Populasi adalah kumpulan dari individu dengan ukuran tertentu.
2. Individu adalah kumpulan dari gen yang direpresentasikan dalam bentuk kromosom.
3. Kromosom adalah rangkaian berurutan dari gen.
4. Gen adalah salah satu unsur dalam genotipe yang membentuk kromosom.
5. Genotipe adalah populasi dalam ruang komputasi. Dalam ruang komputasi, solusi direpresentasikan sedemikian rupa agar dapat dipahami dan dimanipulasi dengan sistem komputasi.
6. Fenotipe adalah populasi di ruang dunia nyata di mana solusi direpresentasikan dalam bentuk penerapan dalam situasi dunia nyata.
7. Fungsi tujuan adalah fungsi untuk menentukan nilai evaluasi kromosom.
8. Seleksi adalah operator yang memilih kromosom dalam populasi untuk menghasilkan populasi baru. Semakin bagus fungsi fungsi tujuannya, semakin tinggi kemungkinan untuk dipilih.
9. *Crossover* adalah operator untuk menggabungkan sifat dua kromosom secara acak untuk menghasilkan keturunan.
10. Mutasi adalah operator untuk mengubah sifat dari satu kromosom secara acak untuk meningkatkan keragaman populasi.
11. Populasi baru adalah populasi yang terdiri dari kumpulan individu setelah proses seleksi.

Menurut [19], langkah-langkah dasar dalam GA dapat dijelaskan sebagai berikut:

a. Inisialisasi

   Inisialisasi merupakan langkah pertama untuk menentukan parameter awal dan menciptakan populasi awal. Populasi awal dihasilkan secara acak dan dapat berukuran berapapun.

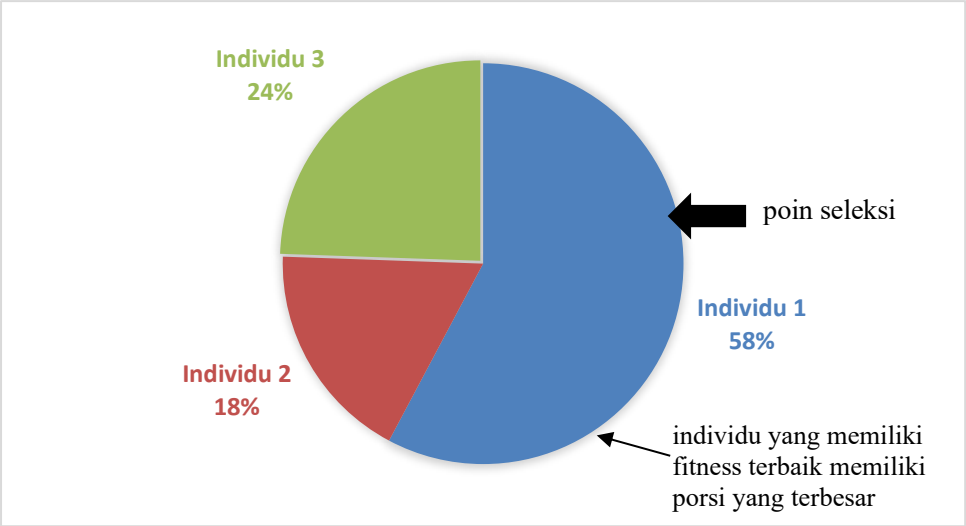
b. Evaluasi

   Setiap anggota populasi dievaluasi sesuai dengan fungsi tujuan.

c. Seleksi induk

   Seleksi induk merupakan proses seleksi untuk menentukan kromosom induk yang akan digunakan dalam proses crossover.

   Seleksi yang digunakan adalah seleksi *roulette wheel*. Pada metode ini, *roulette wheel* dibangun dari hasil fungsi tujuan relatif (rasio dari hasil fungsi tujuan individu dan fungsi tujuan total). Hasil fungsi tujuan relatif kemudian direpresentasikan dalam bentuk *roulette wheel* yang digambarkan sebagai diagram lingkaran. Jumlah fungsi tujuan relatif (S) kemudian dihitung dan dibangkitkan angka random dalam interval 0 hingga S. Setelah itu akan dipilh individu yang sesuai dengan bilangan real random yang dibangkitkan. Individu yang terpilih akan menjadi induk untuk proses *crossover* dan mutasi. Individu dengan nilai fungsi tujuan yang lebih baik akan menempati area yang lebih besar, sehingga kemungkinan untuk terpilih menjadi induk juga akan lebih tinggi [20]. Seleksi *roulette wheel* dapat diilustrasikan dalam Gambar 1.



