

Tugas Besar I IF3070 Dasar Inteligensi Artifisial

Pencarian Solusi Diagonal Magic Cube dengan Algoritma Local Search



Disusun oleh:

K02 - T46

Louis Ferdyo Gunawan / 18222022

Erwan Poltak Halomoan / 18222028

Hartanto Luwis / 18222064

Ammar Naufal / 18222066

**Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganeshya 10, Bandung
40132**

Daftar Isi

Daftar Isi	1
I. Deskripsi Persoalan	2
II. Pembahasan	2
II.1 Pemilihan Fungsi Objektif	2
II.2 Daftar Fungsi yang diimplementasikan pada Tugas Besar I dan Penjelasannya	4
II.3 Penjelasan Algoritma Local Search	29
1. Steepest Ascent Hill-climbing:	29
2. Hill-climbing with Sideways Move	30
3. Random Restart Hill-climbing:	30
4. Stochastic Hill-climbing:	31
5. Simulated Annealing:	32
6. Genetic Algorithm:	33
III. Hasil Eksperimen dan Analisis	35
III.1 Hasil Eksperimen	35
1. Steepest Ascent Hill-climbing	35
2. Hill-climbing with Sideways Move	40
3. Random Restart Hill-climbing	45
4. Stochastic Hill-climbing	51
5. Simulated Annealing	56
6. Genetic Algorithm	62
III.2 Analisis	93
IV. Kesimpulan dan Saran	99
IV.1 Kesimpulan	99
IV.2 Saran	99
V. Pembagian Tugas	101
VI. Referensi	102

Link GitHub: <https://github.com/louisferdyo/Tubes-DAI-K-46.git>

I. Deskripsi Persoalan

Persoalan yang dihadapi dalam tugas kali ini adalah menyelesaikan permasalahan sebuah *diagonal magic cube* berukuran $5 \times 5 \times 5$. *Diagonal magic cube* merupakan kubus yang tersusun dari angka 1 hingga n^3 tanpa pengulangan dengan n adalah panjang sisi pada kubus tersebut (dalam kasus ini, $n = 5$, $n^3 = 125$) yang dibuat sedemikian rupa sehingga jumlah angka untuk setiap baris, kolom, tiang, diagonal potongan bidang, dan diagonal ruang kubus tersebut sama dengan suatu nilai yang disebut *magic number*. Disamping itu, *magic number* tidak harus termasuk dalam angka - angka yang digunakan sebagai susunan kubus tersebut.

Pencarian konfigurasi angka yang tepat untuk memenuhi seluruh syarat *diagonal magic cube* dilakukan dengan menggunakan algoritma *local search*. Setiap iterasi dalam algoritma *local search* akan melakukan penukaran antara 2 angka yang saling bertetangga dalam kubus sampai mendekati hasil konfigurasi yang paling optimum atau bahkan sempurna, yaitu memenuhi seluruh syarat dari suatu *diagonal magic cube*.

II. Pembahasan

II.1 Pemilihan Fungsi Objektif

Definisi Fungsi Objektif

Fungsi objektif adalah fungsi yang digunakan untuk mengecek kesesuaian kondisi saat ini dengan *goals* yang ingin dicapai. Pada kasus pencarian solusi ‘Diagonal Magic Cube’, kami menggunakan fungsi objektif yang akan *me-return* total nilai setiap elemen dari hasil penjumlahan pada baris, kolom, tiang, diagonal bidang, dan diagonal ruang kubus. Fungsi objektif tersebut akan membantu menentukan keberlanjutan algoritma *search* yang digunakan.

Elemen Utama Fungsi Objektif

Di dalam fungsi objektif terdapat elemen utama yang menjadi penentu dalam mendapatkan hasil dari fungsi objektif. Elemen utama tersebut adalah *magic number*. *Magic number* didapatkan menggunakan rumus sebagai berikut.

$$M(n) = \frac{1}{2} n (n^3 + 1).$$

Gambar 1. Formula Mencari Nilai *Magic Number*

Pada kasus *diagonal magic cube* 5x5x5, dengan n sebagai panjang sisi maka didapat bahwa *magic number* sebesar 315. Setiap total jumlah baris, kolom, tiang, diagonal bidang, dan diagonal ruang pada kubus harus sama dengan 315 (*magic number*) sehingga sesuai dengan syarat dari *diagonal magic cube*.

Setelah itu, fungsi objektif dapat memanfaatkan elemen utama (*magic number*) untuk menjadi parameter keberlanjutan proses algoritma *search* yang dilakukan. Fungsi objektif yang digunakan berupa jumlah dari selisih absolut antara *magic number* dalam satu kubus dengan hasil penjumlahan total setiap elemen pada *state* saat ini.

$$\text{total_error} = \sum_{\text{semua elemen}} |\text{magic_number} - \text{jumlah_elemen}|$$

Gambar 2. Formula Total Error atau Deviasi Fungsi Objektif

*elemen yang dimaksud terdiri dari baris, kolom, pilar (tiang), diagonal ruang, dan diagonal masing - masing bidang.

Pseudocode

```

function OBJECTIVE_FUNCTION(cube) returns total_error
    n <- LENGTH(cube)
    // Rumus magic number
    magic_number <- (n * (n^3 + 1)) / 2

    // Inisialisasi total_error
    total_error <- 0

    // Hitung error untuk semua baris, kolom, dan pilar
    for i from 1 to n do
        for j from 1 to n do
            total_error <- total_error + ABS(magic_number - cube.row_sum(i, j)) // Baris
            total_error <- total_error + ABS(magic_number - cube.column_sum(i, j)) // Kolom
            total_error <- total_error + ABS(magic_number - cube.pillar_sum(i, j)) // Pilar/Tiang

    // Hitung error untuk diagonal bidang pada setiap layer (XY, XZ, dan YZ)
    for i from 1 to n do
        total_error <- total_error + ABS(magic_number - cube.xy_diagonal_1(i)) // Diagonal utama XY
        total_error <- total_error + ABS(magic_number - cube.xy_diagonal_2(i)) // Diagonal sekunder XY
        total_error <- total_error + ABS(magic_number - cube.xz_diagonal_1(i)) // Diagonal utama XZ
        total_error <- total_error + ABS(magic_number - cube.xz_diagonal_2(i)) // Diagonal sekunder XZ
        total_error <- total_error + ABS(magic_number - cube.yz_diagonal_1(i)) // Diagonal utama YZ
        total_error <- total_error + ABS(magic_number - cube.yz_diagonal_2(i)) // Diagonal sekunder YZ

```

```

// Hitung error untuk diagonal ruang
total_error <- total_error + ABS(magic_number - cube.space_diagonal1()) // Diagonal ruang 1
total_error <- total_error + ABS(magic_number - cube.space_diagonal2()) // Diagonal ruang 2
total_error <- total_error + ABS(magic_number - cube.space_diagonal3()) // Diagonal ruang 3
total_error <- total_error + ABS(magic_number - cube.space_diagonal4()) // Diagonal ruang 4

return total_error

```

Penjelasan

Fungsi tersebut dirancang untuk menghitung total deviasi (error) dari konfigurasi kubus yang ada terhadap *magic number*. **Total error** ini adalah **Jumlah selisih absolut** antara ***magic number*** dan **penjumlahan elemen - elemen dalam kubus** (baris, kolom, pilar, diagonal bidang, dan diagonal ruang).

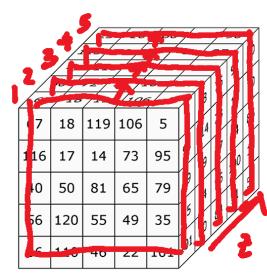
Fungsi objektif ini juga bertujuan untuk meminimalkan error yang terjadi hingga mencapai nol. Semakin kecil total error, maka semakin baik konfigurasi kubus menuju solusi yang seharusnya, *magic cube*. Jika pada akhirnya total error sama dengan nol, maka konfigurasi kubus tersebut adalah *magic cube* yang sempurna dan memenuhi seluruh syarat yang ada. Sebaliknya, jika konfigurasi tidak valid, maka total error akan menjadi penalti yang menunjukkan seberapa jauh konfigurasi angka-angka penyusun kubus saat ini dari solusi atau *magic cube* yang benar.

II.2 Daftar Fungsi yang diimplementasikan pada Tugas Besar I dan Penjelasannya

Berikut adalah daftar fungsi yang digunakan pada implementasi kode untuk setiap algoritma :

No	Nama Fungsi dan Kode	Algoritma Pengguna	Penjelasan
1.	makeCube() Kode : <pre> def makeCube(): cube = [[0 for i in range(5)] for j in range(5)] for k in range(5)] num = 1 for i in range(5): for j in range(5): for k in range(5): cube[i][j][k] = num num += 1 </pre>	Semua	<i>loop</i> pada i,j,k yang pada elemen cube[i][j][k] diisi suatu nilai. Fungsi ini digunakan untuk membuat <i>magic cube</i> yang berisi nilai berulutan dari 1-125.

	<pre> for j in range(5): for k in range(5): cube[i][j][k] = num num += 1 return cube </pre>		
2.	<p>printCube(cube)</p> <p>Kode :</p> <pre> def printCube(cube) : print("Sisi ke sumbu x") for i in range(5) : print() print() print("---- Sisi ke-", i+1, "----") for j in range (5) : print() for k in range (5) : print(cube[i][j][k], " " , end="") print() print() print("-----") print("Sisi ke sumbu y") for j in range(5) : print() print() print("---- Sisi ke-", j+1, "----") for k in range (5) : print() for i in range (5) : print(cube[i][j][k], " " , end="") </pre>	Semua	<p>Fungsi yang memiliki parameter <i>state</i> dari kubus yang digunakan untuk membuat visualisasi kubus dalam dua dimensi yang menampilkan setiap nilai pada sub elemen kubus pada setiap layar (terdiri dari 15 layar) yang dilihat dari sumbu x, sumbu y, sumbu z.</p> <p>Sumbu X</p> <p>Sumbu Y</p> <p>Sumbu Z</p>



```

        print()
        print()

print("-----")
print("Sisi ke sumbu z")
for k in range(5) :
    print()
    print()
    print("---- Sisi ke-",
k+1,"----")
    for i in range (5) :
        print()
        for j in range (5) :
            print(cube[i][j][k],
" | " , end="")
    print()

return 0

```

3.	objectiveFunction(cube) Kode : <pre> def objectiveFunction(cube) : #Menghitung jumlah selisih jumlah setiap row tiang diagonal dengan 315 errorSum = 0 magicNumber = 315 xSum = 0 ySum = 0 zSum = 0 diagonalXSum = 0 contDiagonalXSum = 0 diagonalYSum = 0 contDiagonalYSum = 0 diagonalZSum = 0 </pre>	Semua	<p>Fungsi yang memiliki parameter <i>state</i> dari kubus yang digunakan untuk menghitung nilai objektif dari suatu <i>state</i> pada kubus. Outputnya berupa jumlah <i>error</i> yang merupakan total selisih setiap penjumlahan baris dengan <i>magic number</i>.</p> <p>Pencarian nilai objektif dilakukan dengan melakukan penjumlahan baris ke sumbu x, sumbu y, sumbu z, yang masing-masing nilainya akan dikurangi dengan <i>magic number</i> dan nilainya akan ditambah ke jumlah <i>error</i>. Selanjutnya perhitungan</p>

```

contDiagonalZSum = 0
spaceDiaSum_a = 0
spaceDiaSum_b = 0
spaceDiaSum_c = 0
spaceDiaSum_d = 0
# menghitung ke sumbu x (baris)
for j in range(5) :
    for k in range (5) :
        xSum = 0
        for i in range (5) :
            xSum += cube[i][j][k]
        errorSum += abs(xSum -
magicNumber)

# menghitung ke sumbu y (kolom)
for k in range(5) :
    for i in range (5) :
        ySum = 0
        for j in range (5) :
            ySum += cube[i][j][k]
        errorSum += abs(ySum -
magicNumber)

# Menghitung ke sumbu z
for i in range(5) :
    for j in range(5) :
        zSum = 0
        for k in range(5) :
            zSum += cube[i][j][k]
        errorSum += abs(zSum -
magicNumber)

# print("e1 saat ini : ",
errorSum)
# menghitung diagonal sisi ke
sumbu x
for i in range(5) :
    diagonalXSum = 0
    contDiagonalXSum = 0
    for x in range(5) :
        diagonalXSum +=

```

diagonal dilakukan sisi dengan melakukan *loop* pada masing-masing sisi. Terakhir perhitungan diagonal sisi(dimulai dari atas) yang masing-masing jumlah baris diagonalnya dikurangi *magic number* dan ditambahkan ke total *error*. Semakin kecil nilai objektif (nilai *error*) maka dapat dikatakan *state* tersebut semakin mendekati global optimum (Error = 0).

```
cube[i][x][x]

        contDiagonalXSum +=
cube[i][4-x][x]
        errorSum +=
abs(contDiagonalXSum - magicNumber)
        errorSum += abs(diagonalXSum -
magicNumber)
        # menghitung diagonal sisi ke
sumbu y
        for j in range(5) :
            diagonalYSum = 0
            contDiagonalYSum = 0
            for y in range(5) :
                diagonalYSum +=
cube[y][j][y]

            contDiagonalYSum +=
cube[y][j][4-y]
            errorSum +=
abs(contDiagonalYSum - magicNumber)
            errorSum += abs(diagonalYSum -
magicNumber)
            # menghitung diagonal sisi ke
sumbu z
            for k in range(5) :
                diagonalZSum = 0
                contDiagonalZSum = 0
                for z in range(5) :
                    diagonalZSum +=
cube[z][z][k]

                contDiagonalZSum +=
cube[z][4-z][k]
                errorSum += abs(diagonalZSum -
magicNumber)
                errorSum +=
abs(contDiagonalZSum - magicNumber)
                # menghitung diagonal ruang hitung
```

	<pre> dari atas for a in range (5) : spaceDiaSum_a += cube[a] [4-a] [a] errorSum += abs(spaceDiaSum_a - magicNumber) for b in range (5) : spaceDiaSum_b += cube[4-b] [4-b] [b] errorSum += abs(spaceDiaSum_b - magicNumber) for c in range (5) : spaceDiaSum_c += cube[4-c] [4-c] [4-c] errorSum += abs(spaceDiaSum_c - magicNumber) for d in range (5) : spaceDiaSum_d += cube[d] [4-d] [4-d] errorSum += abs(spaceDiaSum_d - magicNumber) return errorSum </pre>		
4.	<p>initialStateRandom(cube)</p> <p>Kode :</p> <pre> def initialStateRandom(cube) : arrayAngka = [] for i in range(5): for j in range(5): for k in range(5): found = False while found == False : angkaAcak = </pre>	Semua	Fungsi yang memiliki parameter <i>state</i> dari kubus yang digunakan untuk membuat suatu <i>state random</i> pada kubus dan menghasilkan output <i>state</i> kubus yang acak. Pada loop i;j;k, fungsi akan men-generate angka random. Angka random tersebut dilakukan pengecekan terhadap suatu array (berisi nilai-nilai yang sudah ada di

	<pre>random.randint(1,125) if(angkaAcak not in arrayAngka) : arrayAngka.append(angkaAcak) cube[i][j][k] = angkaAcak found = True return 0</pre>		<p><i>magic cube)</i> apakah sudah pernah di <i>assign</i> ke <i>magic cube</i>. Jika belum <i>cube[i][j][k]</i> akan bernilai angka random tersebut dan angka random akan dimasukkan ke array pengecekan.</p>
5.	<p>findBestNeighbor(cube)</p> <p>Kode :</p> <pre>def findBestNeighbor(cube) : bestCube = copy.deepcopy(cube) bestValue = objectiveFunction(cube) for i in range(5): for j in range(5): for k in range(5): for l in range(5): for m in range(5): for n in range(5): if (i, j, k) != (l, m, n): # Ubah posisi cube[i][j][k], cube[l][m][n] = cube[l][m][n], cube[i][j][k] # Cari nilai objektif dari pertukaran tersebut value = objectiveFunction(cube) # jika</pre>	<ul style="list-style-type: none"> - Steepest Ascent - Sideways Move - Random Restart 	<p>Fungi yang memiliki parameter <i>state</i> dari kubus digunakan untuk mencari tetangga yang memiliki nilai objektif terendah dari semua <i>state</i> yang mungkin dan memberikan output berupa <i>best state</i> yang memiliki nilai objektif terendah dari semua kemungkinan. Pencarian dilakukan dengan menggunakan 6 loop, tiga <i>loop</i> sebagai indeks untuk kubus pembanding dan tiga untuk kubus yang dibandingkan.</p>

	<pre> nilai error atau value lebih kecil, lakukan pertukaran if value < bestValue: bestCube = copy.deepcopy(cube) bestValue = value # Ubah kemabali ke cube asal cube[i][j][k], cube[l][m][n] = cube[l][m][n], cube[i][j][k] return bestCube </pre>		
6.	<p>steepestAscent(cube)</p> <pre> def steepestAscent(cube) : startTime = time.time() objectiveValues = [] condition = True iterationSum = 0 while condition == True : currentValue = objectiveFunction(cube) objectiveValues.append(currentValue) neigborCube = findBestNeighbor(cube) neigborValue = objectiveFunction(neigborCube) if(neigborValue < currentValue) : cube = neigborCube iterationSum += 1 else : condition = False endTime = time.time() </pre>	- Steepest Ascent	Fungsi yang menjalankan algoritma Hill Climbing, yaitu Steepest Ascent yang outputnya berupa suatu state yang merupakan lokal optima (bisa global maksimum). <i>Neighbor State</i> didapat dengan memanggil <i>findBestNeighbor</i> yang akan mengembalikan <i>state</i> terbaik (setelah mencoba melakukan pertukaran untuk setiap kemungkinan) yang mungkin berdasarkan <i>state</i> sebelumnya.

	<pre> duration = endTime - startTime plotObjectiveFunction(objectiveValues) return cube, iterationSum, duration </pre>		
7.	<p>sidewaysMove(cube, maxSideways)</p> <p>Kode :</p> <pre> def sidewaysMove(cube, maxSideways) : startTime = time.time() objectiveValues = [] condition = True iterationSum = 0 sidewaysSum = 0 while condition == True : currentValue = objectiveFunction(cube) objectiveValues.append(currentValue) neigborCube = findBestNeighbor(cube) neigborValue = objectiveFunction(neigborCube) if(neigborValue < currentValue) : cube = neigborCube iterationSum += 1 elif(neigborValue == currentValue and sidewaysSum < maxSideways) : cube = neigborCube iterationSum += 1 sidewaysSum += 1 else : condition = False endTime = time.time() duration = endTime - startTime </pre>	<p>- Sideways Move</p> <p>Fungsi yang melakukan algoritma Hill Climbing, yaitu SidewaysMove yang mirip dengan algoritma Steepest Ascent dan disertai parameter tambahan, yakni maxSideways yaitu jumlah maksimal dilakukannya Sideways Move (ketika nilai objektif dari <i>neighbor state = current state</i>). Fungsi ini memungkinkan keluar dari lokal optima ketika ada <i>objective value</i> dari <i>neighbor state = current state</i> dengan mengganti <i>current state</i> menjadi <i>neighbor state</i>. Dengan harapan iterasi selanjutnya bisa menurunkan nilai objektif.</p>	

	<pre> plotObjectiveFunction(objectiveValues) return cube, iterationSum, duration, sidewaysSum </pre>		
8.	<p>randomRestart(cube, maxRestart)</p> <p>Kode :</p> <pre> def randomRestart(cube, maxIteration): : startTime = time.time() i = 0 restartCount = 0 iterationSum = [] objectiveValues = [] condition = True while condition == True : currentValue = objectiveFunction(cube) objectiveValues.append(currentValue) neigborCube = findBestNeighbor(cube) neigborValue = objectiveFunction(neigborCube) if(neigborValue < currentValue) : cube = neigborCube i += 1 elif (restartCount < maxIteration and neigborValue >= currentValue) : cube = initialStateRandom(cube) restartCount += 1 iterationSum.append(i) i = 0 </pre>	<p>- Random Restart</p>	<p>Fungsi yang melakukan algoritma Hill Climbing yaitu Random Restart yang mirip dengan algoritma Steepest Ascent yang disertai parameter tambahan yaitu maxRestart yaitu jumlah maksimal dilakukannya <i>restart</i>. <i>Restart</i> dilakukan ketika program sudah mencapai lokal optima atau tidak ada <i>neighbor state</i> yang kurang dari <i>current state</i> yang dihasilkan dari fungsi findBestNeighbor.</p>

	<pre> else : condition = False endTime = time.time() duration = endTime - startTime plotObjectiveFunction(objectiveValues) return cube, restartCount, iterationSum, duration </pre>		
9.	<p>plotObjectiveFunction(objectivesValue)</p> <p>Kode :</p> <pre> def plotObjectiveFunction(objectiveValues) : plt.figure(figsize=(10, 6)) plt.plot(range(1, len(objectiveValues) + 1), objectiveValues, marker='o', label='Nilai Objective Function') plt.xlabel('Iterasi') plt.ylabel('Nilai Objective Function') plt.title('Plot Nilai Objective Function terhadap Jumlah Iterasi') plt.legend() plt.grid(True) plt.show() </pre>	Semua	Fungi yang menggunakan library math dan berfungi untuk membuat grafik mengenai plot nilai objektif pada setiap iterasi.
10	<p>randomCoorinate()</p> <p>Kode :</p> <pre> def randomCoordinate() : x = random.randint(0, 4) y = random.randint(0, 4) z = random.randint(0, 4) </pre>	<ul style="list-style-type: none"> - Stochastic - Simulated Annealing 	Merupakan fungsi yang melakukan <i>generate</i> koordinat acak yang akan digunakan di fungsi makeRandomState sebagai koordinat acak dari kubus yang akan dilakukan pertukaran. Ouputnya berupa tuple yang berupa nilai x,y,z sebagai koordinat dari kubus.

	<pre>return (x,y,z)</pre>		
11	<p>makeRandomState(cube)</p> <p>Kode :</p> <pre>def makeRandomState(cubeTemp) : condition = False #kondisi untuk cari nilai coor1 = randomCoordinate() coor2 = randomCoordinate() while condition == False : if(coor1 == coor2) : coor1 = randomCoordinate() coor2 = randomCoordinate() else : condition = True break # print("\nstate 1 = \n", cube[coor1[0]][coor1[1]][coor1[2]]) # print("\nstate 2 = \n", cube[coor2[0]][coor2[1]][coor2[2]]) temp = cubeTemp[coor2[0]][coor2[1]][coor2[2]] cubeTemp[coor2[0]][coor2[1]][coor2[2]] = cubeTemp[coor1[0]][coor1[1]][coor1[2]] cubeTemp[coor1[0]][coor1[1]][coor1[2]] = temp return cubeTemp</pre>	<ul style="list-style-type: none"> - Stochastic - Simulated Annealing 	Fungsi yang melakukan pertukaran acak suatu nilai pada kubus dan merupakan <i>neighbor state</i> dari algoritma Stochastic dan Simulated Annealing. Dilakukan dengan memanggil dua kali fungsi randomCoordinate untuk didapat koordinat acak yang akan menjadi indeks pertukaran elemen pada kubus.
12	<p>stochastic(cube, maxIteration)</p> <p>Kode :</p>	<ul style="list-style-type: none"> - Stochastic 	Fungsi yang melakukan algoritma <i>stochastic</i> yang dibatasi oleh maxIteration.

	<pre> def stochastic(cube, maxIteration) : startTime = time.time() i = 0 objectiveValues = [] while i < maxIteration: currentValue = objectiveFunction(cube) objectiveValues.append(currentValue) cubeTemp = copy.deepcopy(cube) neighborCube = makeRandomState(cubeTemp) # print("hasil tengah : ", objectiveFunction(cube)) nextValue = objectiveFunction(neighborCube) if(nextValue < currentValue) : cube = neighborCube i += 1 endTime = time.time() duration = endTime - startTime plotObjectiveFunction(objectiveValues) return cube, duration </pre>		<p>Pada algoritma ini <i>neighbor state</i> didapatkan dengan memanggil fungsi <i>makeRandomState</i> yaitu melakukan perubahan <i>state</i> dengan melakukan pertukaran nilai secara acak. Jika nilai objektif dari pertukaran tersebut lebih kecil maka <i>state</i> tersebut akan menjadi <i>currentState</i>.</p>
13	<p>simulatedAnnealing(cube, Temperature, coolingRate, threshold)</p> <p>Kode :</p> <pre> def simulatedAnnealing(cube, T, coolingRate, threshold) : startTime = time.time() i = 0 stuck = 0 objectiveValues = [] deltaEValues = [] </pre>	<p>- Simulated Annealing</p>	<p>Fungsi yang menjalankan algoritma <i>simulatedAnnealing</i> yang memiliki beberapa parameter. Ketika temperatur belum 0, algoritma akan terus berjalan. Schedule pengurangan temperatur yang kami pilih yaitu pengurangan biasa “T = T - coolingRate” agar program tidak berjalan terlalu lama dan hasilnya lebih mudah dianalisis.</p> <p>Pengurangan temperatur</p>

```

while T > 0:
    T -= coolingRate # penurunan
temperatur dilakukan dengan dikurangi
koefisien penurunan
    i += 1
    if T <= 0 :
        break
    print("\nT : ", T)
    currentValue =
objectiveFunction(cube)

objectiveValues.append(currentValue)
    cubeTemp = copy.deepcopy(cube)

    cube2 =
makeRandomState(cubeTemp)
    # print("hasil tengah : ",
objectiveFunction(cube))
    nextValue =
objectiveFunction(cube2)

    print("current : " ,
currentValue, " next : ", nextValue)
    deltaE = nextValue -
currentValue
    print("delta = ", deltaE)

    if(deltaE < 0) : # karena yang
lebih kecil dari 0 artinya next value
lebih kecil dari current yang artinya
mengurangi error
        cube = cube2
        print("less than 0")
    else :
        prob = math.exp(-deltaE/
T) # karena error harus lebih kecil
maka dikali -1
        deltaEVvalues.append(prob)
        print("prob = ", prob)

```

dilakukan di awal.