Gambar 1. Seleksi *Roulette Wheel*

d. *Crossover*

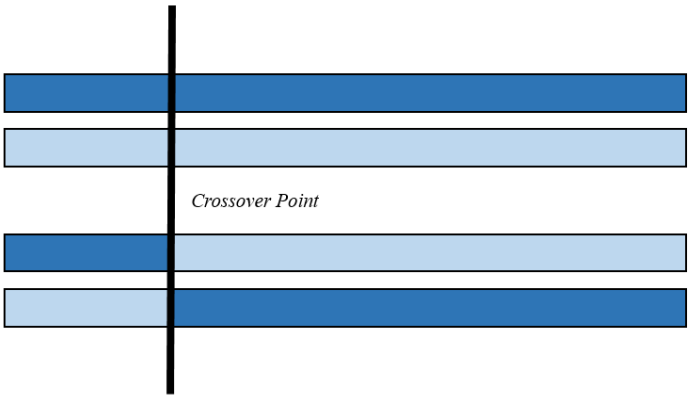
Dalam proses *crossover,* individu baru diciptakan dengan menukar bagian dari gen dua induk tersebut. Tujuannya adalah untuk meghasilkan keturunan dengan menggabungkan sifat kedua induk agar didapatkan keandalan yang lebih baik. Berikut adalah beberapa macam *crossover* dalam AG:

1. *One-Point Crossover*

   Pada *one-point crossover,* titik crossover dipilih secara acak dan gen dalam dua induk tersebut ditukar setelah titik tersebut [21]. Gambar 2 menunjukkan proses *one-point crossover*.

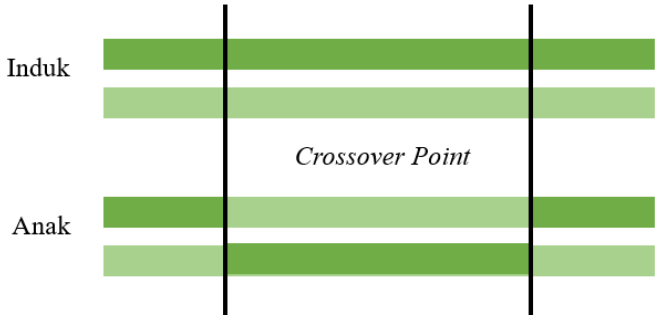
2. *Two-Point Crossover*

   Menurut [22], *two-Point Crossover* bekerja dengan cara memilih dua titik *crossover*



Gambar 2. *One-Point Crossover*

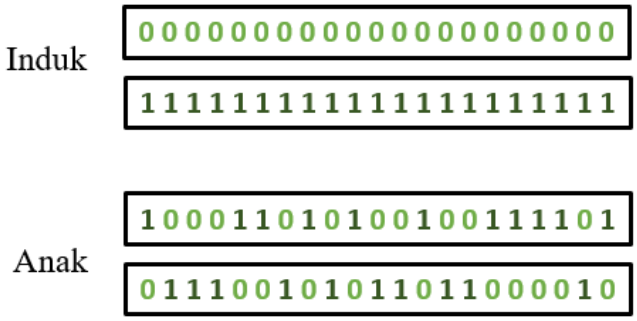
dengan anak pertama mendapatkan bagian pertama dari induk pertama, bagian tengah induk kedua, dan bagian terakhir induk pertama. Anak kedua mewarisi kebalikan dari anak pertama. Ilustrasi *two-point crossover* dapat dilihat pada Gambar 3.



Gambar 3. *Two-Point Crossover*

3. *Uniform Crossover*

Menurut [23], *uniform crossover* dilakukan dengan memilih posisi gen secara acak pada induk dan menukar gen dalam induk tersebut. Ilustrasi *uniform crossover* dapat dilihat pada Gambar 4.