Pada Simulated Annealing, ***neighbor state akan dipilih secara acak*** sama seperti pada Stochastic, akan tetapi parameter perpindahan *state* ditentukan oleh Delta E. **Delta E** didapat dengan mengurangi nilai objektif pada *neighbor state* dengan *current state*. Jika **Delta E kurang dari 0** artinya nilai objektif *neighbor state* lebih kecil yang artinya memiliki *error* lebih sedikit sehingga *current state* saat ini akan diganti oleh *neighbor state*. Jika sebaliknya, **perhitungan probabilitas** akan dilakukan dengan formula **prob = -Delta E / T**. Formula menggunakan minus karena *objective value* berbanding terbalik dengan algoritma umumnya yang mana semakin kecil artinya semakin baik. Ketika **probabilitas lebih dari threshold**, perubahan *state* akan dilakukan. Jika tidak frekuensi *stuck* akan bertambah.

	<pre> if(prob > threshold) : cube = cube2 print("swapped") else : stuck += 1 # print("hasil akhir : ", objectiveFunction(cube)) endTime = time.time() duration = endTime - startTime plotObjectiveFunction(objectiveValues, deltaEValues) return cube, stuck, duration </pre>		
14	<p>plotObjectiveFunction(objectiveValues, deltaValues)</p> <p>Kode :</p> <pre> def plotObjectiveFunction(objectiveValues, deltaEValues): plt.figure(figsize=(12, 6)) # Plot untuk nilai objective function terhadap iterasi plt.subplot(2, 1, 1) plt.plot(range(1, len(objectiveValues) + 1), objectiveValues, label='Nilai Objective Function') plt.xlabel('Iterasi') plt.ylabel('Nilai Objective Function') plt.title('Plot Nilai Objective Function terhadap Iterasi') plt.legend() plt.grid(True) </pre>	- Simulated Annealing	Fungsi yang melakukan plot terhadap delta E dan nilai objektif setiap iterasi.

	<pre># Plot untuk exp(ΔE / T) terhadap iterasi plt.subplot(2, 1, 2) plt.plot(range(1, len(deltaEValues) + 1), deltaEValues, label='Nilai exp(ΔE / T)') plt.xlabel('Iterasi') plt.ylabel('Nilai exp(ΔE / T)') plt.title('Plot Nilai exp(ΔE / T) terhadap Iterasi') plt.legend() plt.grid(True) plt.tight_layout() plt.show()</pre>		
15	<p>generateRandomPopulation(populationSize)</p> <p>Kode :</p> <pre>def generateRandomPopulation(populationSize) : population = [] for i in range(populationSize) : individu = list(range(1, 126)) random.shuffle(individu) population.append(individu) return population</pre>	- Genetic Algorithm	Fungsi yang membuat populasi awal yang menginisiasi setiap <i>state</i> individu dalam populasi secara random.
16	<p>geneticAlgorithm(cube,populationSize, maxIteration)</p> <p>Kode :</p> <pre>def geneticAlgorithm(cube, populationSize, maxIteration) : startTime = time.time() lowestObjective = 9999</pre>	- Genetic Algorithm	Fungsi ini menjalankan algoritma <i>geneticAlgorithm</i> yang akan berjalan sampai iterasi maksimal atau mencapai global maksimum. Pertama populasi dibuat dengan memanggil fungsi <code>generateRandomPopulation</code> yang akan menghasilkan individu dengan <i>random</i>

```

indeks = 0
population = []
population =
generateRandomPopulation(populationSize)

maxObjectivePerIteration = []
averageObjectivePerIteration = []
while(indeks < maxIteration and
lowestObjective > 0) :
    fitnessScore = []
    fitnessSum = 0
    fitnessPercentage = []
    wheelRange = []
    newPopulation = []
    currentObjectives = []
    sumPercentage = 0
    for i in range
(populationSize) :
        individu2 =
convertTo3D(population[i])
        currentObjective =
objectiveFunction(individu2)

        currentObjectives.append(currentObject
ive)
        if(currentObjective <
lowestObjective) :
            bestCube =
copy.deepcopy(cube)
            lowestObjective =
currentObjective
            fitness = 1 / (1 +
currentObjective)

            fitnessScore.append(fitness)
            fitnessSum += fitness

maxObjectivePerIteration.append(max(cu
rrentObjectives))

```

state sejumlah ukuran populasi.

Dalam **perhitungan nilai fitness**, dilakukan dengan menghitung nilai objektif dari setiap individu. Akan tetapi karena nilai fitness ini akan digunakan dalam seleksi spinWheel, maka nilai **fitness** akan **berbanding terbalik dengan nilai objektif**. Hal tersebut dikarenakan semakin kecil nilai objektif artinya semakin baik dan semakin mendekati global optimum, sedangkan nilai fitness yang baik adalah yang semakin besar sehingga memiliki peluang yang lebih besar untuk terpilih pada spinWheel. Oleh karena itu "**nilai fitness = $1/(1+\text{nilai objektif})$** " (ditambah satu untuk menghindari zero division). Nilai fitness ini akan dimasukkan ke dalam array sebagai kumpulan nilai fitness dari setiap individu.

Untuk **menentukan persentase spinWheel**, nilai fitness dari masing-masing individu dihitung dengan dibagi total nilai fitness dan dikali 100% yang nantinya nilai tersebut akan dimasukkan ke **array wheelRange** yang akan menjadi batas pada spinWheel. Setelah batas terdefinisi, akan dilakukan **algoritma spinWheelSelection** untuk memilih *parent* sejumlah ukuran populasi. Setelah

```

averageObjectivePerIteration.append(sum(currentObjectives) / populationSize)
    for i in range(len(fitnessScore)) :
        fitnessPercent =
(fitnessScore[i] / fitnessSum) * 100
            sumPercentage +=
fitnessPercent

wheelRange.append(sumPercentage)

fitnessPercentage.append(fitnessPercent)

parent =
spinWheelSelection(population,
wheelRange)
    for i in range(populationSize) :
        child =
crossOver(populationSize, parent)
            if(child != None) :

newPopulation.append(child)
    for i in range(len(newPopulation)) :
        newPopulation[i] =
mutate(newPopulation[i], 0.1)
        indeks += 1
        print(f"Best Objective :
{lowestObjective}")
        population = newPopulation
endTime = time.time()
duration = endTime - startTime

plotObjectiveFunction(maxObjectivePerIteration,
averageObjectivePerIteration)
    return cube, lowestObjective,

```

didapat *parent* sejumlah ukuran populasi, akan dilakukan pemanggilan **fungsi crossover**. Setelah itu dilanjutkan dengan pemanggilan **fungsi mutate**. Setelah mutasi dilakukan(baik berhasil atau tidak) **individu (child) hasil crossover dan mutasi** akan menjadi **populasi baru** yang akan menjadi populasi awal untuk iterasi selanjutnya.

	indeks, duration, bestCube		
	<p>spinWheelSelection(population, wheelRange)</p> <p>Kode :</p> <pre>def spinWheelSelection(population, wheelRange) : parent = [] totalPopulation = len(population) for i in range(totalPopulation) : randomValue = random.uniform(0,100) for j in range(totalPopulation) : if(j == 0) : if(randomValue >=0 and randomValue <= wheelRange[j]) : parent.append(population[j]) break else : if(randomValue >= wheelRange[j-1] and randomValue < wheelRange[j]) : parent.append(population[j]) break return parent</pre>	<p>- Genetic Algorithm</p>	<p>Fungsi ini memuat algoritma untuk melakukan seleksi dengan metode <i>spin wheel</i> di mana akan dilakukan iterasi sampai jumlah populasi untuk ditemukan <i>parent</i> yang nantinya akan dilakukan <i>crossover</i>.</p> <p>Akan dilakukan pemilihan angka acak dari 0-100. <i>wheelRange</i> merupakan array yang berisi batas-batas persentase dari calon <i>parent</i>. Ketika angka random yang terpilih berada dalam suatu <i>range</i> tertentu, individu yang memiliki <i>range</i> tersebut akan dimasukkan ke array <i>parent</i>.</p>
18	<p>crossOver(populationSize, parent)</p> <p>Kode :</p> <pre>def crossOver(populationSize, population) : x = random.randint(0,populationSize-1) y =</pre>	<p>- Genetic Algorithm</p>	<p>Fungsi ini melakukan crossover antara <i>parent</i> yang terpilih. Pemilihan <i>parent</i> untuk dilakukan <i>crossover</i> dilakukan dengan memilih secara acak pada array <i>parent</i> dan memastikan bahwa <i>parent</i> yang akan di <i>crossover</i> berbeda serta <i>parent</i> lebih dari satu. Setelah</p>

	<pre> random.randint(0,populationSize-1) if(populationSize == 1) : print("Populasi hanya satu, tidak bisa crossover") return None else : while(x == y) : y = random.randint(0,populationSize-1) # print(f"y: {y}") length = len(population[x]) crossoverPoint = random.randint(1, length - 1) # Titik crossover acak dimula dari 1 untuk menghindari crossover seluruh array # Buat child dengan melakukan crossover child = population[x] [:crossoverPoint] + population[y] [crossoverPoint:] return child </pre>		<p>itu akan ada pemilihan titik <i>crossover</i> secara acak. <i>Crossover</i> dilakukan dengan mengambil elemen ke 0 sampai elemen pada titik <i>crossover</i> dari <i>parent 1</i>. Serta mengambil elemen dari titik <i>crossover</i> + 1 sampai elemen terakhir. <i>Crossover</i> dilakukan sebanyak jumlah individu pada populasi agar menghasilkan jumlah <i>child</i> yang sama.</p>
19	<p>mutate(individu, mutationRate)</p> <p>Kode :</p> <pre> def mutate(individu, mutationRate) : indexTochange = 0 if(random.random() < mutationRate): print("\nMutasi Dilakukan\n") mutationIndex = random.randint(0, len(individu) - 1) newValue = random.randint(1, 125) </pre>	- Genetic Algorithm	<p>Fungsi ini melakukan perubahan <i>state</i> pada individu dengan melakukan pertukaran secara acak. Mutasi akan dilakukan ketika angka yang <i>digenerate</i> secara acak kurang dari <i>mutationRate</i>.</p>

	<pre> for i in range(125) : if(individu[i] == newValue) : indexTochange = i break temp = individu[mutationIndex] # melakukan pertukaran secara acak pada indeks yang acak individu[mutationIndex] = newValue individu[indexTochange] = temp # print(f"indeks yang berubah yaitu {mutationIndex} dan indeks yang terdampak yaitu {indexTochange}") # print(f"Nilai yang diubah menjadi {newValue} dan nilai yang berubah yaitu {temp}") else : print("\nMutasi Gagal Dilakukan\n") return individu </pre>		
20	<p>convertTo3D(individu)</p> <p>Kode :</p> <pre> def convertTo3D(individu): return np.reshape(individu, (5, 5, 5)) </pre>	- Genetic Algorithm	<p><i>State</i> individu direpresentasikan dalam array biasa. Agar <i>state</i> tersebut bisa menggunakan <i>objective function</i> maka <i>state</i> tersebut harus di ubah ke dalam bentuk $\text{cube}[i][j][k]$. Fungsi ini menggunakan <i>library numpy</i> untuk mengubah array biasa menjadi matrix 3d.</p>
21	<p>showInitiateCube(cube)</p> <p>Kode:</p> <pre> # Untuk yang kubus awal (Initiate) def showInitiateCube(cube): x, y, z, text = [], [], [], [] </pre>	Semua	<p>Fungsi ini memberikan visualisasi tiga dimensi angka - angka yang menyusun Magic Cube dengan menerapkan array, lalu disesuaikan penempatannya tepat</p>

```

for i in range(5):
    for j in range(5):
        for k in range(5):
            x.append(i)
            y.append(j)
            z.append(k)

text.append(str(cube[i][j][k]))
    #Angkanya harus dibaca dari bawah
    dan sudut pandang yang tepat, karena
    dalam 3D tidak bisa persis membuat
    sesuai dengan hasil cetakan terminal
    (2D)

    fig =
go.Figure(data=[go.Scatter3d(
    x=x,
    y=y,
    z=z,
    mode='text',
    text=text,
    textposition="middle center",
    textfont=dict(size=15,
color='red')
)])
    fig.update_layout(scene=dict(
        xaxis=dict(nticks=5,
range=[-1, 5], showticklabels=False,
showgrid=False, showline=False,
showbackground=False),
        yaxis=dict(nticks=5,
range=[-1, 5], showticklabels=False,
showgrid=False, showline=False,
showbackground=False),
        zaxis=dict(nticks=5,
range=[-1, 5], showticklabels=False,
showgrid=False, showline=False,
showbackground=False),

```

ditengah - tengah sehingga angka - angka tersebut akan **membentuk suatu kubus** yang dapat diputar ataupun *dizoom-in* dan *dizoom-out*.

Catatan: Hasil cetakan tiga dimensi ini akan terlihat berbeda dengan cetakan dua dimensi yang ada diterimal. Namun, hasil objectiveFunction yang divisualisasikan pada showInitiateCube, showFinalCube, ataupun showBestCube tetap konsisten karena seluruhnya memvisualisasikan dari koordinat yang sama (Dimulai dari bawah kiri dan orientasi sumbunya berbeda dengan dua dimensi).

	<pre>), title="Initial State Magic Cube - Kelompok 46", margin=dict(l=0, r=0, b=0, t=30)) fig.show() </pre>		
22	<p>showFinalCube(cube)</p> <p>Kode:</p> <pre> # Untuk kubus akhir def showFinalCube(cube): x, y, z, text = [], [], [], [] for i in range(5): for j in range(5): for k in range(5): x.append(i) y.append(j) z.append(k) text.append(str(cube[i][j][k])) #Hasil cetakan final dan initiate tetap konsisten walau seperti terlihat berbeda pada cetakan 2D fig = go.Figure(data=[go.Scatter3d(x=x, y=y, z=z, mode='text', text=text, textposition="middle center", textfont=dict(size=15, color='green'))]) fig.update_layout(scene=dict(xaxis=dict(nticks=5, </pre>	Semua	<p>Fungsi ini sebenarnya memiliki konsep yang sama dengan showInitiateCube. Fungsi ini dibuat untuk memudahkan main.py dalam memanggil hasil final Magic Cube. Yang membedakan adalah warna angka pada showFinalCube adalah hijau, sedangkan untuk showInitiateCube adalah merah.</p>

	<pre> range=[-1, 5], showticklabels=False, showgrid=False, showline=False, showbackground=False), yaxis=dict(nticks=5, range=[-1, 5], showticklabels=False, showgrid=False, showline=False, showbackground=False), zaxis=dict(nticks=5, range=[-1, 5], showticklabels=False, showgrid=False, showline=False, showbackground=False),), title="Final State Magic Cube - Kelompok 46", margin=dict(l=0, r=0, b=0, t=30)) fig.show() </pre>		
23	<p>showBestCube(cube)</p> <p>Kode:</p> <pre> # Untuk kubus terbaik (Genetic ALgorithm) def showBestCube(cube): x, y, z, text = [], [], [], [] for i in range(5): for j in range(5): for k in range(5): x.append(i) y.append(j) z.append(k) text.append(str(cube[i][j][k])) #Hasil cetakan final dan initiate tetap konsisten walaupun seperti terlihat berbeda pada cetakan 2D fig = </pre>	<p>- Genetic Algorithm</p>	<p>Fungsi ini sama persis dengan showFinalCube. Fungsi ini dibuat untuk mempermudah memvisualisasikan bestCube yang ada hanya pada geneticAlgorithm (jika memang berbeda dengan finalCubanya).</p>

```
go.Figure(data=[go.Scatter3d(
    x=x,
    y=y,
    z=z,
    mode='text',
    text=text,
    textposition="middle center",
    textfont=dict(size=15,
color='green')
) ])

fig.update_layout(scene=dict(
    xaxis=dict(nticks=5, range=[-1,
5]), showticklabels=False,
showgrid=False, showline=False,
showbackground=False),
    yaxis=dict(nticks=5, range=[-1,
5], showticklabels=False,
showgrid=False, showline=False,
showbackground=False),
    zaxis=dict(nticks=5, range=[-1,
5], showticklabels=False,
showgrid=False, showline=False,
showbackground=False),
),
title="Best State Magic Cube
(Genetic Algorithm) - Kelompok 46",
margin=dict(l=0, r=0, b=0, t=30))

fig.show()
```

II.3 Penjelasan Algoritma Local Search

Ada banyak algoritma *local search* yang dapat digunakan dalam mengatasi masalah *diagonal magic cube*. Setiap algoritma akan memiliki kelebihan dan kekurangan masing-masing saat diimplementasikan pada suatu masalah. Berikut adalah penjelasan dari beberapa algoritma *local search* yang digunakan pada proses pencarian *local search* terbaik untuk mengatasi masalah *diagonal magic cube*.

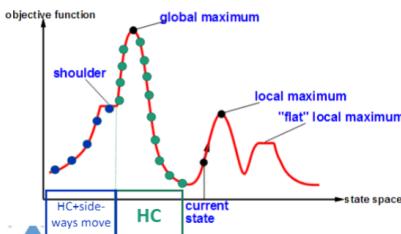
1. Steepest Ascent Hill-climbing:

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current <- MAKE-NODE(problem.INITIAL-STATE)
  loop do
    neighbor <- a lowest-valued successor of current // Neighbor total_error terendah
    if neighbor.VALUE <= current.VALUE then return current.STATE
    current <- neighbor
```

Metode *local search* ‘Steepest Ascent Hill-climbing’ adalah variansi paling dasar dari *local search hill-climbing*. Metode ini akan melakukan iterasi untuk mengevaluasi semua kemungkinan tetangga yang lebih baik dari kondisinya sekarang. Jika dia menemukan yang lebih baik maka akan dilakukan penukaran dua angka pada kubus. Penukaran dilakukan ke tetangga yang memiliki peningkatan terbesar (pendakian curam). Penukaran dilakukan hingga tidak ada lagi tetangga yang lebih besar dengannya.

Steepest Ascent Hill-climbing menjadi salah satu metode yang cukup mudah diimplementasikan. Akan tetapi, terdapat beberapa kasus yang membuat metode ini kurang optimal diterapkan karena beberapa hasil akhir cenderung terjebak pada optimum lokal, bukan pada optimum global (hasil yang terbaik). Hal tersebut karena solusi optimum global mungkin sudah terlewat dari proses iterasi pencarian sekarang.

Dalam permasalahan *diagonal magic cube* 5x5x5, penerapan *Steepest Ascent Hill-climbing* menjadi metode yang efisien untuk mendapatkan solusi yang cepat. Akan tetapi, solusi yang didapatkan akan cenderung kurang optimal dari yang diinginkan.



Gambar 3. Grafik Hill-climbing yang Melewatan Solusi optimum Global

2. Hill-climbing with Sideways Move

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current <- MAKE-NODE(problem.INITIAL-STATE)
  sideways_move <- 0
  max_sideways_move <- 1000 // Batas maksimal sideways move, bisa diganti
  loop do
    neighbor <- a highest-valued successor of current
    if neighbor.VALUE < current.VALUE then return current.STATE
    else if neighbor.VALUE == current.VALUE then
      if sideways_moves >= max_sideways_moves then
        return current.STATE
      sideways_moves <- sideways_moves + 1
    else
      sideways_moves <- 0
    current <- neighbor
```

Metode *local search* ‘Hill-climbing with Sideways Move’ merupakan metode yang cukup mirip dengan metode *Steepest Ascent Hill-climbing*. Hal yang membedakannya adalah jika *Steepest Ascent Hill-climbing* hanya bergerak ke tetangga yang terbaik, maka *Hill-climbing with Sideways Move* memperbolehkan *state* sekarang untuk bergerak ke nilai yang setara dengannya (gerakan samping) dengan syarat jika tidak ada lagi tetangga yang lebih baik.

Hal tersebut bertujuan untuk menghindari terjebak di dataran tinggi dengan cara mencari celah dengan bergerak ke tetangga yang setara dengannya (bagian *shoulder* pada gambar 3). Akan tetapi, metode ini masih bisa membuat *local search* terjebak di minimum lokal.

Jika metode *Hill-climbing with Sideways Move* diterapkan dalam permasalahan *diagonal magic cube* 5x5x5, maka terdapat kesempatan untuk mencari solusi alternatif dari kondisi yang setara dengan saat ini. Namun, ada kemungkinan untuk terjebak dari *shoulder* kondisi yang setara. Selain itu, meskipun tidak terjebak dari *shoulder*, maka masih ada kemungkinan maksimum global untuk terlewatkan dari terminasi hasil solusi terakhir.