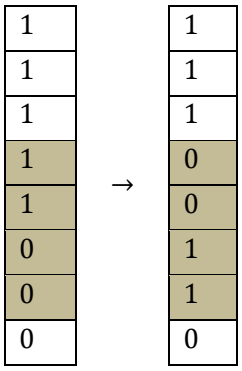
Gambar 4. *Uniform Crossover*

e. Mutasi

Mutasi dilakukan dengan membuat perubahan kecil secara acak pada gen individu. Mutasi dapat meningkatkan keandalan dan menambah diversitas populasi. Berikut adalah beberapa macam mutasi dalam AG:

1. *Inversion Mutation*

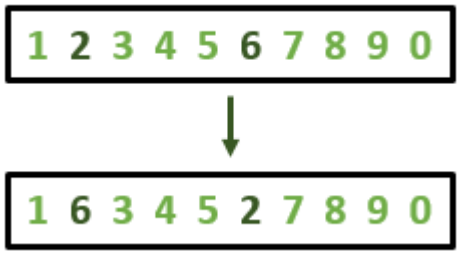
Inversion mutation memilih dua posisi dalam kromosom secara acak dan kemudian menukar gen di antara dua posisi tersebut [24]. Ilustrasi dari *inversion mutation* disajikan pada Gambar 5.



Gambar 5. *Inversion Mutation*

2. *Swap Mutation*

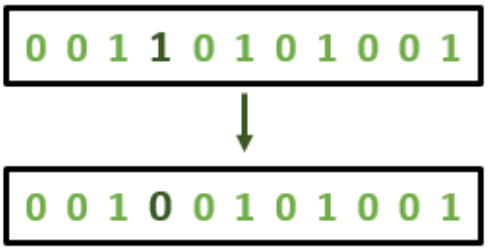
Swap mutation merupakan mutasi ynag dilakukan dengan memilih dua gen secara acak dan menukar posisi gen tersebut [25]. Ilustrasi *swap mutation* dapat dilihat pada Gambar 6.



Gambar 6. *Swap Mutation*

3. *Bit Flip Mutation*

Bit flip mutation bekerja dengan cara memilih satu atau lebih gen secara acak dan dilakukan pembalikan (0 menjadi 1 dan 1 menjadi 0) pada gen tersebut [25]. Ilustrasi *bit flip mutation* dapat dilihat pada **Gambar 2.7.**



Gambar 7. *Bit Flip Mutation*

f. Seleksi populasi baru

Seleksi populasi baru adalah proses untuk memilih individu-individu yang paling baik dengan tujuan meningkatkan peluang adanya individu yang baik untuk dipertahankan di generasi berikutnya.

g. Siklus generasi berulang

Setelah fase mutasi dan *crossover*, akan terbentuk generasi kumpulan individu yang membentuk populasi baru. Setelah itu, dilakukan evaluasi kembali dan memulai kembali proses yang sama untuk generasi berikutnya. Proses ini berlangsung hingga kondisi penghentian tercapai.

h. Kondisi penghentian

Algoritma akan berhenti jika iterasi maksimum telah terpenuhi.

Menurut [26], *gradient descent* adalah teknik optimisasi yang sering digunakan mengoptimalkan nilai minimum dari suatu fungsi. Metode optimisasi ini merupakan metode yang digunakan untuk memperbarui nilai-nilai parameter ke arah gradient negatif pada tiap iterasinya untuk mendapatkan nilai optimal. *Gradient descent* memiliki *learning rate* (γ) yang digunakan untuk menentukan langkah-langkah yang diambil untuk mencapai titik minimum. *Gradient descent* dapat dirumuskan sebagai berikut.

$$s_{t+1} = s_t + \gamma \nabla(-f(s_t)) \quad (9)$$

Pada persamaan (9), s_t mewakili nilai saat ini sedangkan s_{t+1} mewakili nilai pada waktu selanjutnya. Simbol γ adalah ukuran langkah (*learning rate*), dan $f(s_t)$ adalah fungsi biaya. Dalam penelitian ini, *gradient*

descent digunakan untuk menentukan jumlah daya yang dikeluarkan tiap unit per jamnya, dengan s_0 adalah maksimal dari kapasitas daya pada tiap unit.

4. Langkah-Langkah AG untuk Menyelesaikan UCP

Berikut dijelaskan langkah-langkah penerapan Algoritma Genetika (AG) untuk menyelesaikan *Unit Commitment Problem* (UCP).

1. Menginputkan data UCP, yaitu data kebutuhan daya (*demand*), maksimal dan minimal kapasitas daya, biaya bahan bakar, biaya *start-up*, minimal waktu nyala/mati, dan kapasitas cadangan. Melakukan inisialisasi parameter AG, yaitu ukuran populasi awal, probabilitas *crossover* (p_c), probabilitas mutasi (p_m), dan maksimal iterasi.
2. Membangkitkan populasi berupa matriks biner (status nyala/mati) berukuran $jam\ operasi \times jumlah\ unit$ sebanyak ukuran populasi awal, dan matriks daya (maksimal kapsitas unit) yang dihasilkan unit berukuran $jam\ operasi \times jumlah\ unit$.
3. Melakukan modifikasi populasi dengan mengubah elemen matriks biner hingga kendala-kendala minimal waktu nyala/mati terpenuhi. Jika telah memenuhi maka lanjut ke langkah berikutnya dengan mengurangi elemen matriks daya dengan *gradient descent* dengan γ =0.1 dengan memperhatikan *demand* dan minimal kapasitas daya yang dapat dikeluarkan unit.
4. Menghitung fungsi tujuan.
5. Melakukan seleksi induk untuk *crossover* dan mutasi menggunakan seleksi *roulette wheel* (*roulette wheel)*.
6. Crossover dan mutasi.
7. Setelah proses crossover dan mutasi maka didapatkan anak dari hasil *crossover* dan mutasi.
8. Melakukan pengecekan kembali dengan melakukan modifikasi populasi hingga kendala-kendala terpenuhi. Jika telah memenuhi maka lanjut ke langkah berikutnya.
9. Menghitung fungsi tujuan dari hasil *crossover* dan mutasi.
10. Melakukan seleksi individu untuk populasi baru dengan cara menggabungkan hasil evaluasi dari populasi awal, hasil *crossover*, dan mutasi, kemudian menyeleksi populasi sebanyak pop_size dengan hasil fungsi tujuan terbaik untuk membentuk populasi baru.
11. Mengulangi langkah c hingga j sampai maksimum iterasi terpenuhi.
12. Mencetak solusi terbaik.

5. Hasil dan Diskusi

Untuk memudahkan penyelesaian *Unit Commitment Problem* (UCP) menggunakan Algoritma Genetik (AG), langkah-langkah penelitian yang dijelaskan sebelumnya diimplementasikan dalam sebuah program yang ditulis menggunakan bahasa pemrograman C++ dengan bantuan software Borland C++. Impelementasi program untuk menyelesaikan *Unit Commitment Problem* (UCP) menggunakan Algoritma Genetik (AG) menggunakan empat jenis data. Data tersebut terdiri dari data dengan 4 unit, 5 unit, dan 10 unit. Pada implementasi program digunakan probabilitas *crossover* 0.7 dan probabilitas mutasi 0.1 [27].

Data 4 unit diperoleh dari [28] terdiri dari unit dengan jumlah 4 dan jam operasi sebanyak 8 jam. Pada implementasi program dengan data 4 unit generator, digunakan $p_c = 0.7$ dan $p_m = 0.1$, dengan jumlah populasi awal dan maksimal iterasi yang bervariasi.

Tabel 1. Hasil *Running* Data 4 Unit

Biaya Operasi Data 4 Unit, 8 Jam ($) ($p_c = 0.7$ dan $p_m = 0.1$)			
Populasi Awal	Maksimal Iterasi		
	10	50	100
10	88948.22	85414.30	83305.79
50	87273.21	84723.26	83653.31
100	86713.83	84540.35	**83174.04**

Berdasarkan Tabel 1, diperoleh solusi terbaik dengan jumlah populasi awal 100 dan maksimal iterasi 100 dengan total biaya $83,174.04. Pada hasil *running* 4 data, dapat disimpulkan bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan cenderung semakin baik.

Data 10 unit diperoleh dari [29] terdiri dari unit dengan jumlah 5 dan jam operasi sebanyak 24 jam. Pada implementasi program dengan data 5 unit generator, digunakan $p_c = 0.7$ dan $p_m = 0.1$, dengan jumlah populasi awal dan maksimal iterasi yang bervariasi.

Tabel 2. Hasil *Running* Data 5 Unit

Biaya Operasi Data 5 Unit, 24 Jam ($) ($p_c = 0.7$ dan $p_m = 0.1$)			
Populasi Awal	Maksimal Iterasi		
	10	50	100
10	218247.37	175398.07	172044.10
50	218032.66	172085.00	170816.14
100	217448.74	170903.84	**169234.52**

Berdasarkan Tabel 2, diperoleh solusi terbaik dengan jumlah populasi awal 100 dan maksimal iterasi 100 dengan total biaya $169,234.52. Pada hasil *running* 5 data, dapat disimpulkan bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan juga semakin baik.

Data 10 unit terdiri dari [29] unit dengan jumlah 10 dan jam operasi sebanyak 24 jam. Pada implementasi program dengan data 10 unit generator, digunakan $p_c = 0.7$ dan $p_m = 0.1$, dengan jumlah populasi awal dan maksimal iterasi yang bervariasi.