3. Random Restart Hill-climbing:

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  best <- MAKE-NODE(problem.INITIAL-STATE)
  repeat nmax times
    current <- MAKE-NODE(problem.RANDOM-STATE())
    loop do
```

```

neighbor <- a highest-valued successor of current
if neighbor.VALUE <= current.VALUE then
    break
    current <- neighbor
if current.VALUE > best.VALUE then
    best <- current
return best.STATE

```

Metode *local search* ‘Random Restart Hill-climbing’ bekerja dengan logika algoritma seperti *Steepest Ascent Hill-climbing*, hanya saja yang membedakan adalah ketika tidak ada lagi tetangga yang lebih besar dari *state* sekarang, maka iterasi akan dimulai dari awal kembali dengan konfigurasi acak.

Hal ini bertujuan untuk menghindari algoritma terjebak di maksimum lokal dan meningkatkan peluang menemukan solusi global. Meskipun metode ini lebih efektif daripada *Steepest Ascent Hill-climbing* biasa, tetapi *Random Restart Hill-climbing* membutuhkan komputasi lebih rumit karena memerlukan beberapa kali *restart*.

Dalam permasalahan *diagonal magic cube* 5x5x5, implementasi algoritma *Random Restart Hill-Climbing* mampu menghindari hasil maksimum lokal pada penyusunan atau konfigurasi setiap angka penyusun *diagonal magic cube*. Namun, algoritma ini dapat menjadi sangat lambat karena membutuhkan banyak pengulangan percobaan dari posisi acak, terutama jika restart dilakukan sangat sering tapi tidak pernah mendekati solusi yang optimal.

4. Stochastic Hill-climbing:

```

function HILL-CLIMBING(problem) returns a state that is a local maximum
    current <- MAKE-NODE(problem.INITIAL-STATE)
    repeat nmax times
        neighbor <- a random successor of current
        if neighbor.VALUE > current.VALUE then current <- neighbor
    return current.STATE

```

Metode *local search* ‘Stochastic Hill-climbing’ merupakan metode yang cukup mirip dengan metode *Hill-climbing with Sideways Move*. Hal yang membedakannya adalah jika *Hill-climbing with Sideways Move* hanya bergerak ke tetangga yang terbaik dan memperbolehkan berpindah ke tetangga yang setara, maka *Stochastic Hill-climbing* akan membangkitkan *random successor* secara terus menerus selama proses iterasi. Lalu, akan berpindah ke tetangganya jika tetangganya

lebih baik dari *state* sekarang. Proses iterasi akan dilakukan secara terus menerus sehingga ‘nmax times’ yang dimiliki sudah habis

Implementasi algoritma Stochastic Hill-Climbing mampu memilih langkah acak yang dapat keluar dari jebakan hasil maksimum lokal pada penyusunan atau konfigurasi setiap angka penyusun *diagonal magic cube*. Namun, pendekatan algoritma ini cenderung lambat dan tidak konsisten dalam mencapai solusi yang optimal karena tergantung pada probabilitas akan langkah yang diambil secara acak, seperti yang sudah dijelaskan sebelumnya.

5. Simulated Annealing:

```
function SIMULATED-ANNEALING(problem,schedule) returns a solution state
    inputs: problem, a problem to solve
            schedule, a mapping from time to “temperature”
    current <- MAKE-NODE(problem.INITIAL-STATE)
    for t = 1 to  $\infty$  do
        T <- schedule(t)
        if T = 0 then return current.STATE
        next <- a randomly selected successor of current
         $\Delta E$  <- next.VALUE - current.VALUE
        if  $\Delta E > 0$  then current <- next
        else
            with probability  $e^{\frac{\Delta E}{T}}$  do
                current <- next
```

Metode *local search* ‘Simulated Annealing’ merupakan metode pencarian yang menggunakan konsep saat memanaskan sebuah metal atau gelas. Pada metode ini merupakan konsep turunan dari *Stochastic Hill-climbing* yang akan berpindah ke tetangga yang lebih baik jika $\Delta E > 0$. Akan tetapi, hal yang membedakan *Simulated Annealing* adalah ia memiliki probabilitas yang memungkinkan perpindahan ke tetangga yang lebih buruk dengan syarat awal $\Delta E < 0$.

Jika syarat awal terpenuhi, maka akan dihitung probabilitas menggunakan rumus berikut.

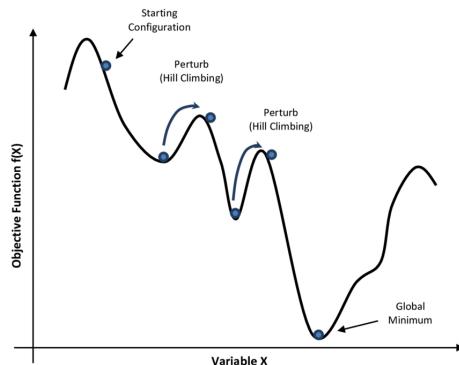
$$e^{\Delta E/T}$$

Gambar 4. Formula Probabilitas Jika $\Delta E < 0$

Keterangan:

ΔE = nilai tetangga - nilai sekarang

T = temperatur yang dapat kita atur sendiri



Gambar 5. Grafik Algoritma Simulated Annealing

Jika hasil probabilitas di atas ambang batas yang telah ditentukan sebelumnya, maka metode ini memperbolehkan perpindahan ke tetangga dengan nilai yang lebih buruk daripada nilai yang sekarang. Metode ini adalah algoritma pencarian terbaik untuk mengatasi masalah terjebaknya di minimum lokal. Akan tetapi, perlu diperhatikan lebih mendalam terkait nilai T dan ambang batas probabilitas yang digunakan.

Algoritma *Simulated Annealing* dapat menjadi sangat efektif dalam menyelesaikan permasalahan *diagonal magic cube 5x5x5* karena memungkinkan *state* saat ini menuju *neighbour* dengan nilai yang terburuk dengan tujuan untuk dapat menghindari *state* terjebak di maksimum lokal. Meskipun probabilitas kesempatan ini akan menurun seiring waktu, metode ini cukup efektif untuk mendapatkan solusi maksimum global.

Kelemahan algoritma ini adalah dalam pengaturan parameter awal, seperti “suhu” awal dan laju “pendinginan” dari algoritmanya. Kesalahan dalam pengaturan awal dapat menyebabkan algoritma terlalu lambat atau justru gagal dalam menemukan solusi yang optimal. Maka dari itu, perlu diperhatikan lebih lanjut dalam melakukan pengaturan parameter awal “suhu” dan laju “pendinginan” pada algoritma.

6. Genetic Algorithm:

```
function GENETIC-ALGORITHM(problem,FITNESS-FN) returns an individual
  inputs: problem, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual
          population <- INITIALIZE-POPULATION(problem)
  repeat
```

```

new_population <- empty set
for i= 1 to SIZE(population) do
    x <- RANDOM-SELECTION(population, FITNESS-FN)
    y<- RANDOM-SELECTION(population, FITNESS-FN)
    child <- REPRODUCE(x,y)
    if (small random probability) then child <- MUTATE(child)
    add child to new_population
    population <- new_population
until some individuals is fit enough, or enough time has elapsed
return the best individual in population, according to FITNESS-FN

```

Metode *local search* ‘Genetic Algorithm’ adalah metode pencarian yang memiliki kompleksitas dan kerumitan yang cukup tinggi. Tahapan dalam metode pencarian ini menggunakan prinsip seleksi alam, yakni seleksi, *cross over*, dan mutasi. Metode pencarian ini dimulai dengan inisialisasi beberapa *state* awal (populasi) dari masalah yang ada. Lalu, beberapa *state* awal tersebut dihitung *fitness function* (sama seperti fungsi objektif) sehingga mendapatkan persentase yang dimiliki dari masing-masing *state*.

Lalu, dilakukan *random selection* menggunakan prinsip *roulette wheel*. Pasangan yang terpilih akan dilakukan *crossover* yang bertujuan untuk menukar bagian-bagian dari masing-masing pasangan (bagian yang dibagi tersebut bebas sesuai yang diinginkan). Jika proses *crossover* telah selesai, akan dilakukan *random mutation point* yang akan mengubah nilai beberapa titik menjadi baru.

Proses tersebut akan diulang terus menerus sehingga mendapatkan hasil yang ideal dan/atau waktu yang dimiliki sudah habis. Penggunaan metode ini dapat mengeksplorasi ruang pencarian yang sangat luas, tetapi membutuhkan langkah yang rumit dan kompleks untuk dilakukan perhitungan, serta penyesuaian parameter, seperti ukuran populasi, laju mutasi, dan bagian yang dilakukan *crossover*. Oleh karena itu, metode ini cukup efektif untuk mencari masalah dalam skala besar dan tidak efektif untuk mencari solusi dari masalah dengan skala kecil.

Dalam menyelesaikan permasalahan *diagonal magic cube*, algoritma ini mampu menjelajahi ruang solusi yang lebih besar dan menghindari maksimum lokal. Namun, algoritma ini memerlukan banyak evaluasi individu yang tentunya memakan waktu yang lama dan performanya sangat bergantung pada desain operator pada proses seleksi, mutasi, dan *crossover*. Terlebih lagi, kompleksitas algoritma ini sangat tidak efisien diterapkan pada *diagonal magic cube* 5x5x5 yang memiliki banyak sekali kemungkinan konfigurasi solusi.

III. Hasil Eksperimen dan Analisis

Pada bagian ini akan diberikan hasil eksperimen dan analisis dari 6 metode search yang sudah didefinisikan sebelumnya. Diberikan juga visualisasi dari program yang telah dibuat untuk memudahkan proses analisis.

III.1 Hasil Eksperimen

Secara umum, hasil eksperimen akan menampilkan beberapa poin berikut:

- State awal dan akhir dari kubus
- Nilai objective function akhir yang dicapai
- Plot nilai objective function terhadap banyak iterasi yang telah dilewati
- Durasi proses pencarian

Namun, terdapat beberapa metode yang akan menampilkan penjelasan tambahan untuk membantu dalam proses analisisnya. Berikut adalah hasil eksperimen dari setiap metode yang dianalisis:

1. Steepest Ascent Hill-climbing

Metode Steepest Ascent Hill-climbing dimulai dengan membuat sebuah cube yang berisikan angka acak. Lalu, akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.1. Proses akan terus berlangsung hingga tidak ada lagi *neighbor value* yang lebih baik daripada *current value*.

Lalu, program akan memberikan kondisi terminate berupa nilai cube (kondisi cube akhir), iterationSum (jumlah iterasi terjadi hingga proses pencarian berhenti), duration (durasi waktu yang terjadi dalam proses pencarian), dan plot nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

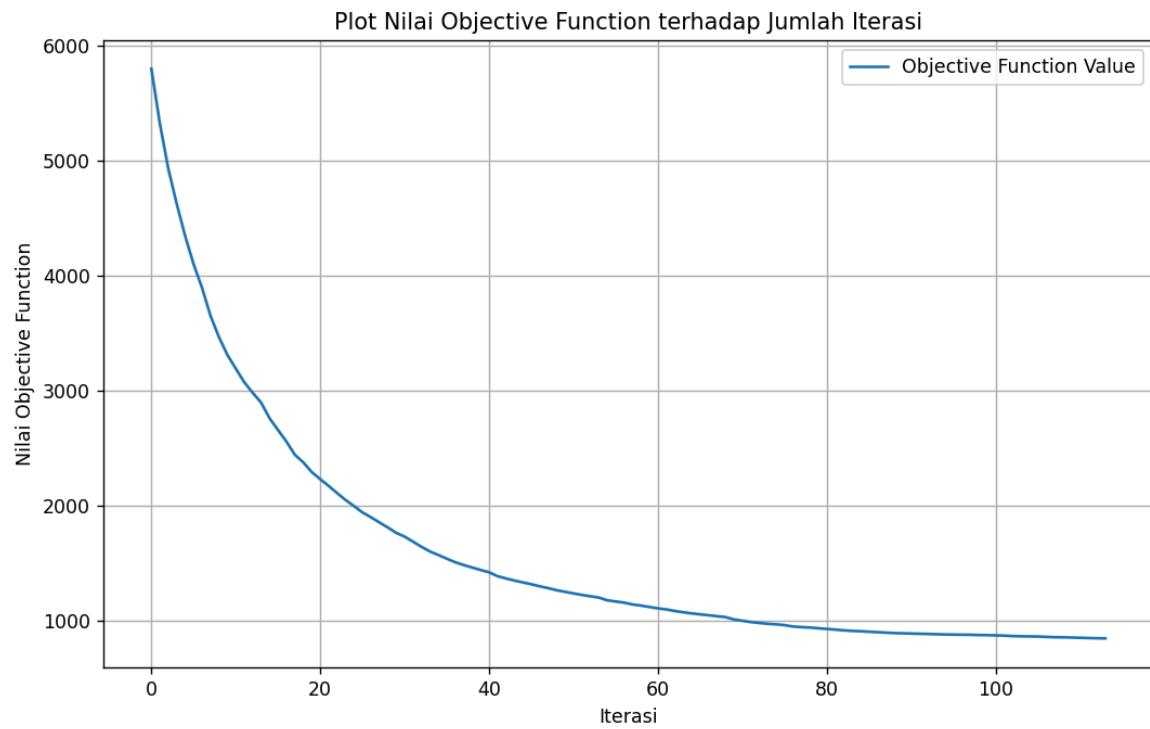
Berikut adalah hasil percobaan menggunakan metode Steepest Ascent Hill-climbing:

Percobaan 1

State Awal Cube	State Akhir Cube
-----------------	------------------

<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>42 16 186 96 98 32 56 5 79 35 93 182 68 48 44 77 36 124 59 64 37 66 8 14 61 </pre> <p>----- Sisi ke- 2 -----</p> <pre>99 90 25 26 23 82 111 101 55 80 103 28 97 100 50 109 62 107 30 72 19 27 2 112 95 </pre> <p>----- Sisi ke- 3 -----</p> <pre>76 4 46 53 10 20 52 15 29 11 119 45 41 34 121 13 67 110 81 117 17 69 83 73 9 </pre> <p>----- Sisi ke- 4 -----</p> <pre>74 123 24 85 18 12 87 115 92 86 118 116 39 122 57 49 7 70 58 47 94 3 33 108 54 </pre> <p>----- Sisi ke- 5 -----</p> <pre>51 71 104 31 88 105 65 6 43 75 78 22 91 125 21 114 38 68 1 89 84 113 40 120 63 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>42 32 93 77 37 99 82 103 109 19 76 28 119 13 17 74 12 118 49 94 51 105 78 114 84 </pre> <p>----- Sisi ke- 2 -----</p> <pre>16 56 182 36 66 98 111 28 62 27 4 52 45 67 69 123 87 116 7 3 71 65 22 38 113 </pre> <p>----- Sisi ke- 3 -----</p> <pre>106 5 60 124 8 25 101 97 107 2 46 15 41 110 83 24 115 39 78 33 184 6 91 68 40 </pre> <p>----- Sisi ke- 4 -----</p> <pre>96 79 48 59 14 26 155 100 30 112 53 29 34 81 73 85 92 122 58 108 31 43 125 1 120 </pre> <p>----- Sisi ke- 5 -----</p> <pre>98 35 44 64 61 23 88 58 72 95 10 11 121 117 9 18 86 57 47 54 88 75 21 89 63 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>42 99 76 74 51 16 90 4 123 71 106 25 46 24 104 96 26 53 85 31 98 23 10 18 88 </pre> <p>----- Sisi ke- 2 -----</p> <pre>32 82 20 12 105 56 111 52 87 65 5 101 15 115 6 79 55 29 92 43 35 80 11 86 75 </pre> <p>----- Sisi ke- 3 -----</p> <pre>93 103 119 118 78 102 28 45 116 22 68 97 41 39 91 48 100 34 122 125 44 50 121 57 21 </pre> <p>----- Sisi ke- 4 -----</p> <pre>77 109 13 49 114 36 62 67 7 38 124 107 110 70 68 59 38 81 58 1 64 72 117 47 89 </pre> <p>----- Sisi ke- 5 -----</p> <pre>37 19 17 94 84 66 27 69 3 113 8 2 83 33 40 14 112 73 108 120 61 95 9 54 63 </pre>	<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>59 20 25 115 98 34 29 120 71 56 105 124 44 6 36 77 62 5 107 64 40 80 121 14 61 </pre> <p>----- Sisi ke- 2 -----</p> <pre>89 93 106 3 23 15 88 102 51 60 22 37 87 118 52 110 76 2 32 95 78 21 18 112 85 </pre> <p>----- Sisi ke- 3 -----</p> <pre>79 19 53 26 125 103 101 13 74 24 114 17 41 54 96 12 67 100 86 50 7 111 108 75 9 </pre> <p>----- Sisi ke- 4 -----</p> <pre>47 117 49 72 38 8 73 69 92 84 94 116 39 10 57 46 45 119 58 43 109 1 33 83 97 </pre> <p>----- Sisi ke- 5 -----</p> <pre>42 66 82 99 38 122 48 11 27 91 4 16 104 123 68 70 65 98 35 55 81 113 28 31 63 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>59 34 105 77 40 89 15 22 110 78 79 103 114 12 7 47 8 94 46 109 42 122 4 70 81 </pre> <p>----- Sisi ke- 2 -----</p> <pre>20 29 124 62 80 93 88 37 76 21 19 101 17 67 111 117 73 116 45 1 66 48 16 65 113 </pre> <p>----- Sisi ke- 3 -----</p> <pre>25 128 44 5 121 106 102 87 2 18 53 13 41 100 108 49 69 39 119 33 82 11 104 90 28 </pre> <p>----- Sisi ke- 4 -----</p> <pre>115 71 6 107 14 3 51 118 32 112 26 74 54 86 75 72 92 10 58 83 99 27 123 35 31 </pre> <p>----- Sisi ke- 5 -----</p> <pre>77 110 12 46 70 62 76 67 45 65 5 2 100 119 90 107 32 86 58 35 64 95 50 43 55 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>59 89 79 47 42 29 93 19 117 66 25 106 53 49 82 115 3 26 72 99 98 23 125 30 38 </pre> <p>----- Sisi ke- 2 -----</p> <pre>34 15 103 8 122 29 88 101 73 11 120 102 13 69 11 71 51 74 92 27 56 60 24 84 91 </pre> <p>----- Sisi ke- 3 -----</p> <pre>105 22 114 94 4 124 37 17 116 16 44 87 41 39 104 6 118 54 10 123 36 52 96 57 68 </pre> <p>----- Sisi ke- 4 -----</p> <pre>77 110 12 46 70 62 76 67 45 65 5 2 100 119 90 107 32 86 58 35 64 95 50 43 55 </pre> <p>----- Sisi ke- 5 -----</p> <pre>40 78 7 109 81 80 21 111 1 113 121 18 108 33 28 14 112 75 83 31 61 85 9 97 63 </pre>
Nilai Objective Function Awal: 5798	Nilai Objective Function Akhir: 842				

Plot nilai objective function:



Banyak iterasi yang dilakukan: 113 iterasi

Durasi proses pencarian: 93609 ms

Percobaan 2

State Awal Cube	State Akhir Cube
$\begin{matrix} & & & 49 & 54 \\ & & 24 & 115 & 31 \\ & 76 & 97 & 120 & 82 \\ 107 & 104 & 88 & 14 & 75 \\ & 26 & 6 & 101 & 121 \\ & 46 & 55 & 102 & 124 \\ & 80 & 86 & 103 & 125 \\ & 22 & 33 & 104 & 115 \\ & 30 & 116 & 105 & 115 \\ & 45 & 69 & 106 & 115 \\ & 3 & 80 & 107 & 115 \\ & 82 & 118 & 108 & 115 \\ & 122 & 102 & 109 & 115 \\ & 51 & 186 & 8861 & 115 \\ & 40 & 666 & 41 & 115 \\ & & 13 & 110 & 115 \\ & & & 37 & 115 \end{matrix}$ z	$\begin{matrix} & & & 59 & 46 \\ & & 28 & 31 & 80 \\ & 76 & 97 & 120 & 39 \\ 106 & 74 & 23 & 79 & 115 \\ & 30 & 109 & 16 & 99 \\ & 44 & 108 & 91 & 49 \\ & 55 & 13 & 1070 & 89 \\ & 56 & 51 & 11096 & 52 \\ & 112 & 65 & 3 82 & 84 \\ & 21 & 111 & 120 & 92 \\ & 100 & 50 & 835 & 107 \\ & 122 & 113 & 88 & 107 \\ & 47 & 98 & 84 & 1 \\ & 24 & 117 & 89 & 101 \\ & 322 & 123 & 45 & 103 \\ & 119 & 124 & 93 & 77 \\ & 53 & 104 & 48 & 18 \\ & 40 & 148 & 78 & 60 \\ & & 54 & 43 & 85 \\ & & 67 & 103 & 66 \\ & & & 86 & 115 \end{matrix}$ z

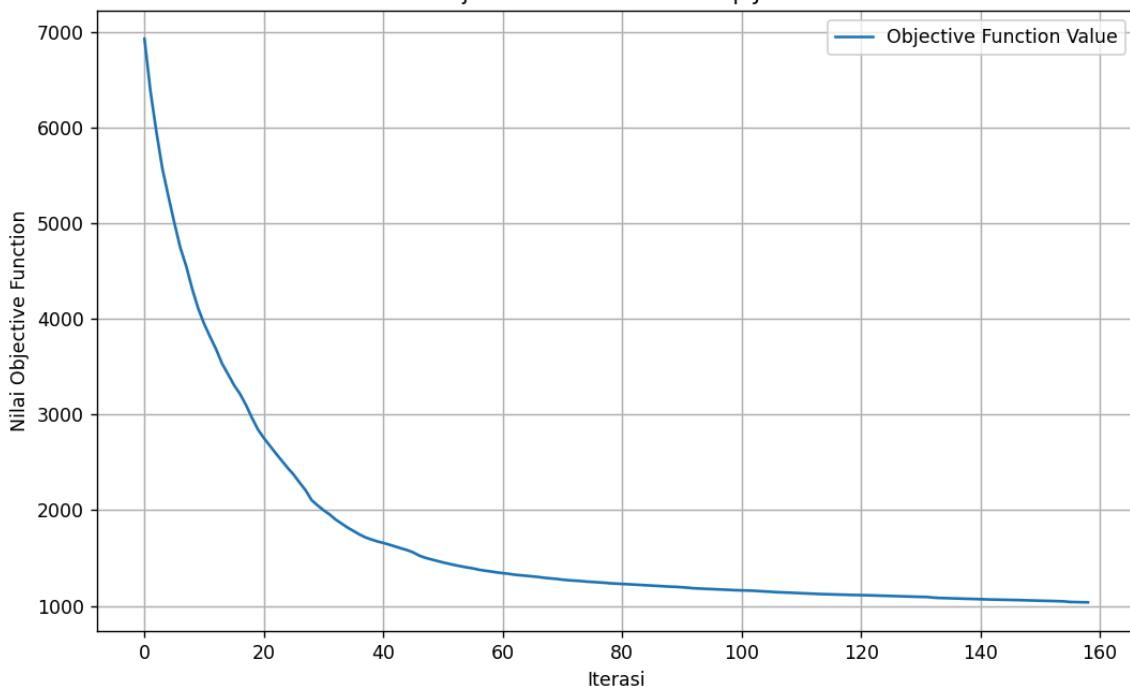
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
15 8 103 120 54 108 85 57 82 74 67 43 27 44 109 81 92 113 72 32 117 9 106 48 23	15 108 67 81 117 79 28 47 29 53 64 60 94 88 41 62 112 121 63 110 122 51 40 13 37	15 79 64 62 122 8 69 118 45 3 103 73 86 80 22 120 77 20 26 46 54 49 24 76 107	15 22 108 121 46 102 41 13 79 80 78 57 82 2 115 103 94 5 64 49 18 101 107 52 25	15 102 78 103 18 47 123 45 37 60 62 29 34 105 85 72 8 118 54 66 119 53 40 17 86	15 47 62 72 119 22 113 122 26 24 108 23 51 112 21 121 73 33 30 44 46 59 28 76 106
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
79 69 73 77 49 28 52 35 12 31 47 42 1 125 39 29 65 59 5 21 53 78 84 83 119	8 185 43 92 9 69 52 42 65 78 118 93 102 10 7 45 36 16 105 61 3 89 134 56 66	108 28 68 112 51 85 52 93 36 89 57 35 38 33 38 82 12 101 6 55 74 31 135 97 104	47 113 23 73 59 123 35 110 16 31 45 42 120 69 39 37 48 61 78 99 60 77 1 92 89	22 41 57 94 101 113 35 42 48 77 122 98 87 7 9 26 117 4 104 63 24 32 124 68 67	102 123 29 8 53 41 35 90 117 32 13 110 50 38 100 79 16 109 55 56 80 31 36 97 74
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
64 118 86 20 24 60 93 38 101 115 94 102 87 124 90 88 10 25 18 75 41 7 19 95 58	103 27 113 106 73 35 1 59 84 86 38 87 25 19 80 33 14 2 99 22 30 4 91 71	67 47 94 121 40 43 42 102 16 34 27 1 87 14 4 44 125 124 96 116 109 39 98 114 68	62 122 51 33 28 29 90 50 109 36 34 87 88 91 20 105 7 75 3 125 85 9 43 83 95	108 13 82 5 107 23 110 120 61 1 51 50 88 75 43 112 38 14 58 93 21 100 11 116 71	78 45 34 118 40 57 42 87 4 124 82 120 88 14 11 2 69 91 65 111 115 39 20 114 27
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
62 45 80 26 76 112 36 33 6 97 121 16 14 96 114 63 105 2 58 17 110 61 99 70 11	120 82 44 72 48 77 12 125 5 83 20 101 124 18 95 26 6 96 50 70 46 55 116 100 98	81 29 88 63 13 92 65 19 105 56 113 59 25 2 91 72 5 18 50 100 32 21 75 17 123	72 26 112 30 76 8 117 38 55 97 118 4 14 65 114 54 104 58 81 19 66 63 93 84 10	121 79 2 64 52 73 16 69 70 92 33 109 91 3 83 30 55 65 81 84 44 56 111 98 6	103 37 105 54 17 94 48 7 104 68 5 61 75 58 116 64 70 3 81 98 49 99 125 19 12
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
122 3 22 46 107 51 89 38 55 104 40 34 4 116 68 13 56 91 100 123 37 66 71 98 111	54 74 109 32 23 49 31 39 21 119 24 115 96 75 58 76 97 114 17 11 107 104 68 123 111	117 53 41 110 37 9 78 7 61 66 106 84 19 99 71 48 83 95 70 98 23 119 58 11 111	119 24 21 44 106 53 32 100 56 74 40 124 11 111 27 17 68 116 98 12 86 67 71 6 96	46 80 115 49 25 59 31 39 99 89 28 36 20 125 95 76 97 114 19 10 166 74 27 12 96	18 60 85 66 86 101 77 19 63 67 107 1 43 93 71 52 92 83 84 6 25 89 95 10 96

Nilai Objective Function Awal: 6928

Nilai Objective Function Akhir: 1037

Plot nilai objective function:

Plot Nilai Objective Function terhadap Jumlah Iterasi



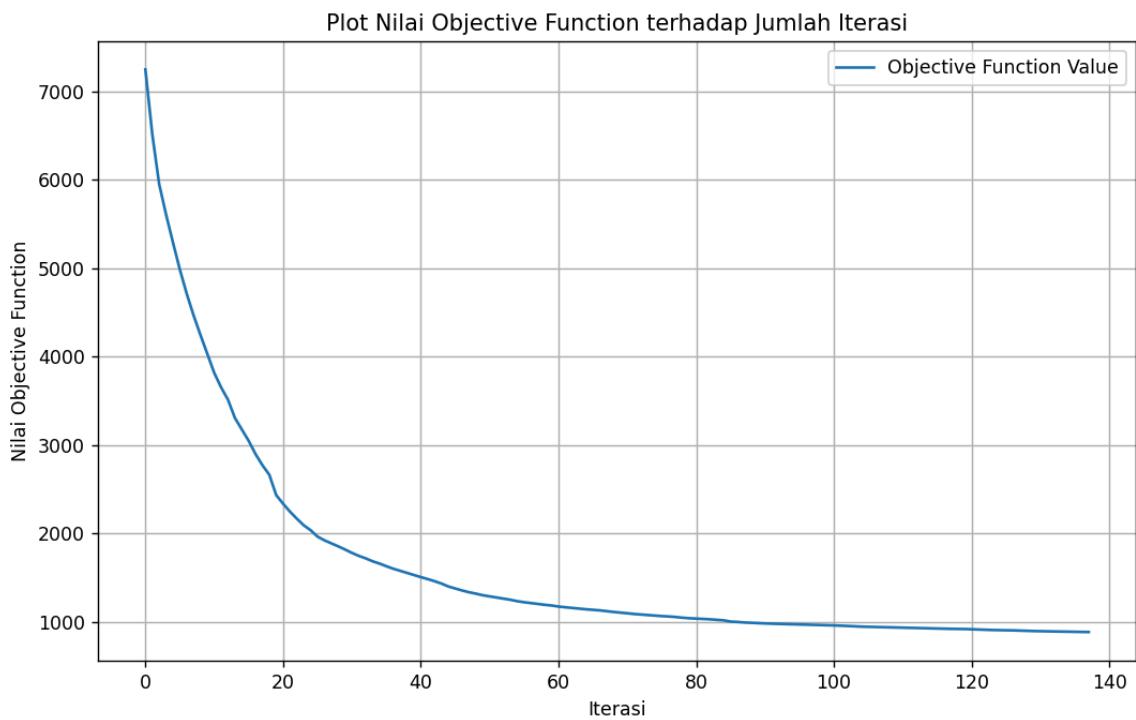
Banyak iterasi yang dilakukan: 158

Durasi proses pencarian: 129134 ms

Percobaan 3

State Awal Cube		State Akhir Cube	
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x
----- Sisi ke- 1 ----- 96 95 110 39 73 58 34 46 25 27 72 48 36 113 100 26 66 47 23 22 3 74 125 28 29 ----- Sisi ke- 2 ----- 57 54 63 71 109 56 79 11 14 60 13 35 89 112 104 99 103 107 8 116 67 53 122 44 70 ----- Sisi ke- 3 ----- 93 102 17 49 18 24 111 85 84 7 40 45 108 101 115 77 68 61 121 92 58 94 90 15 5 ----- Sisi ke- 4 ----- 32 69 1 78 37 12 83 33 76 41 98 4 9 64 81 118 52 38 43 88 59 106 105 42 91 ----- Sisi ke- 5 ----- 87 18 114 6 65 97 31 62 16 2 19 75 55 88 51 123 20 120 82 86 30 124 117 21 119	----- Sisi ke- 1 ----- 96 50 72 26 3 57 56 13 99 67 93 24 40 77 58 32 12 98 118 59 87 97 19 123 30 ----- Sisi ke- 2 ----- 95 34 48 66 74 54 79 35 103 53 102 111 45 68 94 69 83 4 52 106 10 31 75 20 124 ----- Sisi ke- 3 ----- 110 46 36 47 125 63 11 89 107 122 17 85 108 61 90 1 33 9 38 105 114 62 55 120 117 ----- Sisi ke- 4 ----- 39 25 113 23 28 71 14 112 8 44 49 84 101 121 15 78 76 64 43 42 6 16 80 82 21 ----- Sisi ke- 5 ----- 73 27 106 22 29 109 68 104 116 70 18 7 115 92 5 37 41 81 88 91 65 2 51 86 119	----- Sisi ke- 1 ----- 42 23 111 37 110 87 97 45 30 57 107 54 36 19 99 10 63 120 109 18 65 77 3 119 31 ----- Sisi ke- 2 ----- 15 55 72 88 85 34 79 111 83 31 46 11 85 33 62 25 14 84 76 16 27 60 7 41 2 ----- Sisi ke- 3 ----- 72 13 40 98 19 48 35 45 4 75 36 89 108 9 55 113 112 101 64 80 100 104 115 81 51 ----- Sisi ke- 4 ----- 26 99 77 118 123 66 103 68 52 20 47 107 61 38 120 23 8 121 43 82 22 116 92 88 86 ----- Sisi ke- 5 ----- 3 67 58 59 30 74 53 94 106 124 125 122 90 105 117 28 44 15 42 21 29 70 5 91 119	----- Sisi ke- 1 ----- 42 87 107 10 65 15 102 43 105 52 125 22 44 9 114 32 12 116 98 56 90 95 5 100 28 ----- Sisi ke- 2 ----- 23 97 54 63 77 102 91 16 13 96 43 61 76 115 20 105 94 29 53 34 52 14 122 46 80 ----- Sisi ke- 3 ----- 111 45 36 120 3 72 16 76 29 122 7 81 71 66 89 6 106 73 40 93 121 68 58 60 8 ----- Sisi ke- 4 ----- 37 30 19 109 119 88 13 115 53 46 62 75 83 84 11 124 79 25 49 35 4 117 74 21 104 ----- Sisi ke- 5 ----- 110 57 99 18 31 85 96 20 34 88 17 113 67 118 1 39 47 78 59 92 64 2 51 86 112
Nilai Objective Function Awal: 7250		Nilai Objective Function Akhir: 888	

Plot nilai objective function:



Banyak iterasi yang dilakukan: 137 iterasi

Durasi proses pencarian: 110954 ms

2. Hill-climbing with Sideways Move

Metode Hill-climbing with Sideways Move dimulai dengan membuat sebuah cube yang berisikan angka acak. Pada metode ini akan menggunakan parameter `maxSideways` yang akan menjadi batas maksimum untuk menukar nilai *neighbor* yang sama dengan *current value*. Parameter tersebut juga dapat dimasukkan ketika menjalankan program melalui CLI. Lalu, akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.2.

Proses akan terus berlangsung hingga max iterasi telah didapatkan dan/atau max sideways move juga didapatkan. Lalu, akan diberikan kondisi terminate berupa nilai cube (kondisi cube akhir), `iterationSum` (jumlah iterasi terjadi hingga proses pencarian berhenti), `duration` (durasi waktu yang terjadi dalam proses pencarian), dan `plot` nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