Tabel 3. Hasil *Running* Data 10 Unit

Biaya Operasi Data 10 Unit, 24 Jam ($) ($p_c = 0.7$ dan $p_m = 0.1$)			
Populasi Awal	Maksimal Iterasi		
	10	50	100
10	690655.48	628702.73	615931.11
50	688536.12	625409.79	613788.26
100	687645.45	618493.37	**599382.72**

Berdasarkan Tabel 3, diperoleh solusi terbaik dengan jumlah populasi awal 100 dan maksimal iterasi 100 dengan total biaya $599,382.72. Pada hasil *running* 10 data, dapat disimpulkan bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan juga semakin baik.

Data 26 unit diperoleh dari [30] terdiri dari unit dengan jumlah 26 dan jam operasi selama 24 jam. Pada implementasi program dengan data 10 unit generator, digunakan $p_c = 0.7$ dan $p_m = 0.1$, dengan jumlah populasi awal dan maksimal iterasi yang bervariasi.

Tabel 4. Hasil *Running* Data 26 Unit

Biaya Operasi Data 26 Unit, 24 Jam ($) ($p_c = 0.7$ dan $p_m = 0.1$)			
Populasi Awal	Maksimal Iterasi		
	10	50	100
10	1008000.13	913078.09	905136.88
50	1005014.65	910436.75	908237.01
100	999214.71	904636.42	**900387.01**

Berdasarkan Tabel 4, diperoleh solusi terbaik dengan jumlah populasi awal 100 dan maksimal iterasi 100 dengan total biaya $900,38.01. Pada hasil *running* 26 data, dapat disimpulkan bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan jugq semakin baik. Dari hasil *running* untuk implementasi program menggunakan 4 macam data, dapat dilihat bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan juga semakin baik.

6. Kesimpulan

Algoritma Genetika (AG) dapat digunakan untuk menyelesaikan *Unit Commitment Problem* (UCP) dengan langkah-langkahnya adalah input data, inisialisasi parameter, membangkitkan populasi awal, modifikasi populasi awal, menghitung nilai fungsi tujuan, menyeleksi induk *crossover* dan mutasi dengan metode *roulette wheel,* proses *crossover* dan mutasi, modifikasi anak hasil *crossover* dan mutasi, menghitung nilai fungsi tujuan anak, membentuk populasi baru, dan mengulangi langkah-langkah sebelumnya hingga iterasi maksimal terpenuhi. Berdasarkan hasil running program dengan keempat jenis data, dapat disimpulkan bahwa semakin banyak iterasi maka nilai fungsi tujuan yang dihasilkan semakin baik, dan semakin banyak jumlah populasi maka nilai fungsi tujuan yang dihasilkan juga semakin baik.

Daftar Pustaka

[1] Riyanto, S., Suyono, Dahlan, M. S. (2012). Penjadawalan Pembangkit, Tenaga Listrik Jangka Pendek Menggunakan Ant Colony Optimization. *Jurnal EECCIS,* 6(2), 97-106.

[2] Padhy, N.P. (2004). Unit Commitment—A Bibliographical Survey. *IEEE Transactions on Power Systems*, 19, 1196-1205.

[3] Sujono, Priyadi, A., Pujiantara, M., Anam, S., Yorino, N., Purnomo, M. H. (2021). Optimal Generation Scheduling Considering Distributed Generator for Cost Minimization based on Adaptive Modified Firefly Algorithm. *International Electronics Symposium (IES),* 131-136.

[4] Vaisakh, K., Srinivas, L.R. (2011). Evolving Ant Colony Optimization Based Unit Commitment. *Applied Soft Computing*, 11(2), 2863-2870.

[5] Patra, P. K., Pradhan. P. L. (2014). Dynamic Virtual Programming Optimizing the Risk on Operating System. *Telkomnika Indonesian Journal Of Electrical Engineering*, 12(8), 6369-6379.

[6] Belgin E., T, Zeybekoğlu, Y. (2012). A New Approach for Solving the Unit Commitment Problem by Enhanced Particle Swarm Optimization. *IFAC Proceedings Volumes*, 45(21), 73-78.

[7] Srikanth, B., Ch, R.R, Ch, N.S., Reddy, U.R., Mohit, B., Bdereddin, A.S., Mokhtar, S., Salah, K. (2022). Integrated System Using the Gray Wolf Optimization Technique for Power Quality Improvement. *Frontiers in Energy Research,* 10.