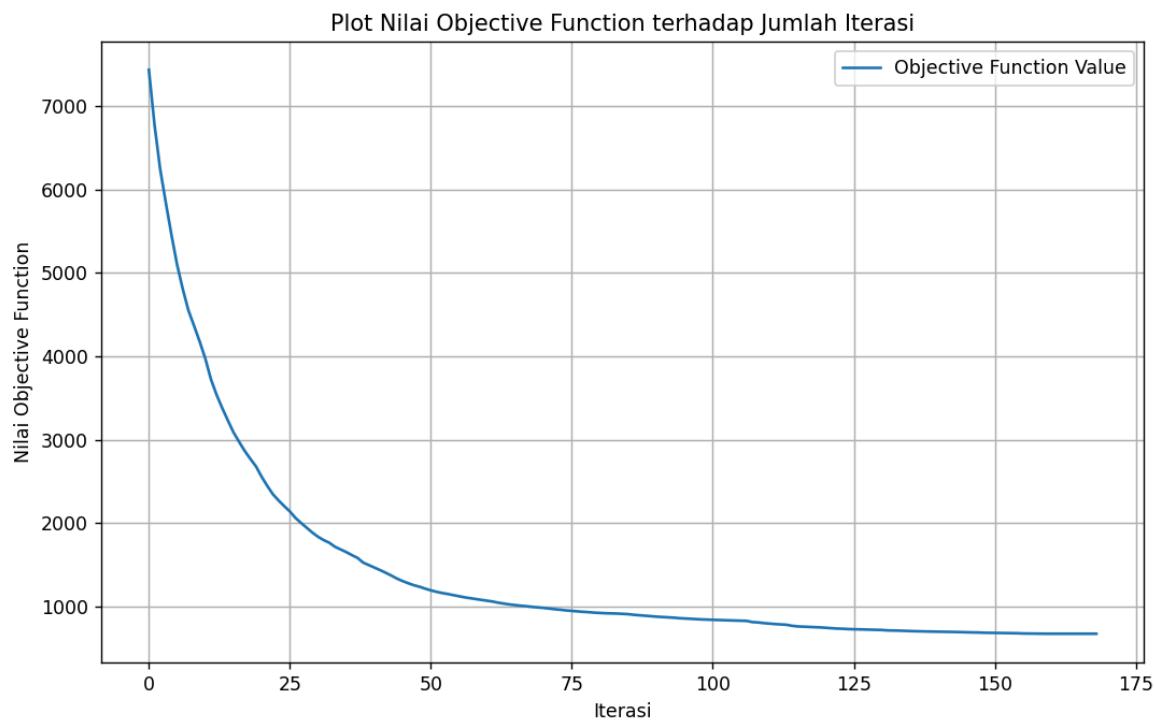
Pada percobaan ini akan dilakukan maksimum sideways move sebesar 10. Berikut adalah hasil percobaan menggunakan metode Hill-climbing with Sideways Move:

Percobaan 1

State Awal Cube	State Akhir Cube
-----------------	------------------

<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>56 21 103 105 55 88 63 40 52 92 114 62 68 39 125 50 123 93 20 74 72 59 104 38 83 </pre> <p>----- Sisi ke- 2 -----</p> <pre>90 99 70 106 14 82 118 91 112 158 57 35 69 123 45 18 25 8 64 97 49 111 42 65 28 </pre> <p>----- Sisi ke- 3 -----</p> <pre>47 109 117 16 9 36 122 107 22 44 26 3 29 75 78 76 77 124 43 84 95 48 51 94 69 </pre> <p>----- Sisi ke- 4 -----</p> <pre>66 79 85 53 15 33 108 73 113 81 67 96 34 30 12 54 6 37 115 27 5 101 46 71 61 </pre> <p>----- Sisi ke- 5 -----</p> <pre>89 1 100 120 110 41 32 10 4 121 116 2 86 11 87 13 98 80 24 119 102 31 7 17 19 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>56 88 114 50 72 90 82 57 18 49 47 36 26 76 95 66 33 67 54 5 89 41 116 13 102 </pre> <p>----- Sisi ke- 2 -----</p> <pre>21 63 62 123 59 99 118 35 25 111 109 122 3 77 48 79 188 96 6 101 1 32 2 98 31 </pre> <p>----- Sisi ke- 3 -----</p> <pre>103 40 68 93 104 70 91 60 8 42 117 107 29 124 51 85 73 34 37 46 100 10 86 80 7 </pre> <p>----- Sisi ke- 4 -----</p> <pre>105 52 39 20 38 106 112 23 64 65 16 22 75 43 94 53 113 30 115 71 120 4 11 24 17 </pre> <p>----- Sisi ke- 5 -----</p> <pre>55 92 125 74 83 14 58 45 97 28 9 44 78 84 69 15 81 12 27 61 110 121 87 119 19 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>56 90 47 66 89 21 99 109 79 1 103 70 117 85 100 105 186 16 53 128 55 14 9 15 110 </pre> <p>----- Sisi ke- 2 -----</p> <pre>88 82 36 33 41 63 118 122 108 32 40 91 107 73 10 52 112 22 113 4 92 58 44 81 121 </pre> <p>----- Sisi ke- 3 -----</p> <pre>114 57 26 67 116 62 35 3 96 2 68 68 29 34 86 39 23 75 30 11 125 45 78 12 87 </pre> <p>----- Sisi ke- 4 -----</p> <pre>50 18 76 54 13 123 25 77 6 98 93 8 124 37 80 20 64 43 115 24 74 97 84 27 119 </pre> <p>----- Sisi ke- 5 -----</p> <pre>72 49 95 5 102 59 111 48 101 31 104 42 51 46 7 38 65 94 71 17 83 28 69 61 19 </pre>	<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>58 12 109 101 35 61 88 5 57 108 113 22 66 98 16 51 125 44 19 72 32 68 92 39 84 </pre> <p>----- Sisi ke- 2 -----</p> <pre>40 99 53 111 9 81 25 85 86 59 6 55 122 10 119 115 26 7 75 82 73 110 148 33 50 </pre> <p>----- Sisi ke- 3 -----</p> <pre>54 103 23 45 91 36 105 60 62 52 13 118 47 71 64 112 21 124 31 27 94 2 42 106 77 </pre> <p>----- Sisi ke- 4 -----</p> <pre>96 79 30 34 76 89 15 74 107 28 69 123 78 14 29 49 43 37 70 116 11 56 95 90 63 </pre> <p>----- Sisi ke- 5 -----</p> <pre>67 20 100 24 104 46 83 93 8 65 114 3 4 121 87 1 97 80 120 18 102 117 38 17 41 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>58 61 113 51 32 40 81 6 115 73 54 36 13 112 94 96 89 69 49 11 67 46 114 1 102 </pre> <p>----- Sisi ke- 2 -----</p> <pre>12 88 22 125 68 99 25 55 26 110 103 105 118 21 2 79 15 123 43 56 20 83 3 97 117 </pre> <p>----- Sisi ke- 3 -----</p> <pre>109 5 66 44 92 53 85 122 7 48 23 60 47 124 42 30 74 178 37 95 100 93 4 80 38 </pre> <p>----- Sisi ke- 4 -----</p> <pre>101 57 98 19 39 111 86 10 75 33 45 62 71 31 106 34 107 14 70 90 24 8 121 120 17 </pre> <p>----- Sisi ke- 5 -----</p> <pre>32 73 94 11 102 68 110 2 56 117 92 48 42 95 38 39 33 106 98 17 84 50 77 63 41 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>58 40 54 96 67 12 99 103 79 20 109 53 23 30 100 101 111 45 34 24 35 9 91 76 104 </pre> <p>----- Sisi ke- 2 -----</p> <pre>61 81 36 89 46 88 25 185 15 83 5 85 60 74 93 57 86 62 167 8 108 59 52 28 65 </pre> <p>----- Sisi ke- 3 -----</p> <pre>113 6 13 69 114 22 55 118 123 3 66 122 47 78 4 98 10 71 14 121 16 119 64 29 87 </pre> <p>----- Sisi ke- 4 -----</p> <pre>51 115 112 49 1 125 26 21 43 97 44 7 124 37 80 19 75 31 70 120 72 82 27 116 18 </pre> <p>----- Sisi ke- 5 -----</p>
Nilai Objective Function Awal: 7440	Nilai Objective Function Akhir: 669				

Plot Nilai Objective Function:



Banyak Iterasi yang Dilakukan: 168
Banyak Sideways Move yang Dilakukan: 10
Durasi Proses Pencarian: 142416 ms

Percobaan 2

State Awal Cube				State Akhir Cube			
		x	y			x	y
z							
101	30	107	3	71	93	47	108
	88	53	290	61	65	34	43
	72	79	16	35	94	18	105
	51	121	123	105	125	56	1167
64	73	389	63	57	692	23	19
	59	75	29	122	50	87	1178
	76	20	39	96	48	77	27
103	37	69	22	110	95	40	14
	25	69	110	119	144	91	58
	28	113	119	104	103	41	46
89	132	180	180	160	5	98	26
	38	13	424	424	8045	14	14
	54	83	66	642	84	84	84
	102	62	52	49	4	85	21
		x	y			x	y
		18	14				

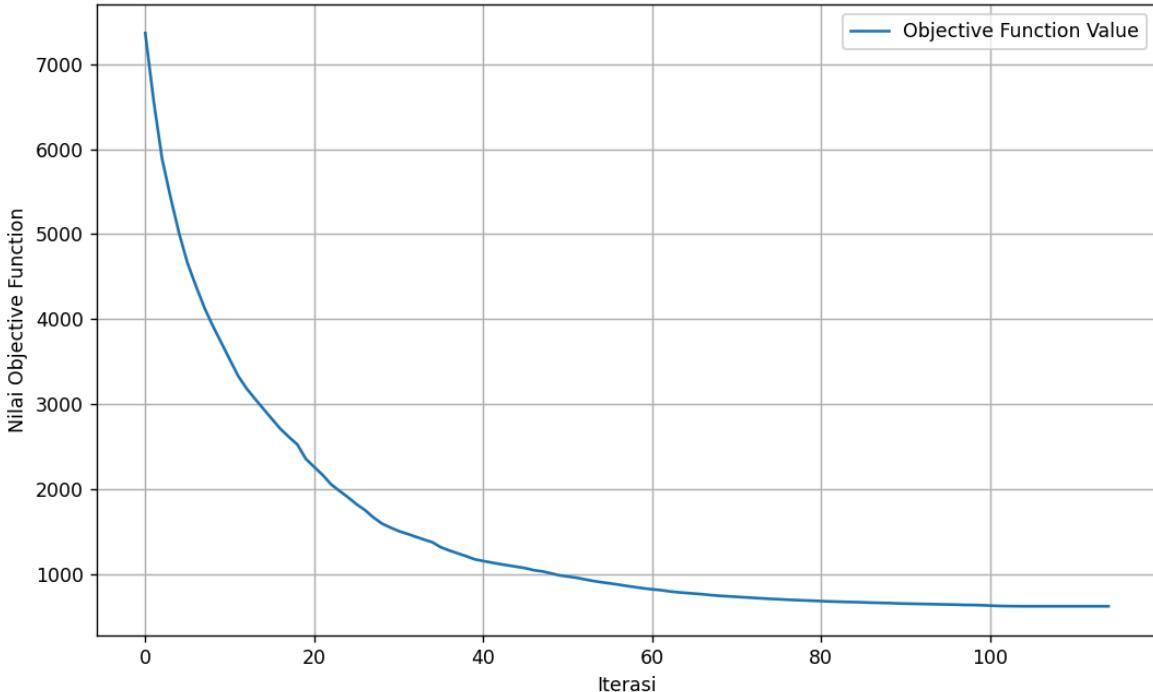
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
95 50 105 2 71 109 48 57 61 47 104 100 7 34 198 98 12 78 97 43 14 46 27 19 86	95 109 104 98 14 111 115 17 89 84 33 1 44 66 85 32 31 8 62 4 38 54 102 49 21	95 111 33 32 38 50 39 69 28 89 105 36 75 76 103 2 79 121 51 64 71 3 107 38 101	60 72 109 4 70 32 49 59 120 52 63 100 9 77 66 47 3 110 107 41 113 104 5 7 86	60 32 63 47 113 61 122 2 54 84 23 33 69 105 85 125 31 81 48 14 38 94 103 62 18	60 61 23 125 38 72 80 45 28 98 109 21 64 74 46 4 117 124 30 48 70 36 53 55 101
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
111 39 36 79 3 115 118 125 35 93 17 114 87 18 119 80 83 77 92 68 84 26 82 23 55	50 118 114 83 26 39 118 114 83 26 69 120 10 60 45 28 9 110 124 42 89 112 15 13 52	109 115 1 31 54 48 118 120 9 112 57 125 122 37 25 61 35 123 73 59 47 93 53 88 72	61 80 21 117 36 122 24 37 27 98 2 115 92 10 114 54 71 83 87 15 84 25 82 73 51	72 49 109 3 104 80 24 115 71 25 45 111 22 93 44 28 19 58 121 89 90 24 102 20 57	32 122 33 31 94 49 24 111 19 112 59 37 123 102 1 120 27 11 75 68 52 98 34 88 43
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
33 69 75 121 107 1 120 122 123 53 44 10 40 63 90 66 60 91 116 65 85 45 41 117 106	105 57 7 78 27 36 125 87 77 82 75 122 40 91 41 76 37 67 70 5 103 25 113 81 74	104 17 44 8 102 100 114 10 110 15 7 87 40 67 113 34 18 63 29 20 108 119 90 11 16	23 45 64 124 53 100 111 123 11 34 69 22 79 116 26 105 93 8 12 95 85 44 42 39 106	109 59 9 110 5 21 37 92 83 82 64 123 79 8 42 74 102 50 6 119 46 1 97 108 67	63 2 69 81 103 100 115 22 58 20 9 92 79 50 97 77 10 116 35 76 66 114 26 91 17
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
32 28 76 51 30 31 9 37 73 88 8 110 67 29 11 62 124 70 96 94 4 42 5 58 6	2 61 34 97 19 79 35 18 92 23 121 123 63 116 117 51 73 29 96 58 64 59 28 22 24	98 80 66 62 49 12 83 68 124 13 78 77 91 70 81 97 92 116 96 22 43 68 65 94 99	125 28 74 30 55 31 19 102 75 88 81 58 50 35 93 48 121 6 96 65 14 89 119 78 16	4 120 77 107 7 117 27 10 87 73 124 11 116 12 39 30 75 35 96 78 40 68 76 13 118	47 54 105 48 62 3 71 93 121 29 110 83 8 6 108 107 87 12 96 13 41 15 95 65 99
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
38 89 103 64 101 54 112 25 59 72 102 15 113 20 16 49 13 81 22 99 21 52 74 24 56	71 47 108 43 86 3 93 119 68 55 107 53 98 65 106 30 88 11 94 6 101 72 16 99 56	14 84 85 4 21 46 26 45 42 52 27 82 41 5 74 19 23 117 58 24 86 55 106 6 56	38 90 46 40 101 94 112 1 68 43 103 20 97 76 17 62 29 108 13 99 18 57 67 118 56	70 52 66 41 86 36 98 114 15 51 53 34 26 95 106 55 88 91 65 16 101 43 17 99 56	113 84 85 14 18 104 25 44 89 57 5 82 42 119 67 7 73 39 78 118 86 51 106 16 56

Nilai Objective Function Awal: 7364

Nilai Objective Function Akhir: 626

Plot Nilai Objective Function:

Plot Nilai Objective Function terhadap Jumlah Iterasi



Banyak Iterasi yang Dilakukan: 114

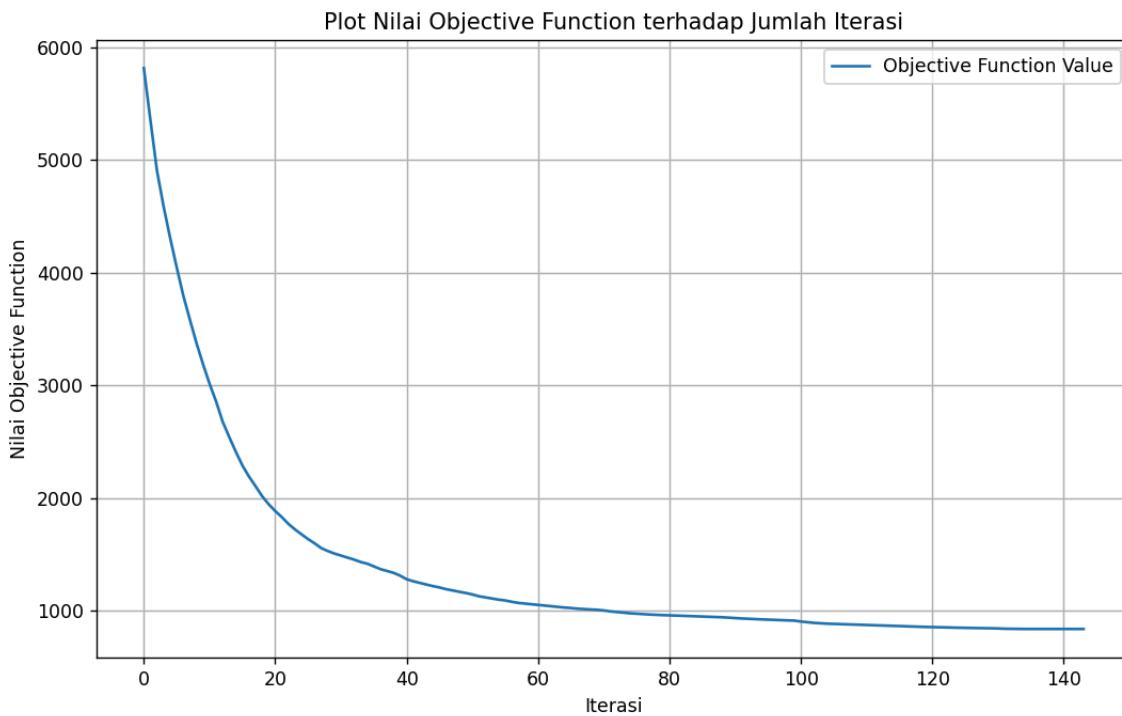
Banyak Sideways Move yang Dilakukan: 10

Durasi Proses Pencarian: 95105 ms

Percobaan 3

State Awal Cube		State Akhir Cube	
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x
<p>----- Sisi ke- 1 -----</p> 49 85 52 41 102 82 19 87 84 43 108 46 93 30 88 16 6 48 38 123 75 121 7 44 13 <p>----- Sisi ke- 2 -----</p> 111 79 55 98 29 60 51 62 69 23 12 71 83 64 40 120 28 17 58 80 33 21 91 81 42 <p>----- Sisi ke- 3 -----</p> 26 45 34 109 57 77 37 15 104 115 25 2 89 22 63 125 4 65 35 3 32 101 9 122 66 <p>----- Sisi ke- 4 -----</p> 116 50 117 92 99 8 105 78 59 86 118 112 5 14 67 24 114 103 10 90 94 107 73 39 72 <p>----- Sisi ke- 5 -----</p> 95 74 70 20 53 124 113 118 27 68 100 31 56 76 47 54 96 1 119 61 97 36 18 11 106	<p>----- Sisi ke- 1 -----</p> 49 82 108 16 75 111 60 12 120 33 26 77 25 125 32 116 8 118 24 94 95 124 100 54 97 <p>----- Sisi ke- 2 -----</p> 85 19 46 6 121 79 51 71 28 21 45 37 2 4 101 50 105 112 114 107 74 113 31 96 36 <p>----- Sisi ke- 3 -----</p> 52 87 93 48 7 55 62 183 17 91 34 15 89 65 9 117 78 5 103 73 70 110 56 1 18 <p>----- Sisi ke- 4 -----</p> 41 84 30 38 44 98 69 64 58 81 109 104 22 35 122 92 59 14 10 39 20 27 76 119 11 <p>----- Sisi ke- 5 -----</p> 102 43 88 123 13 29 23 40 80 42 57 115 63 3 66 99 86 67 98 72 53 68 47 61 106	<p>----- Sisi ke- 1 -----</p> 49 111 26 116 95 85 79 45 50 74 52 55 34 117 70 41 98 109 92 20 102 29 57 99 53 <p>----- Sisi ke- 2 -----</p> 82 60 77 8 124 19 51 37 105 113 87 62 15 78 110 84 69 104 59 27 43 23 115 86 68 <p>----- Sisi ke- 3 -----</p> 108 12 25 118 100 46 71 2 112 31 93 83 89 5 56 30 64 22 14 76 88 40 63 67 47 <p>----- Sisi ke- 4 -----</p> 25 34 45 115 98 91 77 17 28 102 44 4 78 125 64 121 79 80 35 2 32 120 93 12 58 <p>----- Sisi ke- 5 -----</p> 101 7 112 85 10 1 100 24 106 86 117 108 5 15 70 3 99 109 29 76 94 9 66 74 72	<p>----- Sisi ke- 1 -----</p> 53 75 51 73 61 41 119 11 113 31 25 91 44 121 32 101 1 117 3 94 97 37 92 8 82 <p>----- Sisi ke- 2 -----</p> 123 19 42 6 124 119 50 65 67 16 11 122 84 62 36 113 26 59 18 96 31 23 98 111 60 <p>----- Sisi ke- 3 -----</p> 49 95 103 13 55 22 65 84 59 90 45 17 78 80 93 112 24 5 109 66 87 114 46 54 14 <p>----- Sisi ke- 4 -----</p> 38 83 30 118 47 57 67 62 18 111 115 28 125 35 12 85 106 15 29 74 20 27 81 116 71 <p>----- Sisi ke- 5 -----</p> 52 43 89 110 21 107 16 36 96 60 98 102 64 2 58 10 86 70 76 72 48 68 56 33 105
Nilai Objective Function Awal: 5813	Nilai Objective Function Akhir: 839		

Plot Nilai Objective Function:



Banyak Iterasi yang Dilakukan: 143
 Banyak Sideways Move yang Dilakukan: 10
 Durasi Proses Pencarian: 122882 ms

3. Random Restart Hill-climbing

Metode Random Restart Hill-climbing dimulai dengan membuat sebuah cube yang berisikan angka acak. Lalu, akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.3. Ketika sudah tidak ada lagi best neighbor yang dapat ditukar, program akan melakukan restart dan membuat cube baru secara random. Selain itu, fungsi ini akan memiliki parameter maximum restart yang akan menghentikan pencarian jika sudah mencapai maximum restart.

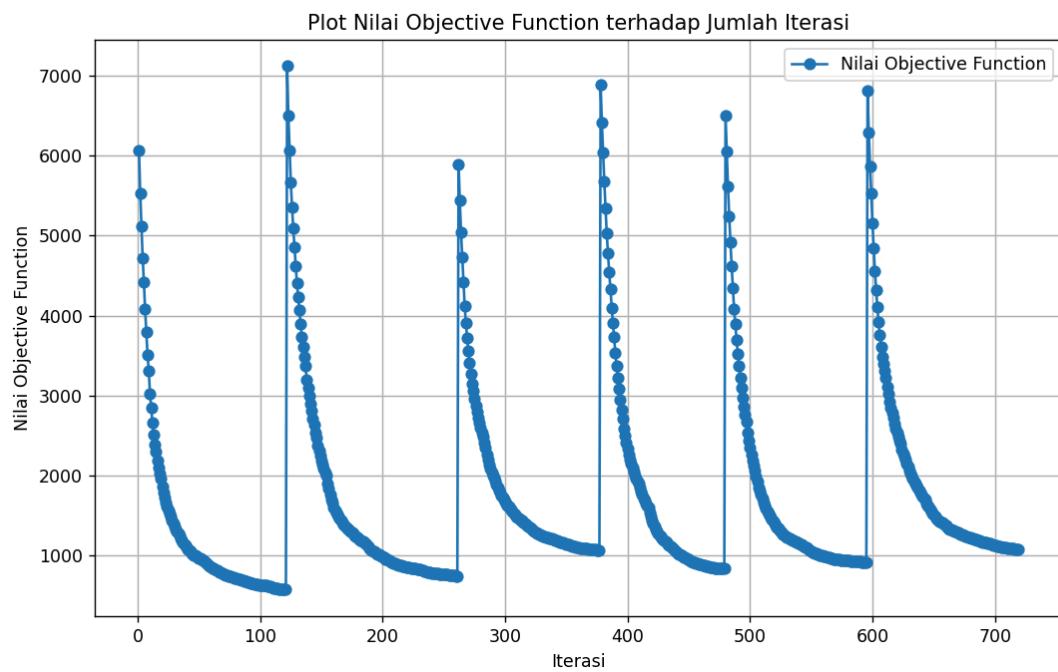
Proses akan terus berlangsung hingga maximum restart telah habis. Lalu, kondisi terminate yang akan memberikan nilai cube (kondisi cube akhir), restartCount (jumlah restart yang terjadi), iterationPerRestart (jumlah iterasi setiap restart), duration (durasi waktu yang terjadi dalam proses pencarian), dan plot nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

Percobaan kali ini dilakukan dengan mengatur jumlah restart maksimal sebesar 5. Berikut adalah hasil percobaan menggunakan metode Random Restart Hill-climbing:

Percobaan 1

State Awal Cube		State Akhir Cube	
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x
----- Sisi ke- 1 ----- 49 32 17 90 68 82 48 53 38 16 54 112 95 103 5 84 76 26 70 110 44 62 57 2 55 ----- Sisi ke- 2 ----- 10 80 116 35 73 113 105 41 50 24 86 79 99 122 63 64 89 3 125 102 7 120 52 47 21 ----- Sisi ke- 3 ----- 71 40 78 85 88 42 92 60 15 19 36 13 18 101 96 87 12 74 14 22 117 108 93 27 37 ----- Sisi ke- 4 ----- 111 67 61 100 46 123 25 106 75 83 51 1 28 124 107 33 94 97 109 23 69 66 34 121 56 ----- Sisi ke- 5 ----- 65 77 45 58 4 43 26 91 31 11 118 30 8 98 59 81 39 72 6 115 114 29 104 9 119	----- Sisi ke- 1 ----- 49 82 54 84 44 10 113 86 64 7 71 42 36 87 117 111 123 51 33 69 65 43 118 81 114 ----- Sisi ke- 2 ----- 32 48 112 76 62 88 105 79 89 129 49 92 13 12 108 67 25 1 94 66 77 28 38 39 29 ----- Sisi ke- 3 ----- 17 53 95 26 57 116 41 99 3 52 78 60 18 74 93 61 106 28 97 34 45 91 8 72 104 ----- Sisi ke- 4 ----- 99 38 103 70 2 35 50 122 125 47 85 15 101 14 27 100 75 124 109 121 58 31 98 6 9 ----- Sisi ke- 5 ----- 68 16 5 110 55 73 24 63 102 21 88 19 96 22 37 46 83 107 23 56 4 11 59 115 119	----- Sisi ke- 1 ----- 49 10 71 111 65 32 80 48 67 77 17 116 78 61 45 90 35 85 100 58 68 73 88 46 4 ----- Sisi ke- 2 ----- 82 113 42 123 43 88 105 92 25 29 53 41 60 106 91 38 50 15 75 31 16 24 19 83 11 ----- Sisi ke- 3 ----- 54 86 36 51 118 112 79 13 1 38 95 99 18 28 8 103 122 181 124 98 5 63 96 107 59 ----- Sisi ke- 4 ----- 84 64 87 33 81 76 89 12 94 39 26 3 74 97 72 70 125 14 109 6 110 102 22 23 115 ----- Sisi ke- 5 ----- 44 7 117 69 114 62 120 108 66 29 57 52 93 34 104 2 47 27 121 9 55 21 37 56 119	----- Sisi ke- 1 ----- 26 79 53 96 61 79 44 65 37 87 17 113 100 20 29 96 4 122 82 111 61 99 17 81 63 ----- Sisi ke- 2 ----- 55 44 113 4 99 90 50 10 58 107 1 54 19 124 119 34 104 91 74 12 75 66 114 13 49 ----- Sisi ke- 3 ----- 115 65 100 22 17 83 10 19 91 114 93 52 60 16 94 27 68 69 80 72 6 120 67 106 18 ----- Sisi ke- 4 ----- 95 37 20 82 81 45 58 124 74 13 123 7 69 110 8 35 101 80 30 70 78 42 72 47 76 ----- Sisi ke- 5 ----- 98 59 6 64 86 11 36 120 46 88 89 116 167 5 38 15 14 106 108 73 102 84 18 92 21 ----- Sisi ke- 6 ----- 24 87 29 111 63 28 107 119 12 49 32 2 121 51 112 125 33 8 78 76 86 88 38 73 21
Nilai Objective Function Awal: 6071		Nilai Objective Function Akhir: 1079	

Plot Nilai Objective Function:



Banyak Restart yang Terjadi: 5

Jumlah Iterasi Setiap Restart:

- Iterasi Restart 0: 120
- Iterasi Restart 1: 139
- Iterasi Restart 2: 115
- Iterasi Restart 3: 101
- Iterasi Restart 4: 115
- Iterasi Restart 5: 123

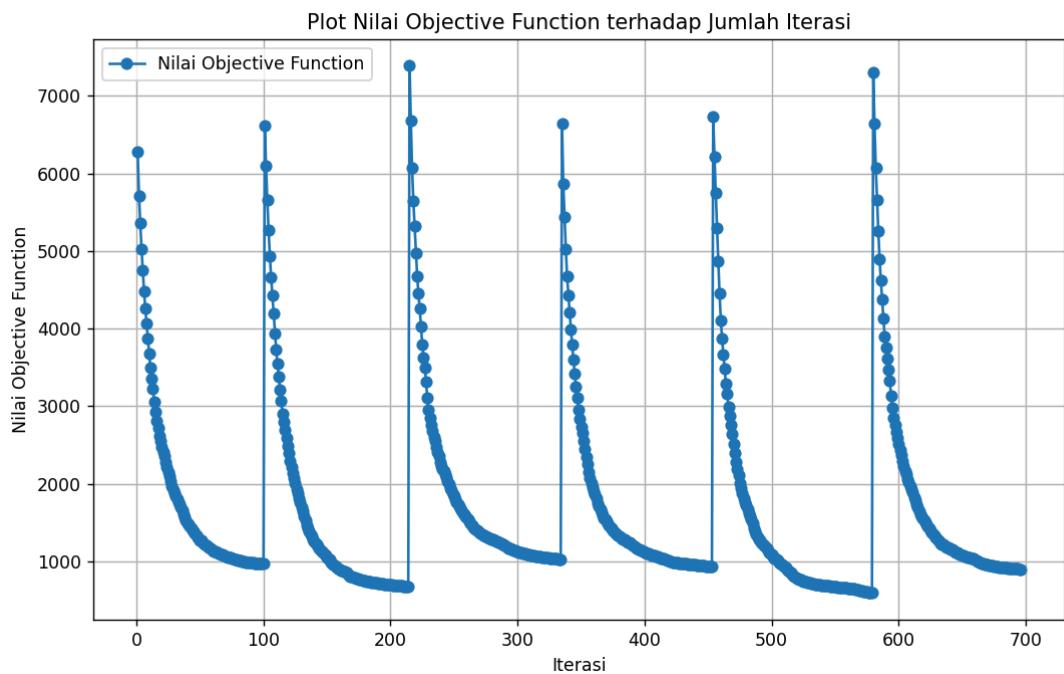
Durasi Proses Pencarian: 542162 ms

Percobaan 2

State Awal Cube	State Akhir Cube
-----------------	------------------

<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre> 43 89 59 83 45 50 26 41 22 70 96 38 33 84 67 69 29 82 57 91 27 103 53 3 121 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 31 5 99 48 14 52 123 81 79 6 49 124 94 66 111 77 120 76 12 65 122 1 46 88 117 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 56 64 71 116 87 109 35 2 72 10 60 55 16 119 15 24 95 90 68 106 85 9 125 108 115 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 114 63 8 42 30 104 74 98 13 58 25 21 61 20 44 54 17 118 11 93 75 92 80 36 28 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 107 73 40 102 78 97 86 39 47 19 113 112 51 7 34 37 18 110 105 4 32 101 62 23 100 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre> 43 50 96 69 27 31 52 49 77 122 56 109 60 24 85 114 104 25 54 75 107 97 113 37 32 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 89 26 38 29 103 5 123 124 120 1 64 135 55 95 9 63 74 21 17 92 73 86 112 18 101 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 59 41 33 82 53 99 81 94 76 46 71 2 16 98 125 8 98 61 118 80 40 39 51 110 62 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 83 22 84 57 3 48 79 66 12 88 116 72 119 68 108 42 13 20 11 36 102 47 7 105 23 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 45 70 67 91 121 14 6 111 65 117 87 10 15 196 115 30 58 44 93 28 78 19 34 4 100 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre> 43 31 56 114 107 89 5 64 63 73 59 99 71 8 40 83 48 116 42 102 45 14 87 30 78 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 50 52 109 104 97 26 123 35 74 86 41 81 2 98 39 22 79 72 13 47 70 6 10 58 19 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 96 49 60 25 113 38 124 55 21 112 33 94 16 61 51 84 66 119 20 7 67 111 15 44 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 69 77 24 54 37 29 120 95 17 18 82 76 90 118 110 57 12 68 11 105 91 65 106 93 4 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 27 122 85 75 32 103 1 9 92 101 53 46 125 80 62 3 88 108 36 23 121 117 115 28 100 </pre>	<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre> 25 81 65 67 77 114 51 10 21 119 122 74 99 7 13 20 85 33 107 70 34 19 108 113 37 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 58 98 15 44 100 117 42 110 46 3 16 12 63 101 123 115 97 8 52 43 9 66 121 72 47 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 124 93 68 30 1 18 27 45 116 109 48 83 56 86 41 14 89 98 31 91 106 23 60 58 76 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 38 11 79 82 105 64 102 111 32 6 26 118 61 49 62 71 4 59 94 88 112 87 15 57 54 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 69 29 84 80 53 2 92 39 184 78 103 28 36 73 75 95 40 125 35 17 55 120 24 22 96 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre> 25 114 122 20 34 58 117 16 115 9 124 18 48 14 106 38 64 26 71 112 69 2 103 95 55 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 81 51 74 85 19 98 42 12 97 66 16 12 63 101 123 11 102 118 14 87 29 92 28 40 120 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 65 10 99 33 108 15 110 63 8 121 68 45 56 98 60 79 111 61 59 5 84 39 36 125 24 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 67 21 7 107 113 44 46 101 52 72 30 116 86 31 58 82 32 49 94 57 80 104 73 35 22 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 77 119 13 78 37 100 3 123 43 47 1 109 41 91 76 105 6 62 88 54 53 78 75 17 96 </pre>	<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre> 25 58 124 38 69 81 98 93 11 29 65 15 68 79 84 67 44 30 82 80 77 100 1 195 53 </pre> <p>----- Sisi ke- 2 -----</p> <pre> 114 117 18 64 2 51 42 27 102 92 10 110 45 111 39 21 46 116 32 104 119 3 109 6 78 </pre> <p>----- Sisi ke- 3 -----</p> <pre> 122 16 48 26 103 74 12 83 118 28 99 63 56 61 36 7 101 86 49 73 13 123 41 62 75 </pre> <p>----- Sisi ke- 4 -----</p> <pre> 20 115 14 71 95 85 97 89 4 40 33 8 90 59 125 107 52 31 94 35 70 43 91 88 17 </pre> <p>----- Sisi ke- 5 -----</p> <pre> 34 9 106 112 55 19 66 23 87 120 108 121 60 15 24 113 72 58 57 22 37 47 76 54 96 </pre>
Nilai Objective Function Awal: 6276	Nilai Objective Function Akhir: 900				

Plot Nilai Objective Function:



Banyak Restart yang Terjadi: 5

Jumlah Iterasi Setiap Restart:

- Iterasi Restart 0: 99
- Iterasi Restart 1: 113
- Iterasi Restart 2: 119
- Iterasi Restart 3: 118
- Iterasi Restart 4: 125
- Iterasi Restart 5: 116

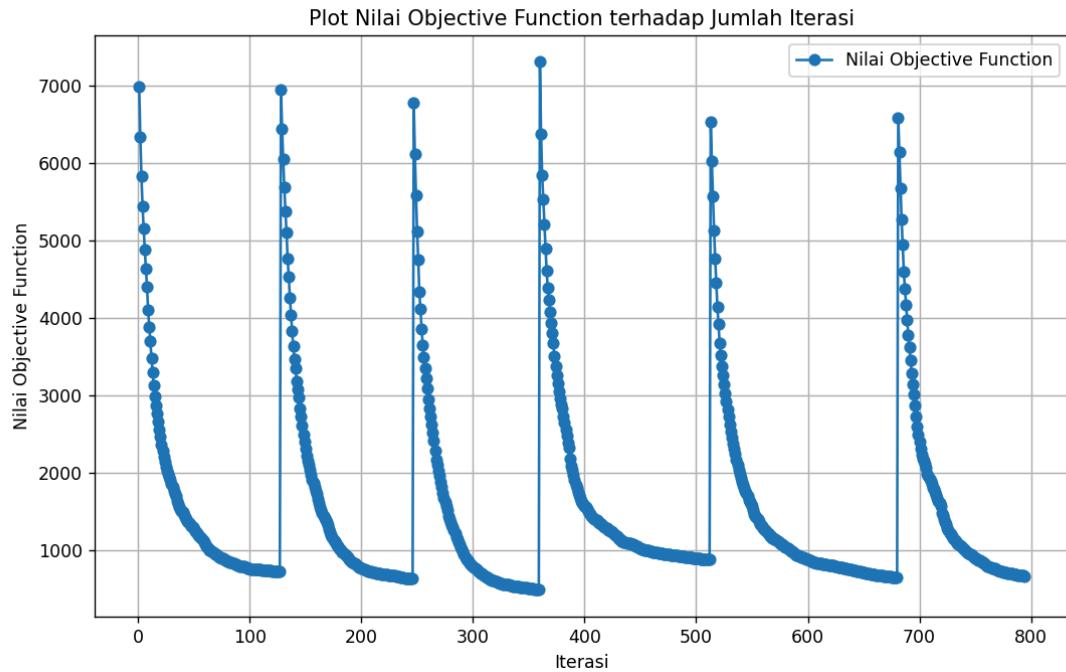
Durasi Proses Pencarian: 549890 ms

Percobaan 3

State Awal Cube	State Akhir Cube
-----------------	------------------

<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>8 110 57 12 76 35 10 79 96 103 107 34 125 95 3 21 67 113 52 45 30 58 115 2 71 </pre> <p>----- Sisi ke- 2 -----</p> <pre>97 39 1 80 55 18 87 72 33 50 94 17 36 48 4 68 7 54 120 62 22 118 78 82 83 </pre> <p>----- Sisi ke- 3 -----</p> <pre>91 119 26 24 85 32 75 53 36 89 109 59 46 66 61 104 122 23 5 105 99 112 47 123 19 </pre> <p>----- Sisi ke- 4 -----</p> <pre>73 86 64 27 70 108 43 60 65 106 56 6 98 69 117 74 14 84 81 92 28 13 88 90 15 </pre> <p>----- Sisi ke- 5 -----</p> <pre>114 63 121 111 9 101 77 116 44 102 11 25 20 37 100 41 51 124 93 48 49 31 42 16 29 </pre>	<p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>8 35 107 21 30 97 18 94 68 22 91 32 109 104 99 73 108 56 74 28 114 101 11 41 49 </pre> <p>----- Sisi ke- 2 -----</p> <pre>110 10 34 67 58 39 87 17 7 118 119 75 59 122 121 86 43 6 14 13 63 77 25 51 31 </pre> <p>----- Sisi ke- 3 -----</p> <pre>57 79 125 113 115 1 72 36 54 78 26 53 46 23 47 64 60 98 84 88 121 116 20 124 42 </pre> <p>----- Sisi ke- 4 -----</p> <pre>12 96 95 52 2 80 33 48 120 82 24 38 66 5 123 27 65 69 81 90 111 44 37 93 16 </pre> <p>----- Sisi ke- 5 -----</p> <pre>76 103 3 45 71 55 50 4 62 83 85 89 61 105 19 70 106 117 92 15 9 102 100 40 29 </pre>
<p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>8 97 91 73 114 110 39 119 86 63 57 1 26 64 121 12 80 24 27 111 76 55 85 70 9 </pre> <p>----- Sisi ke- 2 -----</p> <pre>93 44 106 56 16 1 79 115 58 71 90 42 13 124 47 8 88 78 51 99 103 59 89 106 102 </pre> <p>----- Sisi ke- 3 -----</p> <pre>38 118 21 34 104 102 4 95 65 48 55 70 57 28 105 94 61 23 122 17 27 62 119 66 41 </pre> <p>----- Sisi ke- 4 -----</p> <pre>30 89 39 72 85 125 69 19 91 9 22 96 49 45 103 98 43 88 84 5 35 20 121 26 113 45 62 105 92 40 </pre> <p>----- Sisi ke- 5 -----</p> <pre>73 63 109 32 37 30 22 99 28 49 58 118 112 13 31 115 78 47 88 42 2 82 123 90 16 71 83 19 15 29 </pre>	<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>81 3 40 114 76 58 59 75 46 77 117 97 186 18 6 7 116 60 33 92 52 36 53 112 64 </pre> <p>----- Sisi ke- 2 -----</p> <pre>93 44 106 56 16 1 79 115 58 71 90 42 13 124 47 8 88 78 51 99 123 74 2 29 87 </pre> <p>----- Sisi ke- 3 -----</p> <pre>38 118 21 34 104 102 4 95 65 48 55 70 57 28 105 94 61 23 122 17 27 62 119 66 41 </pre> <p>----- Sisi ke- 4 -----</p> <pre>114 46 10 33 112 56 50 124 51 29 34 65 28 122 66 72 91 45 84 26 32 68 108 25 82 </pre> <p>----- Sisi ke- 5 -----</p> <pre>76 77 6 92 64 16 71 47 99 87 104 48 105 17 41 85 9 103 5 113 37 111 54 101 14 </pre>
<p>Nilai Objective Function Awal: 6979</p>	<p>Nilai Objective Function Akhir: 670</p>

Plot Nilai Objective Function:



Banyak Restart yang Terjadi: 5

Jumlah Iterasi Setiap Restart:

- Iterasi Restart 0: 126
- Iterasi Restart 1: 118
- Iterasi Restart 2: 112
- Iterasi Restart 3: 152
- Iterasi Restart 4: 167
- Iterasi Restart 5: 113

Durasi Proses Pencarian: 637409 ms

4. Stochastic Hill-climbing

Metode Stochastic Hill-climbing dimulai dengan membuat sebuah cube yang berisikan angka acak. Ketika menggunakan metode ini, pengguna dapat mengatur batas jumlah iterasi maksimum yang dapat diubah melalui CLI. Lalu, akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.4.

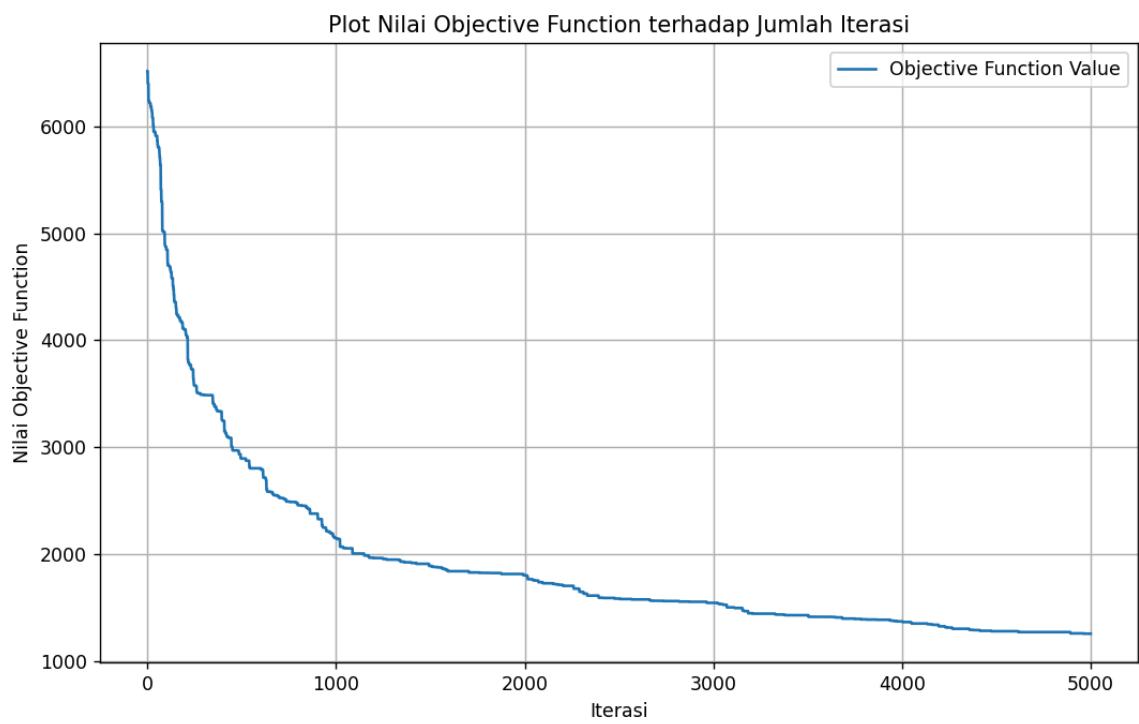
Program akan memberikan kondisi terminate berupa nilai cube (kondisi cube akhir), iterationSum (jumlah iterasi yang terjadi hingga proses pencarian berhenti), duration (durasi waktu yang terjadi dalam proses pencarian), dan plot nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

Pada eksperimen kali ini, kami mengatur batas jumlah iterasi maksimum adalah 5000. Berikut adalah hasil percobaan menggunakan metode Stochastic Hill-climbing:

Percobaan 1

State Awal Cube		State Akhir Cube	
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x
----- Sisi ke- 1 ----- 102 26 33 110 56 111 24 17 92 11 55 43 86 103 114 89 61 87 45 108 30 120 54 80 62 ----- Sisi ke- 2 ----- 34 125 75 10 83 59 23 100 37 105 19 118 12 47 69 50 91 52 5 32 48 2 94 25 21 ----- Sisi ke- 3 ----- 57 31 70 35 68 98 71 74 17 49 116 73 22 112 115 72 39 51 29 119 79 42 106 9 64 ----- Sisi ke- 4 ----- 109 38 65 40 63 123 81 4 8 36 97 60 122 121 101 3 104 99 95 53 16 1 13 113 44 ----- Sisi ke- 5 ----- 84 124 96 78 41 46 67 107 18 20 28 14 66 77 88 7 76 27 15 58 82 6 85 90 93	----- Sisi ke- 1 ----- 102 111 55 89 30 34 59 19 50 48 57 98 116 72 79 109 123 97 3 16 84 46 28 7 82 ----- Sisi ke- 2 ----- 26 24 43 61 120 125 23 118 91 2 31 71 73 39 42 38 81 60 104 1 124 67 14 76 6 ----- Sisi ke- 3 ----- 33 17 86 87 54 75 100 12 52 94 70 74 22 51 106 65 4 122 99 13 96 107 66 27 85 ----- Sisi ke- 4 ----- 110 92 103 45 80 19 37 47 5 25 35 117 112 29 9 40 8 121 95 113 78 18 77 15 90 ----- Sisi ke- 5 ----- 56 11 114 108 62 83 105 69 32 21 68 49 115 119 64 63 36 101 53 44 41 20 88 58 93	----- Sisi ke- 1 ----- 102 34 57 109 84 26 125 31 38 124 33 75 70 65 96 110 10 35 40 78 56 83 68 63 41 ----- Sisi ke- 2 ----- 111 59 98 123 46 24 23 71 81 67 17 100 74 4 107 92 37 117 8 18 11 105 49 36 20 ----- Sisi ke- 3 ----- 55 19 116 97 28 43 118 73 60 14 86 12 22 122 66 103 47 112 121 77 114 69 115 101 88 ----- Sisi ke- 4 ----- 89 50 72 3 7 61 91 39 104 76 87 52 51 99 27 45 5 29 95 15 108 32 119 53 58 ----- Sisi ke- 5 ----- 30 48 79 16 82 120 2 42 1 6 54 94 106 13 85 80 25 9 113 90 62 21 64 44 93	----- Sisi ke- 1 ----- 39 89 40 73 78 122 45 31 96 21 58 29 109 94 22 111 34 53 8 112 1 120 76 51 82 ----- Sisi ke- 2 ----- 63 122 58 111 1 63 59 71 65 52 97 10 64 57 98 26 103 16 70 101 85 19 99 30 74 ----- Sisi ke- 3 ----- 89 45 29 34 120 104 35 124 42 14 71 124 17 60 41 66 43 68 18 117 46 77 84 88 24 9 115 13 119 33 ----- Sisi ke- 4 ----- 40 31 109 53 76 5 105 17 87 110 69 95 55 6 106 125 2 83 79 15 72 102 49 91 11 ----- Sisi ke- 5 ----- 111 65 57 70 30 34 42 18 88 119 53 87 6 79 91 8 80 123 93 12 112 32 118 4 54 ----- Sisi ke- 6 ----- 1 52 90 101 74 120 14 117 24 33 76 110 106 15 11 51 28 7 113 116 82 114 3 36 81
Nilai Objective Function Awal: 6515		Nilai Objective Function Akhir: 1257	

Plot Nilai Objective Function:



Banyak Iterasi yang Dilakukan: 5000

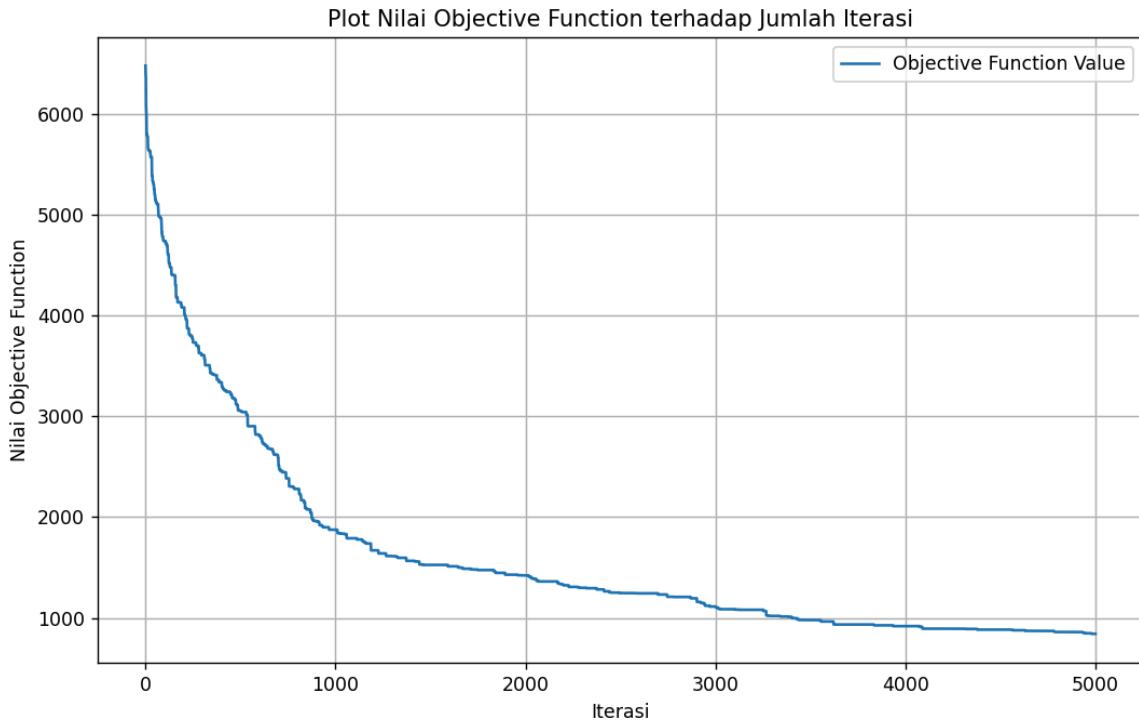
Durasi Proses Pencarian: 856 ms

Percobaan 2

State Awal Cube	State Akhir Cube
$\begin{matrix} & & & 42 & \\ & & 1 & 45 & \\ & 87 & 115 & 20 & 17 & 108 & 52 & 22 & 43 & \\ & 27 & 70 & 37 & 86 & 34 & 5 & 121 & 6 & 7 \\ & 11 & 88 & 25 & 101 & 8 & 79 & 596 & & \\ 106 & 51 & 324 & 97 & 13 & 6700 & & & 93 \\ 107 & 62 & 92 & 104 & 123 & 11165 & & & \\ 103 & 39 & 113 & 33 & 123 & 5 & & & \\ 81 & 31 & 99 & 56 & 7476 & 348 & 61 & & \\ & 69 & 119 & 49 & 124 & 105 & 102 & & \\ & 80 & 90 & 362 & 54 & 17 & & & \\ 82 & 985 & 28 & 115 & 20 & 9489 & & & \\ 71 & 158 & 784 & 115 & 16 & 60 & & & \\ 44 & 41 & 40 & 15 & 22 & 57 & & & \\ 109 & 23 & 46 & 22 & & & & & \\ & 9 & & 125 & & & & & \\ & 29 & & & & & & & \end{matrix}$ <p style="text-align: center;">z y x</p>	$\begin{matrix} & & & 52 & 29 & \\ & & 45 & 114 & 34 & 94 & \\ & & 66 & 8317 & 1013 & 106 & 92 & 43 \\ & 38 & 115 & 61 & 24 & 46 & 535 & 65 & 6317 \\ & 90 & 26 & 108 & 18 & 67 & 123 & 11347 & 12586 \\ & 6 & & & 599 & 21 & 15 & 12586 & 57 \\ & 81 & 75 & 77 & 123 & 11347 & 6862 & & \\ & 102 & 44 & 25 & 96 & 1384 & 85 & 1055 & \\ & 36 & 490 & 42 & 98 & 59 & 384 & 107 & \\ & 490 & 42 & 60 & 30 & 384 & 88 & & \\ & 91 & 110 & 89 & 99 & 5687 & 314 & & \\ & 235 & 110 & 118 & 149 & 5687 & 314 & 74 & \\ & 71 & 104 & 112 & 12876 & 12876 & 314 & & \\ & 78 & 93 & 1169 & 1169 & 1169 & 103 & & \\ & 54 & 22 & 50 & 112 & 82 & & & \\ & & 22 & 50 & 112 & 82 & & & \\ & & & 87 & & & & & x \end{matrix}$ <p style="text-align: center;">z y</p>

Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
112 33 24 83 42 4 76 13 5 52 12 105 65 79 22 94 85 48 26 43 60 77 102 93 7	112 4 12 94 60 118 38 110 117 68 40 21 73 15 57 35 53 41 23 125 71 44 109 19 29	112 118 40 35 71 33 99 31 119 82 24 32 62 103 81 83 37 70 11 106 42 45 1 115 87	99 13 67 83 52 80 95 15 101 29 1 104 62 58 94 119 31 55 17 92 39 74 107 57 43	99 80 1 119 39 89 11 97 8 103 20 23 73 116 82 35 124 93 50 14 71 78 54 22 87	99 89 20 35 71 13 98 42 70 91 67 32 77 102 36 83 46 108 6 81 52 45 61 115 38
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
118 99 32 37 45 38 124 104 101 108 110 91 55 66 34 117 10 64 100 121 68 89 19 61 6	33 76 105 85 77 99 124 91 10 89 31 95 18 50 16 119 80 58 63 122 82 90 116 130 46	4 38 21 53 44 76 124 95 80 90 13 104 113 2 69 5 101 78 51 107 52 108 17 20 27	89 98 32 46 45 11 30 113 53 114 97 37 85 65 34 8 28 88 86 106 103 120 4 72 16	13 95 104 31 74 98 30 37 28 120 42 105 49 121 2 70 64 48 111 19 91 27 79 9 112	80 11 23 124 78 95 30 185 64 27 15 113 96 25 40 101 53 18 69 75 29 114 66 24 98
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
40 31 62 70 1 21 95 113 78 17 73 18 56 97 86 15 50 47 111 8 57 16 75 3 59	24 13 65 48 102 32 104 55 64 19 62 113 56 47 75 103 2 28 72 120 81 69 98 96 84	12 110 73 41 109 105 91 18 58 116 65 55 56 28 98 79 66 97 92 39 22 34 86 25 88	20 42 77 108 61 23 105 96 18 66 73 49 59 21 117 116 121 7 68 3 82 2 76 100 63	67 15 62 55 107 32 113 85 88 4 77 96 59 17 76 102 25 10 51 122 36 49 110 118 12	1 97 73 93 54 104 37 49 48 79 62 85 59 18 110 58 65 21 123 44 94 34 117 41 26
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
35 119 103 11 115 53 80 12 51 20 41 58 28 92 25 23 63 72 74 36 125 122 120 14 67	83 5 79 26 93 37 101 66 100 61 70 78 97 111 3 11 51 92 74 14 106 107 39 49 54	94 117 15 23 9 85 10 50 63 30 48 64 47 72 96 26 100 111 74 49 43 121 8 36 114	35 70 102 6 115 124 64 25 69 24 93 48 10 123 41 50 111 51 84 5 14 19 122 33 125	83 101 58 17 57 46 53 65 86 72 108 18 21 68 100 6 69 123 84 33 81 75 44 60 56	119 8 116 50 22 31 28 121 111 9 55 88 7 51 118 17 86 68 84 60 92 106 3 5 109
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
71 82 81 106 87 44 90 69 107 27 109 116 98 39 88 9 30 96 49 114 29 46 84 54 123	42 52 22 43 7 45 108 34 121 6 1 17 86 8 59 115 20 25 36 67 87 27 88 114 123	60 68 57 125 29 77 89 16 122 46 182 19 75 120 84 93 61 3 14 54 7 6 59 67 123	71 91 36 81 38 78 27 40 75 90 54 79 110 44 26 22 9 118 60 109 87 112 12 56 47	52 29 94 92 43 45 114 34 106 16 61 66 117 3 63 115 24 41 5 125 38 96 26 109 47	39 103 82 14 87 74 120 2 19 112 107 4 76 122 12 57 72 100 33 56 43 16 63 125 47

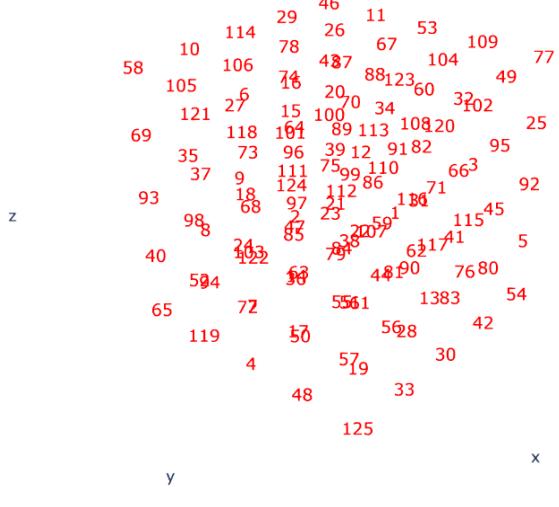
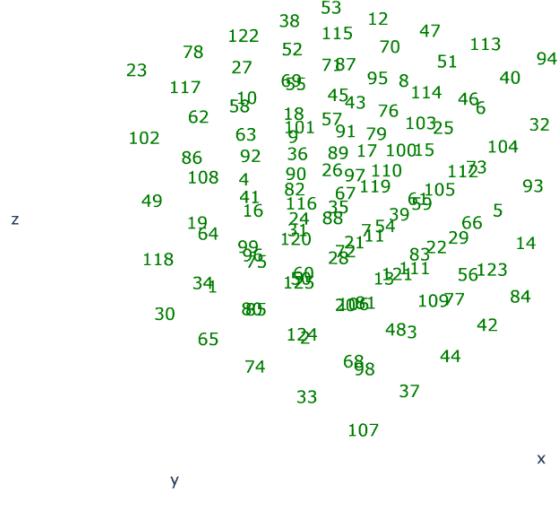
Plot Nilai Objective Function:



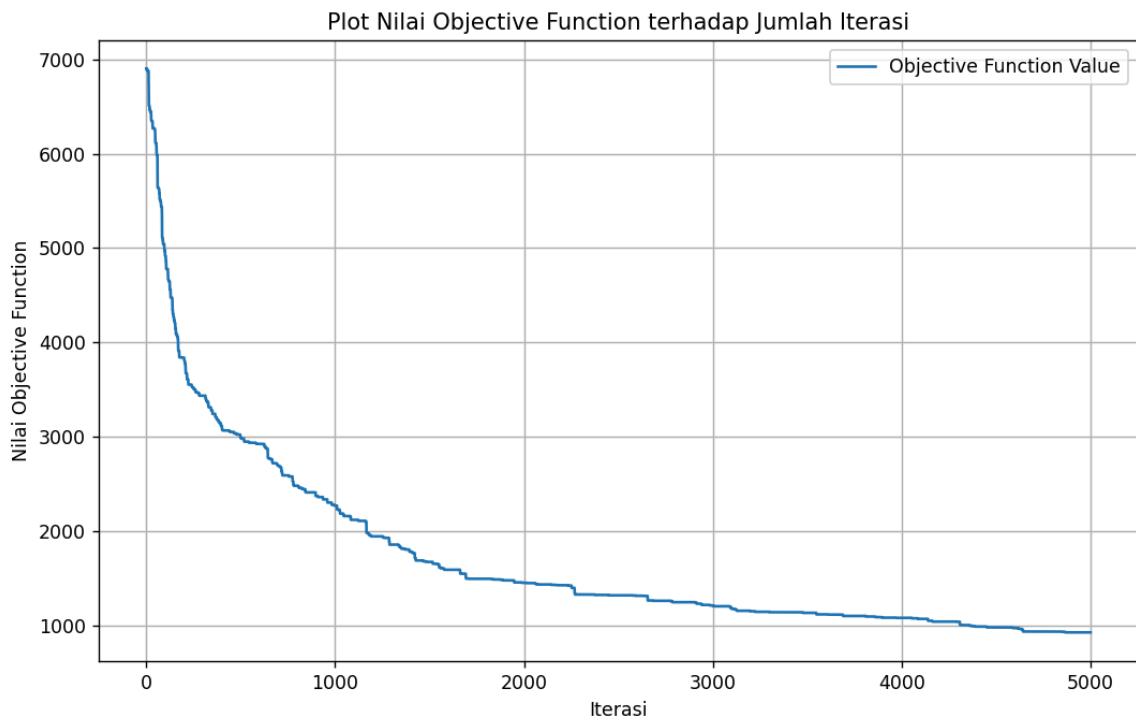
Banyak Iterasi yang Dilakukan: 5000

Durasi Proses Pencarian: 914 ms

Percobaan 3

State Awal Cube	State Akhir Cube				
					
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