[8] Dudek, G. (2010). Adaptive Simulated Annealing Schedule to the Unit Commitment Problem. *Electric Power Systems Research*, 80, 465-472.

[9] Mitchell, M. (1998) *An Introduction to Genetic Algorithms*. MIT Press. Cambridge. USA.

[10] D'Angelo G., Palmieri,F. (2021). GGA: A Modified Genetic Algorithm with Gradient-Based Local Search for Solving Constrained Optimization Problems. *Information Sciences*, 547, 136-162.

[11] Liu, F., Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TS. *Expert Systems with Applications,* 36(2), 6995-7001.

[12] Liu, M. (2011). An Adaptive Annealing Genetic Algorithm for The Job-Shop Planning and Scheduling Problem. *Expert Systems with Applications,* 38(8), 9248-9255.

[13] Tasan, S, Gen, M. (2012). A Genetic Algorithm Based Approach to Vehicle Routing Problem with Simultaneous Pick-Up and Deliveries. *Computers & Industrial Engineering*, 62(3), 755-761.

[14] Nair, T.R., Sooda, K. (2010). Comparison of Genetic Algorithm and Simulated Annealing Technique for Optimal Path Selection In Network Routing. *KCG College of Technology*, 9, 36-41.

[15] Ashari, I. A., Muslim, M. A., Alamsyah. (2016). Comparison Performance of Genetic Algorithm and Ant Colony Optimization in Course Scheduling Optimizing. *Scientific Journal of Informatics,* 3(2), 51-60.

[16] Yuan, X., Ji, B., Zhang, S., Tian, H., Hou, Y. (2014). A New Approach for Unit Commitment Problem Via Binary Gravitational Search Algorithm. *Applied Soft Computing*, 22, 249–260.

[17] Chu, P., Beasley, J. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4, 63–86.

[18] Rutkovskaya, D. Pilinsky, M. Rutkovsky, L. (2006). Neural Networks, Genetic Algorithms and Fuzzy Systems. Goryachaya liniyaTelekom. Moscow.

[19] Camara, D. (2015). Evolution and Evolutionary Algorithms. *Bio-inspired Networking*, 1-30.

[20] Behera, N. (2020). Analysis of Microarray Gene Expression Data Using Information Theory and Stochastic Algorithm. *Handbook of Statistics,* 43(8), 349-378.

[21] Soon, G. K., Guan, T. T., On, C. K., Alfred, R., Anthony, P. (2013). A Comparison on the Performance of Crossover Techniques in Video Game. *IEEE International Conference on Control System, Computing and Engineering*, 493-498.

[22] Kobak, V.G., Porksheyan, V.M., Jukovskiy, A.G., Shkabriy, R.S. (2021). *Journal of Physics: Conference Series,* 2131.

[23] Umbarkar, A.J., Sheth, P.D. (2015). Crossover Operators in Genetic Algorithms: A Review. *ICTACT Journal on Soft Computing*, 6(1), 1083-1084.

[24] Deep, K., dan Mebrahtu, H. (2011). Combined Mutation Operators of Genetic Algorithm for the Travelling Salesman problem. *International Journal of Combinatorial Optimization Problems and Informatics*, 2(3), 1-23.

[25] Soni, N., Kumar, T. (2014). *International Journal of Computer Science and Information Technologies*, 5(3), 4519-4521.

[26] Andrearczyk, V., Whelan P. F. (2017). Deep Learning in Texture Analysis and Its Application to Tissue Image Classification. *Biomedical Texture Analysis*, 4, 104-123.

[27] Yang, Z., Zhang, W., Liu, H. (2018). Artificial Intelligence Techniques on Real-time Strategy Games. CSAI

'18: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, 11-21.

[28] Kazarlis, S. A., Bakirtzis, A. G., Petridis, V. (1996). A Genetic Algorithm Solution to the Unit Commitment Problem. *IEEE Transactions on Power Apparatus and Systems,* 11(1), 83-92.

[29] Singhal, P. K., Sharma, N. V., Kumar, N. G. (2014). Solution of Unit Commitment Problem Using Enhanced Genetic Algorithm. *2014 Eighteenth National Power Systems Conference (NPSC).*

[30] Roque, L.A.C., Fontes, D. B.B. M., Fontes. F. A. C .C. (2011). A Biased Random Key Genetic Algorithm Approach for Unit Commitment Problem. *Experimental Algorithms in Computer Science*, 6630, 327-339.