<pre>----- Sisi ke- 1 ----- 23 75 100 43 46 107 86 113 88 11 62 31 82 60 53 76 115 3 102 109 54 5 92 25 77 ----- Sisi ke- 2 ----- 85 124 101 74 29 79 21 39 20 26 44 59 110 34 67 13 117 71 120 104 42 80 45 95 49 ----- Sisi ke- 3 ----- 122 68 73 6 114 36 47 111 15 78 55 84 112 89 87 56 81 1 91 123 30 83 41 66 32 ----- Sisi ke- 4 ----- 94 8 37 121 10 7 103 18 118 106 17 14 2 96 16 57 51 38 99 70 33 28 90 116 108 ----- Sisi ke- 5 ----- 65 40 93 69 58 119 52 98 35 185 4 72 24 9 27 48 50 63 97 64 125 19 61 22 12 </pre>	<pre>----- Sisi ke- 1 ----- 23 107 62 76 54 85 79 44 13 42 122 36 55 56 30 94 7 17 57 33 65 119 4 48 125 ----- Sisi ke- 2 ----- 75 86 31 115 5 124 21 59 117 80 68 47 84 81 83 8 103 14 51 28 40 52 72 50 19 ----- Sisi ke- 3 ----- 100 113 82 3 92 101 39 110 71 45 73 111 112 1 41 37 18 2 38 90 93 98 24 63 61 ----- Sisi ke- 4 ----- 115 126 108 60 102 74 20 34 120 95 6 15 89 91 66 121 118 96 99 116 69 35 9 97 22 ----- Sisi ke- 5 ----- 54 42 30 33 125 5 80 83 28 19 92 45 41 90 61 10 106 16 70 108 58 105 27 64 12 </pre>	<pre>----- Sisi ke- 1 ----- 23 88 122 94 65 75 124 68 8 40 100 101 73 37 93 43 74 6 121 69 46 29 114 10 58 ----- Sisi ke- 2 ----- 88 26 57 71 53 11 119 79 95 12 83 59 15 114 47 56 66 73 6 113 84 14 93 32 94 ----- Sisi ke- 3 ----- 120 82 9 69 38 28 35 89 45 115 13 54 110 76 70 109 22 105 25 51 42 123 5 104 40 ----- Sisi ke- 4 ----- 75 16 92 10 122 125 31 90 18 52 20 72 67 91 87 48 121 39 100 8 44 77 29 112 46 ----- Sisi ke- 5 ----- 1 64 108 62 78 85 96 41 63 27 124 50 24 36 55 68 106 21 97 43 37 3 111 61 103 ----- Sisi ke- 6 ----- 30 118 49 102 23 65 34 19 86 117 74 80 99 4 58 33 2 60 116 101 107 98 81 7 17 </pre>	<pre>----- Sisi ke- 1 ----- 88 11 83 56 84 120 28 13 109 42 75 125 26 48 44 1 85 124 68 37 30 65 74 33 107 ----- Sisi ke- 2 ----- 26 119 59 66 14 82 35 54 22 123 16 31 72 121 77 64 96 50 106 13 118 34 80 2 98 ----- Sisi ke- 3 ----- 57 79 15 73 93 9 89 110 105 5 92 90 67 39 29 108 41 24 21 111 49 19 99 60 81 ----- Sisi ke- 4 ----- 71 95 114 6 32 69 45 76 25 104 10 18 91 100 112 62 63 36 97 61 102 86 4 116 7 ----- Sisi ke- 5 ----- 53 12 47 113 94 38 115 78 51 40 122 52 87 8 46 78 27 55 43 103 23 117 58 101 17 </pre>		
Nilai Objective Function Awal: 6903			Nilai Objective Function Akhir: 924		

Plot Nilai Objective Function:



Banyak Iterasi yang Dilakukan: 5000

Durasi Proses Pencarian: 844 ms

5. Simulated Annealing

Metode Simulated Annealing dimulai dengan membuat sebuah cube yang berisikan angka acak. Lalu, akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.5. Metode ini memerlukan input parameter berupa T (temperatur), coolingRate (jumlah penurunan temperatur tiap iterasi), dan threshold (ambang batas yang membolehkan perubahan nilai jika kondisi neighbor lebih kecil dari current value).

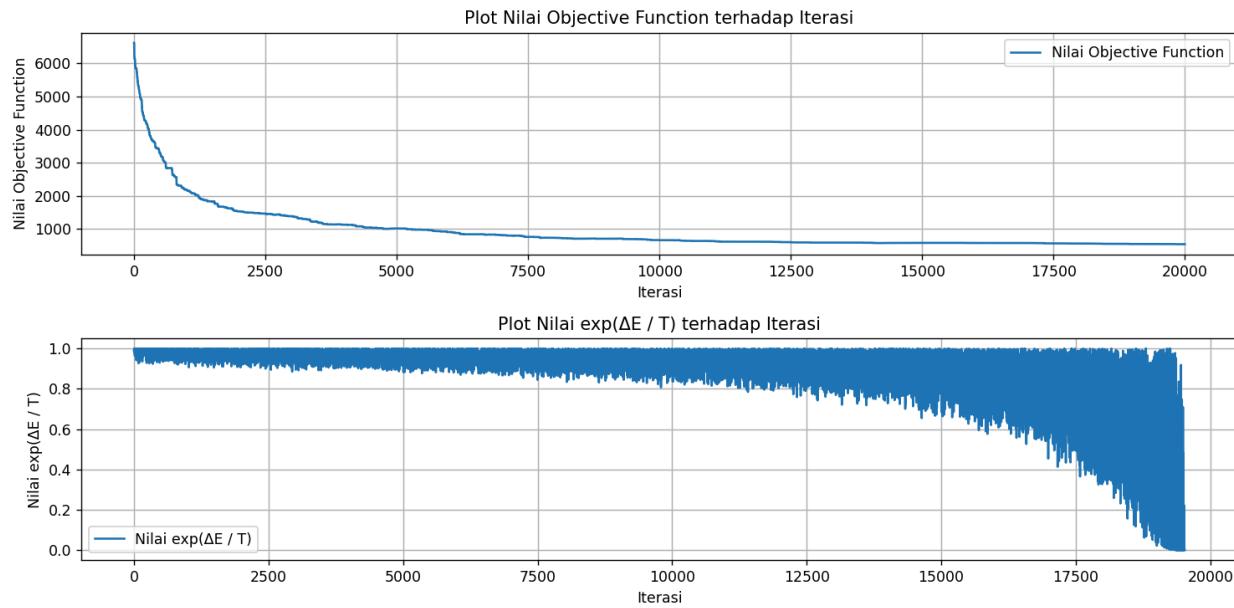
Lalu, program akan berhenti ketika nilai T sudah mencapai 0 dan/atau sudah mendapatkan nilai maksimal. Dari fungsi ini akan memberikan kondisi terminate berupa nilai cube (kondisi cube akhir), stuck (jumlah ketika nilai neighbor > current value, tetapi probabilitas yang dihasilkan di bawah threshold), duration (durasi waktu yang terjadi dalam proses pencarian), dan plot nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

Percobaan kali ini dilakukan dengan menetapkan temperatur awal sebesar 10000, penurunan suhu sebesar 0.5, dan threshold sebesar 0.9995 untuk percobaan 1 dan 2. Dan temperatur awal 1000, penurunan suhu 1, dan threshold sebesar 0.5 untuk percobaan 3. Berikut adalah hasil percobaan menggunakan metode Simulated Annealing:

Percobaan 1

State Awal Cube		State Akhir Cube	
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x
----- Sisi ke- 1 ----- 35 68 30 61 43 20 95 89 99 48 112 102 15 26 41 81 110 22 83 37 66 65 28 57 25 ----- Sisi ke- 2 ----- 104 97 1 3 14 124 12 18 105 23 96 78 5 29 74 16 115 54 39 18 63 9 117 92 101 ----- Sisi ke- 3 ----- 75 79 21 69 119 108 86 84 59 116 27 122 46 76 106 111 93 55 45 118 100 42 70 52 82 ----- Sisi ke- 4 ----- 8 103 113 80 114 53 6 32 49 50 107 13 40 120 98 33 91 56 44 60 19 17 85 109 51 ----- Sisi ke- 5 ----- 71 73 125 64 36 38 94 31 72 7 67 24 77 34 58 121 11 62 2 88 87 123 4 47 90	----- Sisi ke- 1 ----- 35 20 112 81 66 104 124 96 16 63 75 108 27 101 100 8 53 107 33 19 71 38 67 121 87 ----- Sisi ke- 2 ----- 68 95 102 110 65 97 12 78 115 9 79 86 122 93 42 103 6 13 91 17 73 94 24 11 123 ----- Sisi ke- 3 ----- 30 89 15 22 28 1 18 5 54 14 21 84 46 55 70 113 32 40 56 85 125 31 77 62 4 ----- Sisi ke- 4 ----- 61 99 26 83 57 3 105 29 39 92 69 59 76 45 52 80 49 120 44 109 64 72 34 2 47 ----- Sisi ke- 5 ----- 43 48 41 37 25 14 23 74 10 101 119 116 106 118 82 114 50 98 60 51 36 7 58 88 90	----- Sisi ke- 1 ----- 35 104 75 8 71 68 97 79 103 73 30 1 21 113 125 61 3 69 80 64 43 14 119 114 36 ----- Sisi ke- 2 ----- 101 89 120 44 40 50 41 122 34 69 120 111 45 10 28 44 99 103 58 40 62 43 104 65 ----- Sisi ke- 3 ----- 1 41 111 99 62 89 86 39 102 3 19 39 21 112 123 6 102 64 56 87 105 3 124 32 48 ----- Sisi ke- 4 ----- 70 77 7 83 81 88 26 97 17 90 24 78 63 84 60 46 54 29 118 68 95 80 119 13 16 ----- Sisi ke- 5 ----- 72 122 45 15 43 72 30 21 64 124 7 97 63 29 119 36 51 115 109 9 107 12 71 100 20	----- Sisi ke- 1 ----- 61 101 70 14 82 1 89 77 96 52 92 72 7 36 107 67 37 83 106 22 94 14 81 73 53 ----- Sisi ke- 2 ----- 50 85 88 66 25 41 86 26 49 114 122 30 97 51 12 34 75 17 59 125 69 42 90 91 23 ----- Sisi ke- 3 ----- 120 19 24 117 35 111 39 78 76 11 45 21 63 115 71 10 112 84 2 108 28 123 60 8 93 ----- Sisi ke- 4 ----- 44 6 46 98 121 99 102 54 38 18 15 64 29 109 100 103 56 118 33 5 58 87 68 31 79 ----- Sisi ke- 5 ----- 40 105 95 27 47 62 3 80 55 116 43 124 119 9 20 104 32 13 110 57 65 48 16 113 74
Nilai Objective Function Awal: 6623	Nilai Objective Function Akhir: 544		

Plot Nilai Objective Function dan Nilai $e^{\frac{\Delta E}{T}}$ terhadap Banyak Iterasi:



Frekuensi Terjadi Stuck: 19196

Durasi Proses Pencarian: 23753 ms

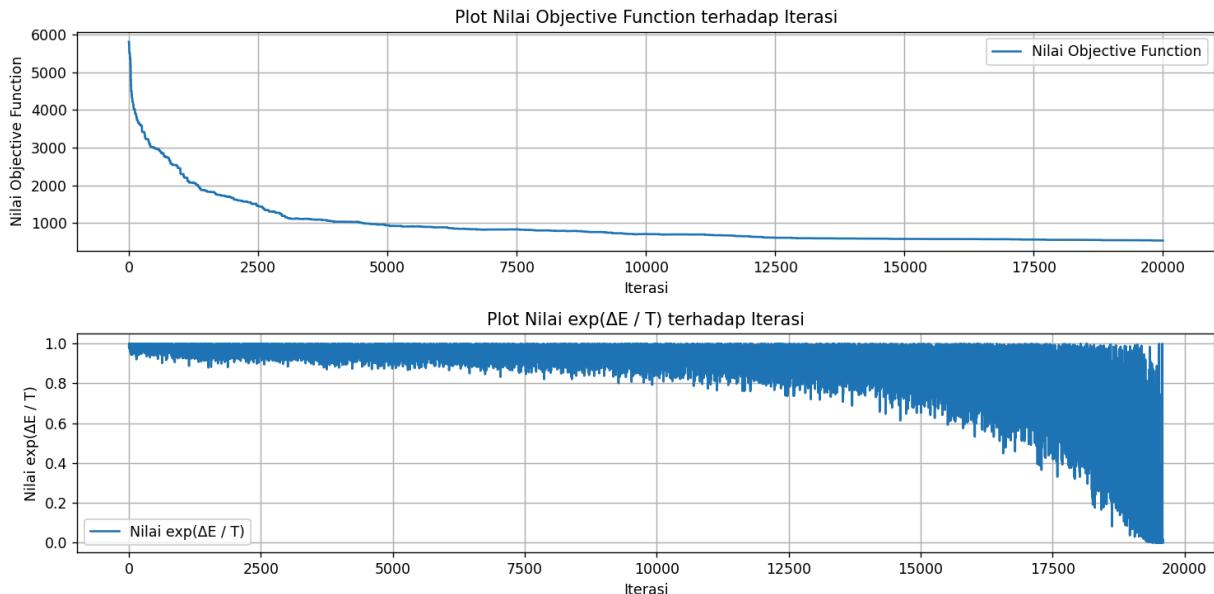
Percobaan 2

Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
66 103 124 87 56 16 119 91 33 106 26 37 62 8 101 120 53 25 27 54 42 92 75 104 15	66 16 26 120 42 18 84 112 55 39 31 35 69 19 6 98 58 94 60 68 100 7 45 46 67	66 18 31 98 100 103 80 2 3 108 124 122 97 12 102 87 52 5 59 86 56 116 57 111 77	10 13 114 88 87 48 85 79 65 49 44 113 43 14 97 112 22 46 117 18 100 80 35 33 66	10 48 44 112 100 62 122 49 28 64 59 27 105 69 56 111 11 118 30 37 71 106 9 77 53	10 62 59 111 71 13 61 29 110 102 114 76 99 6 19 88 3 90 70 75 87 121 39 17 50
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
18 80 122 52 116 84 72 83 82 99 112 64 50 115 20 55 41 29 74 76 39 93 43 73 38	103 119 37 53 92 88 72 64 41 93 2 123 51 125 23 3 14 110 65 89 108 9 22 81 10	16 84 35 58 7 119 72 123 14 9 91 83 47 21 107 33 82 113 118 117 106 99 30 13 32	62 61 76 3 121 122 98 7 67 16 40 94 31 124 24 28 8 104 74 103 64 58 96 47 51	13 85 113 22 80 61 98 94 8 58 29 41 107 101 36 110 32 1 60 115 102 57 5 123 26	48 122 27 11 106 85 98 41 32 57 79 7 45 78 108 65 67 73 109 2 49 16 125 83 42
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
31 2 97 5 57 35 123 47 113 30 69 51 85 4 121 19 125 90 61 11 6 23 1 71 95	124 91 62 25 75 122 83 58 29 43 97 47 85 90 1 12 21 28 44 105 102 107 79 34 36	26 112 69 94 45 37 64 51 110 22 62 50 85 28 79 8 115 4 63 88 101 20 121 114 24	59 29 99 90 39 27 41 45 73 125 62 50 87 68 23 12 69 101 86 38 20 56 36 15 92 116	114 79 43 46 35 76 7 31 104 96 99 45 68 86 15 6 78 82 54 95 19 108 91 25 72	44 40 105 118 9 113 94 107 1 5 43 31 68 82 91 14 124 23 21 120 97 24 12 93 89
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
98 3 12 59 111 58 14 21 118 13 94 110 28 63 114 60 65 44 70 78 68 89 105 49 17	87 33 8 27 104 52 82 115 74 73 5 113 4 61 71 59 118 63 70 49 86 117 88 40 109	120 55 19 60 46 53 41 125 65 81 25 29 90 44 34 27 74 61 70 40 54 76 11 78 96	111 110 6 70 17 11 32 78 109 83 118 1 82 21 93 30 60 54 52 119 37 115 95 63 4	88 65 14 117 33 3 67 124 74 47 90 73 23 38 92 70 109 21 52 63 75 2 120 34 84	112 28 69 30 77 22 8 101 60 123 46 104 86 54 25 117 74 38 52 34 18 103 20 119 55
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
100 108 102 86 77 7 9 107 117 32 45 22 79 88 24 46 81 34 40 96 67 10 36 109 48	56 106 101 54 15 116 99 20 76 38 57 30 121 11 95 111 13 114 78 17 77 32 24 96 48	42 39 6 68 67 92 93 23 89 10 75 43 1 105 36 104 73 71 49 109 15 38 95 17 48	71 102 19 75 50 106 57 108 2 42 9 5 91 120 89 77 123 25 34 55 53 26 72 84 81	87 49 97 18 66 121 16 24 103 51 39 125 12 20 116 17 83 93 119 4 50 42 89 55 81	100 64 56 37 53 88 58 36 115 26 35 96 15 95 72 33 47 92 63 84 66 51 116 4 81

Nilai Objective Function Awal: 5831

Nilai Objective Function Akhir: 534

Plot Nilai Objective Function dan Nilai $e^{\frac{\Delta E}{T}}$ terhadap Banyak Iterasi:



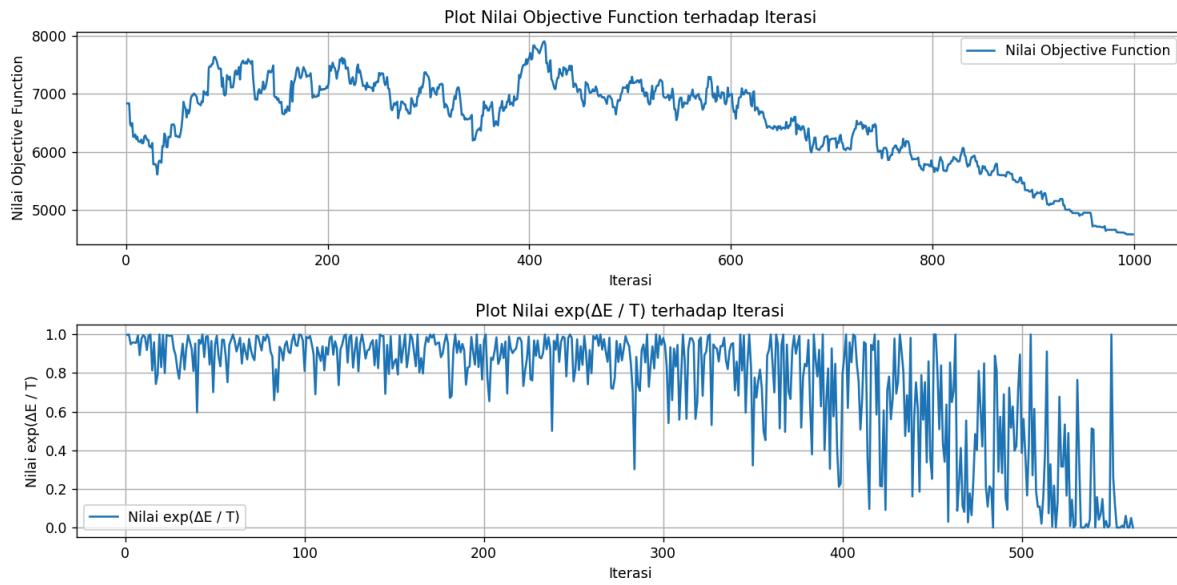
Frekuensi Terjadi Stuck: 19294

Durasi Proses Pencarian: 26302 ms

Percobaan 3

State Awal Cube		State Akhir Cube			
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 ----- 94 49 83 42 63 22 5 27 89 82 81 87 122 102 33 48 45 91 115 93 60 100 58 109 12 ----- Sisi ke- 2 ----- 18 66 55 17 9 38 11 46 108 123 77 85 73 4 78 52 90 35 99 86 67 15 79 30 31 ----- Sisi ke- 3 ----- 95 1 121 29 64 25 7 32 96 97 116 3 23 114 65 112 88 72 69 28 84 118 21 16 54 ----- Sisi ke- 4 ----- 70 40 76 101 28 50 51 111 117 104 107 26 103 57 62 44 6 68 43 41 34 10 105 56 47 ----- Sisi ke- 5 ----- 92 113 14 36 120 37 98 24 59 88 13 125 74 8 39 19 2 119 61 110 124 71 75 106 53	----- Sisi ke- 1 ----- 94 22 81 48 60 18 38 77 52 67 95 25 116 112 84 78 50 107 44 34 92 37 13 19 124 ----- Sisi ke- 2 ----- 49 5 87 45 100 66 11 85 90 15 1 7 3 88 118 40 51 26 6 10 113 98 125 2 71 ----- Sisi ke- 3 ----- 83 27 122 91 58 55 46 73 35 79 121 32 23 72 21 76 111 103 66 105 14 24 74 119 75 ----- Sisi ke- 4 ----- 42 89 102 115 109 17 108 4 99 30 29 96 114 69 16 101 117 57 43 56 36 59 8 61 106 ----- Sisi ke- 5 ----- 63 82 33 93 12 9 123 78 86 31 64 97 65 20 54 28 104 62 41 47 120 80 39 110 53	----- Sisi ke- 1 ----- 94 18 95 70 92 49 66 1 48 113 83 55 121 76 14 42 17 29 101 36 63 9 64 28 120 ----- Sisi ke- 2 ----- 22 38 25 50 37 5 11 7 51 98 27 46 32 111 24 89 108 96 117 59 82 123 97 104 80 ----- Sisi ke- 3 ----- 81 77 116 107 13 55 46 73 35 79 121 32 23 72 21 102 4 114 57 8 33 78 65 62 39 ----- Sisi ke- 4 ----- 48 52 112 44 19 45 90 88 6 2 91 35 72 68 119 115 99 69 43 61 93 86 20 41 110 ----- Sisi ke- 5 ----- 60 67 84 34 124 100 15 118 18 71 58 79 21 105 75 109 30 16 56 106 12 31 54 47 53	----- Sisi ke- 1 ----- 13 94 61 29 105 15 86 87 112 3 120 79 6 107 9 42 24 71 20 124 113 19 137 60 91 ----- Sisi ke- 2 ----- 13 15 120 45 113 115 116 14 7 51 122 28 16 90 17 56 83 49 106 77 32 117 110 11 22 ----- Sisi ke- 3 ----- 94 86 79 24 19 39 78 47 111 8 14 47 98 104 96 7 111 108 80 44 51 8 118 103 59 ----- Sisi ke- 4 ----- 61 87 6 71 37 84 42 98 108 118 97 76 40 26 119 46 50 48 73 30 12 68 57 34 21 ----- Sisi ke- 5 ----- 105 3 9 124 91 10 92 96 44 59 101 70 102 43 5 54 53 58 52 69 66 89 63 75 1		
Nilai Objective Function Awal: 6833		Nilai Objective Function Akhir: 4581			

Plot Nilai Objective Function dan Nilai $e^{\frac{\Delta E}{T}}$ Terhadap Banyak Iterasi:



Frekuensi Terjadi Stuck: 102

Durasi Proses Pencarian: 1093 ms

6. Genetic Algorithm

Metode Genetic Algorithm akan menerima input parameter berupa nilai jumlah populasi dan banyak iterasi. Lalu, akan dibuat cube setara dengan jumlah populasi yang diberikan. Setiap cube akan dilakukan proses penukaran tiap nilai di dalamnya yang sesuai dengan penjelasan bagian II.2.6.

Program akan berhenti ketika jumlah maksimum iterasi sudah tercapai. Dari fungsi ini juga akan memberikan kondisi terminate berupa nilai cube (kondisi cube akhir), duration (durasi waktu yang terjadi dalam proses pencarian), jumlah populasi, banyak iterasi, dan plot nilai fungsi objektif terhadap banyak iterasi yang telah dilewati.

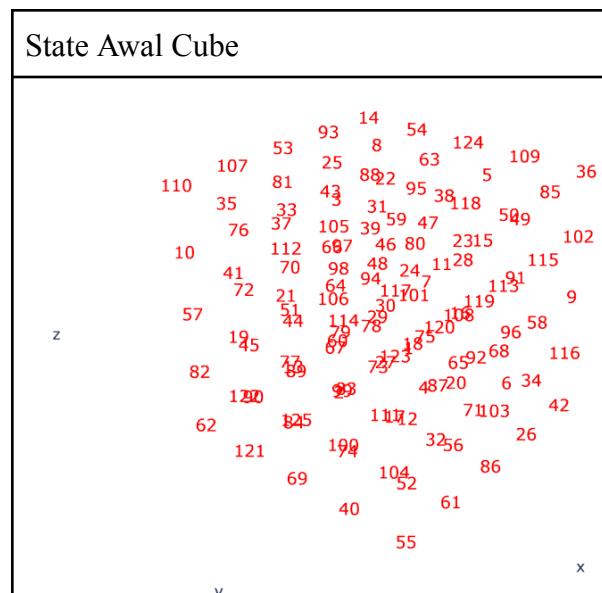
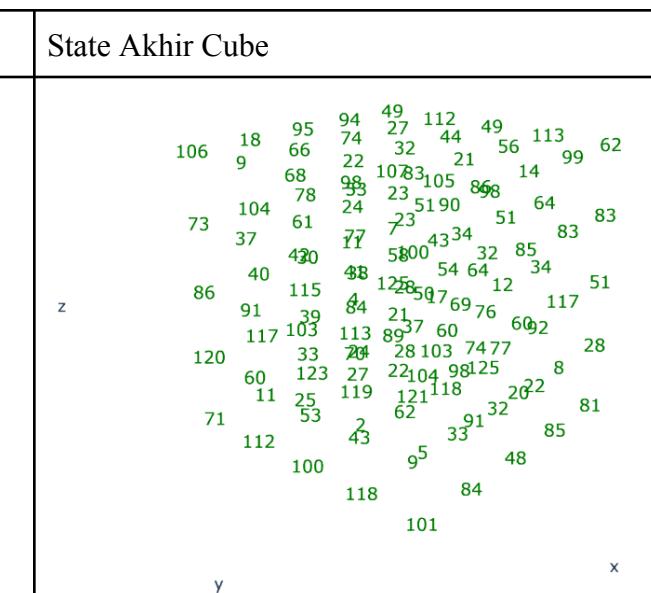
Percobaan ini akan dilakukan 2 variasi, yakni jumlah populasi sebagai kontrol dan jumlah iterasi sebagai kontrol. Berikut adalah hasil percobaan menggunakan metode Genetic Algorithm:

1. Variasi Kontrol Jumlah Populasi:

Variasi ini akan menetapkan jumlah populasi sebanyak 100 sebagai kontrol. Lalu, akan diberikan jumlah maksimum iterasi menjadi 3 jenis, yakni sebesar 10, 100, dan 1000. Berikut adalah hasil percobaan menggunakan metode Genetic Algorithm dengan variansi jumlah populasi sebagai kontrol yang ditetapkan:

1.1 - Jumlah Maksimum Iterasi 10

Percobaan 1

State Awal Cube	State Akhir Cube				
					
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z

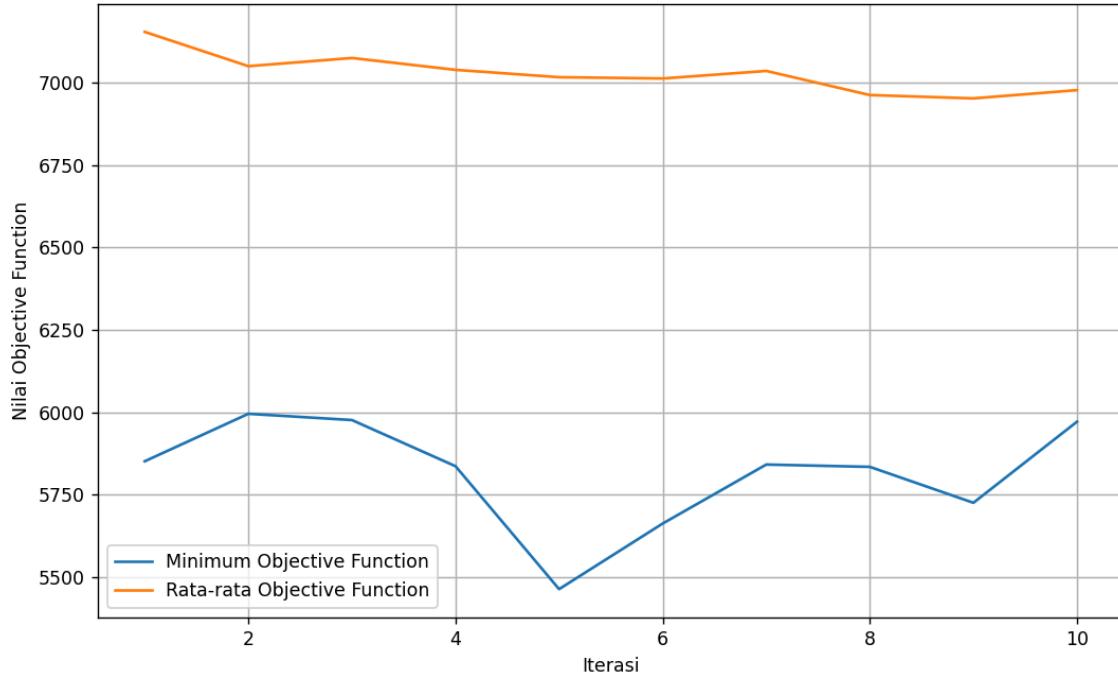
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
78 94 39 88 14 18 101 80 95 54 65 16 28 118 124 6 96 91 49 109 42 116 9 102 36	78 18 65 6 42 67 73 4 71 26 89 2 111 32 86 90 125 100 104 61 62 121 69 40 55	78 67 89 90 62 94 106 44 45 82 39 66 70 72 57 88 43 33 76 10 14 93 53 107 110	89 125 7 107 49 103 17 43 105 112 125 76 32 98 49 22 92 34 64 113 81 28 51 83 62	89 103 125 22 81 70 22 118 32 85 123 119 62 33 48 11 53 43 9 84 71 112 100 118 101	89 70 123 11 71 125 4 39 117 120 7 11 30 40 86 187 98 78 104 73 49 94 95 18 106
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
67 106 66 43 93 73 29 48 31 8 4 75 7 147 63 71 92 119 15 5 26 34 58 115 85	94 101 16 96 116 106 29 75 92 34 44 60 27 87 103 45 13 99 17 56 82 122 84 74 52	18 73 2 125 121 101 29 60 13 122 80 48 64 51 19 95 31 105 112 41 54 8 25 81 35	70 4 11 98 94 101 21 58 23 27 118 68 54 98 44 32 77 12 51 56 85 8 117 83 99	125 17 76 92 28 4 21 68 17 8 39 113 28 98 20 117 33 27 121 91 120 60 25 2 5	103 22 119 53 112 17 21 113 33 60 43 58 41 115 91 105 23 24 61 37 112 27 74 66 9
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
89 44 70 33 53 2 60 64 105 25 111 27 38 46 22 32 87 120 11 38 86 103 68 113 50	39 88 28 91 9 66 48 7 119 58 70 64 36 120 68 72 51 79 123 20 57 19 77 83 12	65 4 111 108 69 16 75 27 99 84 28 7 30 79 77 118 47 46 98 21 124 63 22 3 37	123 39 30 78 95 119 113 41 24 74 62 28 23 32 33 98 69 34 21 48 20 60 85 14	7 43 32 34 51 11 58 54 12 117 30 41 28 69 60 40 115 84 37 74 86 91 103 24 104	125 118 62 43 100 76 60 28 27 25 32 54 28 84 103 98 90 23 77 42 49 44 32 22 68
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
90 45 72 76 107 125 13 51 112 181 100 99 79 98 3 104 17 123 117 59 61 56 20 108 23	88 95 118 49 102 43 131 47 15 115 33 105 46 11 113 76 112 98 117 108 10 41 21 114 1	6 71 32 104 40 96 92 87 17 74 91 119 120 123 83 49 15 11 117 114 109 5 38 59 97	11 117 40 104 18 53 33 115 61 66 43 27 84 77 22 9 121 37 100 83 84 91 74 64 86	107 105 98 64 83 98 23 90 51 83 78 24 23 34 85 104 61 77 100 64 73 37 42 38 50	22 32 33 9 118 92 77 98 121 2 34 12 69 37 24 64 51 34 100 38 113 56 21 83 53
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
62 82 57 10 110 121 122 19 41 35 69 84 77 21 37 40 74 83 114 97 55 52 12 1 24	14 54 124 109 36 93 18 63 5 85 53 25 22 38 50 107 81 3 59 23 110 35 37 97 24	42 26 86 61 55 116 34 103 56 52 9 58 68 20 12 102 115 113 108 1 36 85 50 23 24	71 120 86 73 106 112 68 91 37 9 100 25 103 42 68 118 2 24 38 53 101 5 104 50 51	49 112 49 113 62 94 27 44 56 99 95 74 32 21 14 18 66 22 83 86 106 9 68 53 51	81 85 48 84 101 28 8 20 91 5 51 117 60 74 104 83 83 85 64 50 62 99 14 86 51

Nilai Objective Function Awal: 6965

Nilai Objective Function Akhir: 5971
(Best: 5463)

Plot Nilai Objective Function Minimum dan Rata-rata:

Plot Nilai Objective Function terhadap Jumlah Iterasi



Nilai Objective Function Terendah dari Populasi Terakhir: 5971

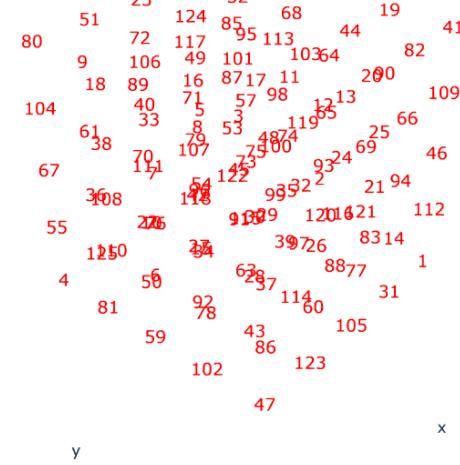
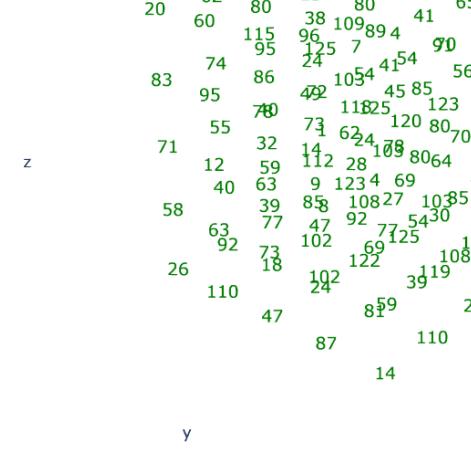
Nilai Objective Function Rata-rata dari Populasi Terakhir: 6977.04

Nilai Objective Function Terbaik dari Seluruh Populasi: 5463

Banyak Iterasi yang Dilakukan: 10

Durasi Proses Pencarian: 417 ms

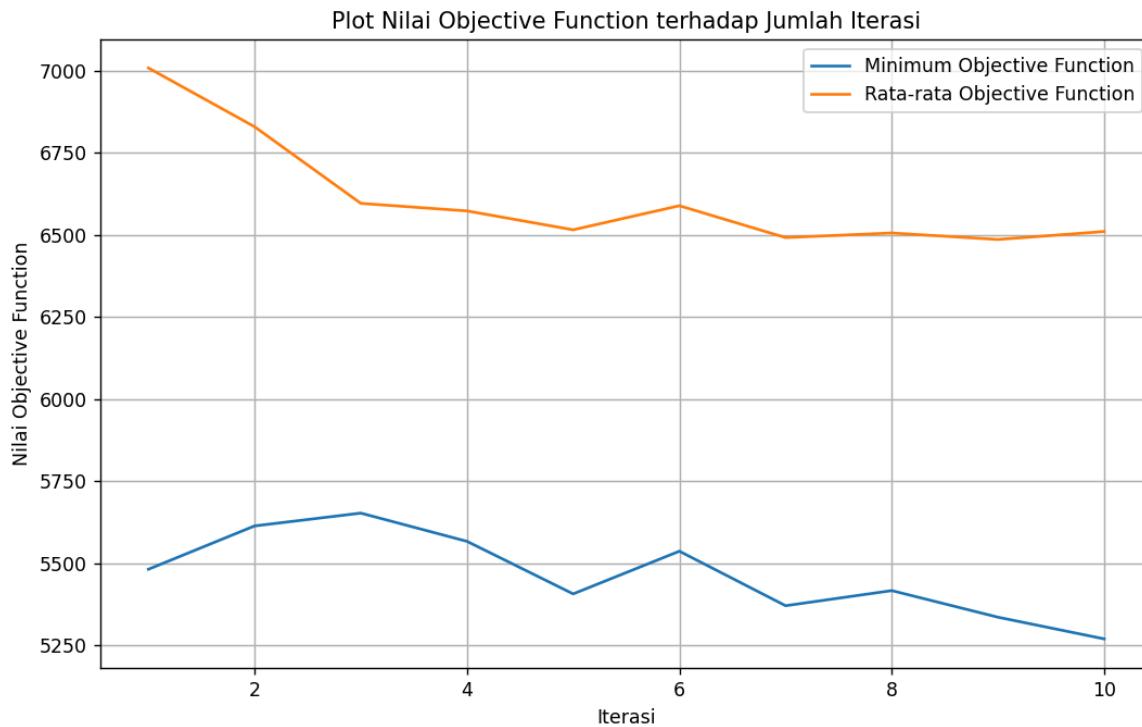
Percobaan 2

State Awal Cube		State Akhir Cube			
 <p>z</p> <p>y</p>		 <p>z</p> <p>y</p>	x		
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
----- Sisi ke- 1 ----- 122 53 87 85 58 99 100 98 113 56 128 93 65 64 84 83 21 25 90 19 1 112 46 109 41 ----- Sisi ke- 2 ----- 118 107 71 117 62 91 45 3 101 52 39 35 74 11 68 88 116 24 13 44 31 14 94 66 82 ----- Sisi ke- 3 ----- 76 7 33 196 23 27 42 79 16 124 63 115 73 157 95 114 97 32 119 103 105 77 121 69 20 ----- Sisi ke- 4 ----- 110 108 38 18 51 6 10 111 40 72 92 15 96 8 49 43 28 30 75 17 123 60 26 2 12 ----- Sisi ke- 5 ----- 4 55 67 104 80 81 125 36 61 9 59 50 22 70 89 102 78 34 54 5 47 86 37 29 48	----- Sisi ke- 1 ----- 122 99 120 83 1 118 91 39 88 31 76 27 63 114 105 110 6 92 43 123 4 81 59 102 47 ----- Sisi ke- 2 ----- 53 100 93 21 112 107 45 35 116 14 7 42 115 97 77 108 10 15 28 60 55 125 50 78 86 ----- Sisi ke- 3 ----- 87 98 65 25 46 71 3 74 24 94 33 79 73 32 121 38 111 96 30 26 67 36 22 34 37 ----- Sisi ke- 4 ----- 85 113 64 98 109 117 101 11 13 66 106 16 57 119 69 18 40 8 75 2 104 61 70 54 29 ----- Sisi ke- 5 ----- 58 56 84 19 41 62 52 68 44 82 23 124 95 103 20 51 72 49 17 12 80 9 89 5 48	----- Sisi ke- 1 ----- 123 118 76 110 4 53 107 7 108 55 87 71 33 38 67 85 117 106 18 104 58 62 23 51 80 ----- Sisi ke- 2 ----- 99 91 27 6 81 107 45 35 116 14 98 3 73 111 36 113 101 16 48 61 56 52 124 72 9 ----- Sisi ke- 3 ----- 120 39 63 92 59 93 35 115 15 150 65 74 73 96 22 64 11 57 8 70 84 68 95 49 89 ----- Sisi ke- 4 ----- 83 88 114 43 102 21 116 97 28 178 25 24 32 30 34 90 13 119 75 54 19 44 103 17 5 ----- Sisi ke- 5 ----- 1 31 105 123 47 112 14 77 60 186 46 94 121 26 137 109 66 69 12 29 41 82 20 12 48	----- Sisi ke- 1 ----- 123 62 112 98 90 60 80 13 25 99 115 95 38 109 89 74 86 41 54 41 83 95 24 7 25 ----- Sisi ke- 2 ----- 71 32 73 62 24 12 59 112 28 108 40 63 9 123 4 58 39 85 108 27 63 92 47 92 77 26 73 18 122 39 110 47 24 39 19 87 87 108 108 39 14 14 20 110 ----- Sisi ke- 3 ----- 123 27 30 93 39 85 92 125 108 119 77 102 122 39 20 92 18 24 81 110 26 110 47 87 14 ----- Sisi ke- 4 ----- 62 78 64 6 125 14 28 69 85 19 59 9 108 54 108 40 39 47 69 119 58 63 73 102 59 ----- Sisi ke- 5 ----- 27 92 102 18 110 78 28 9 39 63 45 118 73 32 12 4 7 24 86 95 99 25 13 80 60 ----- Sisi ke- 6 ----- 123 85 77 92 26 62 69 149 40 58 103 49 40 55 71 189 96 95 74 83 90 98 112 62 20 ----- Sisi ke- 7 ----- 123 125 122 24 47 64 108 108 47 73 123 120 24 112 63 70 54 54 72 78 76 35 80 38 115 ----- Sisi ke- 8 ----- 30 125 122 24 47 64 108 108 47 73 123 120 24 112 63 70 54 54 72 78 76 35 80 38 115 ----- Sisi ke- 9 ----- 93 108 39 81 87 6 85 54 69 102 106 78 80 4 8 29 56 85 125 1 1 65 41 89 125 ----- Sisi ke- 10 ----- 39 119 20 110 14 125 19 108 119 59 110 35 54 103 77 188 49 22 80 103 91 39 109 91 41		

Nilai Objective Function Awal: 6021

Nilai Objective Function Akhir: 5270
(Best: 5270)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5270

Nilai Objective Function Rata-rata dari Populasi Terakhir: 6510.54

Nilai Objective Function Terbaik dari Seluruh Populasi: 5270

Banyak Iterasi yang Dilakukan: 10

Durasi Proses Pencarian: 441 ms

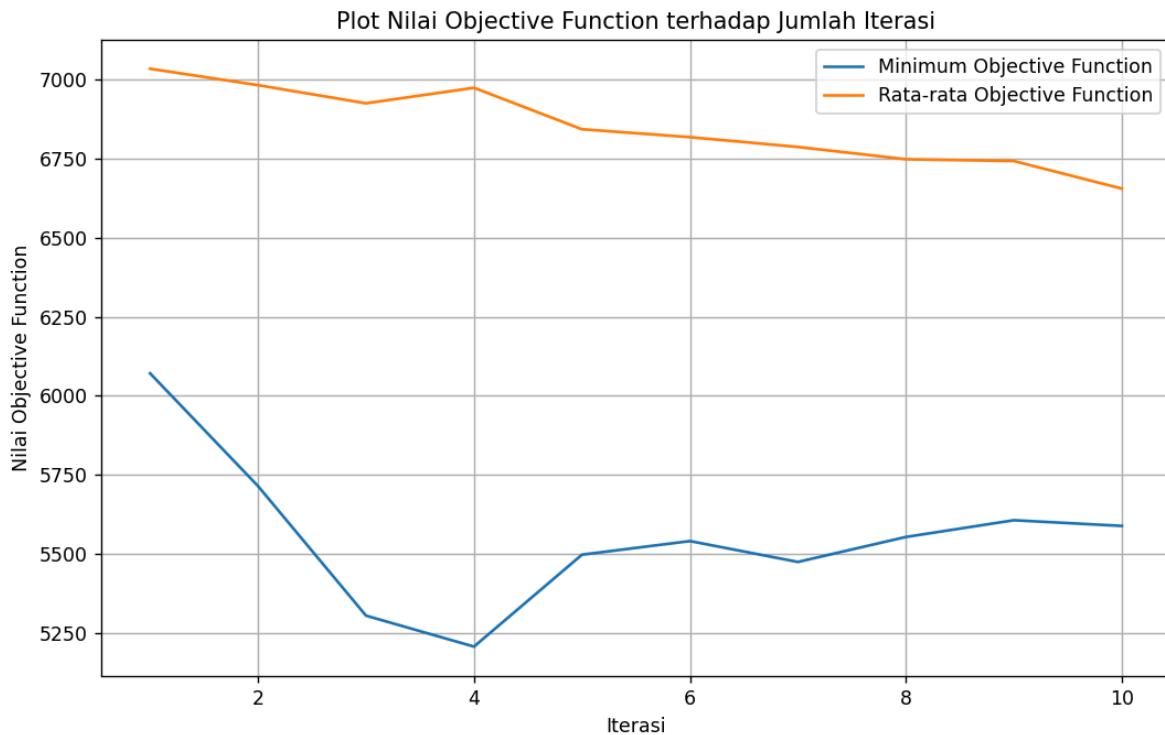
Percobaan 3

State Awal Cube

State Akhir Cube

<p>Sumbu X</p> <pre>----- Sisi ke- 1 ----- 81 52 99 14 36 57 111 119 91 94 64 27 70 69 98 53 49 51 59 114 125 108 2 101 19 ----- Sisi ke- 2 ----- 9 45 74 56 92 48 12 33 5 90 32 58 102 75 41 62 109 77 117 61 85 124 55 15 78 ----- Sisi ke- 3 ----- 80 112 50 93 79 54 63 21 37 40 38 104 66 7 6 95 25 24 88 8 87 73 10 96 65 ----- Sisi ke- 4 ----- 88 121 123 23 60 105 116 76 103 113 22 44 106 39 11 120 20 35 68 46 17 107 18 29 110 ----- Sisi ke- 5 ----- 115 47 72 97 26 16 82 100 43 30 83 122 3 4 31 86 71 1 67 34 84 42 118 28 13 </pre>	<p>Sumbu Y</p> <pre>----- Sisi ke- 1 ----- 81 57 64 53 125 9 48 32 62 85 80 54 38 95 87 88 105 22 120 17 115 16 83 86 84 ----- Sisi ke- 2 ----- 52 111 27 49 108 45 12 58 199 124 112 63 104 25 73 121 116 44 20 107 47 82 122 71 42 ----- Sisi ke- 3 ----- 99 119 70 51 2 74 33 182 77 55 58 21 66 24 10 123 76 106 35 18 72 100 3 1 118 ----- Sisi ke- 4 ----- 14 91 69 59 101 56 5 75 117 15 93 37 7 89 96 23 103 39 68 29 97 43 4 67 28 </pre>	<p>Sumbu Z</p> <pre>----- Sisi ke- 1 ----- 81 9 80 88 115 52 45 112 121 47 99 74 50 123 72 14 56 93 23 97 36 92 79 68 26 </pre>	<p>Sumbu X</p> <pre>----- Sisi ke- 1 ----- 73 118 97 21 58 19 106 4 37 13 84 50 54 61 81 78 79 7 94 19 52 62 44 35 100 </pre>	<p>Sumbu Y</p> <pre>----- Sisi ke- 1 ----- 73 19 84 70 52 92 53 59 60 67 88 34 73 79 27 109 42 117 125 104 110 4 92 100 52 </pre>	<p>Sumbu Z</p> <pre>----- Sisi ke- 1 ----- 118 92 88 109 110 118 4 72 23 110 97 55 25 78 87 21 14 27 115 78 58 63 76 67 88 </pre>
Nilai Objective Function Awal: 7355	Nilai Objective Function Akhir: 5588 (Best: 5206)				

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5588

Nilai Objective Function Rata-rata dari Populasi Terakhir: 6655.61

Nilai Objective Function Terbaik dari Seluruh Populasi: 5206

Banyak Iterasi yang Dilakukan: 10

Durasi Proses Pencarian: 434 ms

1.2 - Jumlah Maksimum Iterasi 100

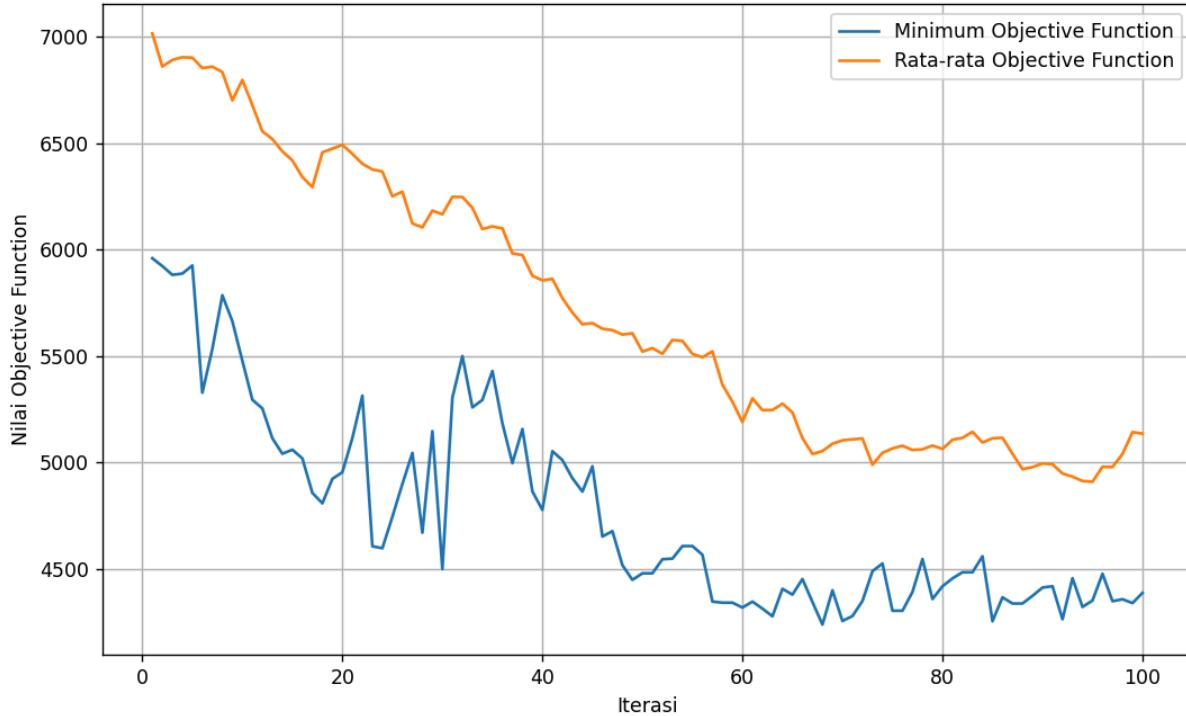
Percobaan 1

State Awal Cube	State Akhir Cube																																																																																																																																																																																																																																													
<p>z</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>50</td><td>32</td><td>30</td><td>119</td><td>121</td><td>6</td><td>61</td><td>102</td><td>115</td><td>2</td></tr> <tr><td>58</td><td>87</td><td>82</td><td>78</td><td>106</td><td>5</td><td>64</td><td>84</td><td>124</td><td></td></tr> <tr><td>92</td><td>113</td><td>12</td><td>67</td><td>100</td><td>9</td><td>15</td><td>39</td><td>128</td><td></td></tr> <tr><td>20</td><td>40</td><td>380</td><td>29</td><td>125</td><td>68</td><td>49</td><td>98</td><td></td><td></td></tr> <tr><td>76</td><td>111</td><td>107</td><td>19</td><td>54</td><td>728</td><td>53</td><td>45</td><td></td><td></td></tr> <tr><td>104</td><td>43</td><td>34</td><td>91</td><td>4435</td><td>728</td><td>89</td><td>185</td><td>66</td><td></td></tr> <tr><td>21</td><td>31</td><td>48</td><td>88</td><td>37</td><td>41</td><td>9</td><td>1</td><td>27</td><td></td></tr> <tr><td>112</td><td>1182</td><td>55</td><td>42</td><td>437</td><td>481</td><td>4</td><td>69</td><td>25</td><td></td></tr> <tr><td>110</td><td>110</td><td>116</td><td>288</td><td>51</td><td>109</td><td>47</td><td></td><td></td><td></td></tr> <tr><td>22</td><td></td><td></td><td>10</td><td>85</td><td>55</td><td>105</td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>26</td><td>120</td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>13</td><td>96</td><td></td><td></td><td></td><td></td><td></td></tr> </table> <p>y</p> <p>x</p>	50	32	30	119	121	6	61	102	115	2	58	87	82	78	106	5	64	84	124		92	113	12	67	100	9	15	39	128		20	40	380	29	125	68	49	98			76	111	107	19	54	728	53	45			104	43	34	91	4435	728	89	185	66		21	31	48	88	37	41	9	1	27		112	1182	55	42	437	481	4	69	25		110	110	116	288	51	109	47				22			10	85	55	105							26	120									13	96						<p>z</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>70</td><td>56</td><td>54</td><td>13</td><td>105</td><td>7</td><td>40</td><td>53</td><td>36</td></tr> <tr><td>23</td><td>109</td><td>3</td><td>98</td><td>41</td><td>21</td><td>100</td><td></td><td></td></tr> <tr><td>82</td><td>84</td><td>125</td><td>193</td><td>76</td><td>119</td><td>22</td><td>65</td><td></td></tr> <tr><td>71</td><td>97</td><td>79</td><td>5527</td><td>75</td><td>1255</td><td></td><td></td><td></td></tr> <tr><td>102</td><td>5</td><td>140</td><td>21</td><td>63</td><td>58</td><td>114</td><td>67</td><td></td></tr> <tr><td>450</td><td>46</td><td>13</td><td>94</td><td>2850</td><td>11892</td><td></td><td></td><td></td></tr> <tr><td>105</td><td>81</td><td>112</td><td>63</td><td>30</td><td>4117</td><td>20</td><td>18</td><td></td></tr> <tr><td>526</td><td>62</td><td>54</td><td>64</td><td>1098</td><td>11715</td><td>64</td><td></td><td></td></tr> <tr><td>82</td><td>104</td><td>112</td><td>62</td><td>752</td><td>578094</td><td>80</td><td></td><td></td></tr> <tr><td>81</td><td>79</td><td>526</td><td>36</td><td>6253</td><td>1355</td><td></td><td></td><td></td></tr> <tr><td>50</td><td>50</td><td>8694</td><td>6111</td><td>2741</td><td>105</td><td></td><td></td><td></td></tr> <tr><td>78</td><td>122</td><td>122</td><td>76</td><td>20</td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>54</td><td></td><td></td><td></td><td></td><td></td></tr> </table> <p>y</p> <p>x</p>	70	56	54	13	105	7	40	53	36	23	109	3	98	41	21	100			82	84	125	193	76	119	22	65		71	97	79	5527	75	1255				102	5	140	21	63	58	114	67		450	46	13	94	2850	11892				105	81	112	63	30	4117	20	18		526	62	54	64	1098	11715	64			82	104	112	62	752	578094	80			81	79	526	36	6253	1355				50	50	8694	6111	2741	105				78	122	122	76	20								54					
50	32	30	119	121	6	61	102	115	2																																																																																																																																																																																																																																					
58	87	82	78	106	5	64	84	124																																																																																																																																																																																																																																						
92	113	12	67	100	9	15	39	128																																																																																																																																																																																																																																						
20	40	380	29	125	68	49	98																																																																																																																																																																																																																																							
76	111	107	19	54	728	53	45																																																																																																																																																																																																																																							
104	43	34	91	4435	728	89	185	66																																																																																																																																																																																																																																						
21	31	48	88	37	41	9	1	27																																																																																																																																																																																																																																						
112	1182	55	42	437	481	4	69	25																																																																																																																																																																																																																																						
110	110	116	288	51	109	47																																																																																																																																																																																																																																								
22			10	85	55	105																																																																																																																																																																																																																																								
			26	120																																																																																																																																																																																																																																										
			13	96																																																																																																																																																																																																																																										
70	56	54	13	105	7	40	53	36																																																																																																																																																																																																																																						
23	109	3	98	41	21	100																																																																																																																																																																																																																																								
82	84	125	193	76	119	22	65																																																																																																																																																																																																																																							
71	97	79	5527	75	1255																																																																																																																																																																																																																																									
102	5	140	21	63	58	114	67																																																																																																																																																																																																																																							
450	46	13	94	2850	11892																																																																																																																																																																																																																																									
105	81	112	63	30	4117	20	18																																																																																																																																																																																																																																							
526	62	54	64	1098	11715	64																																																																																																																																																																																																																																								
82	104	112	62	752	578094	80																																																																																																																																																																																																																																								
81	79	526	36	6253	1355																																																																																																																																																																																																																																									
50	50	8694	6111	2741	105																																																																																																																																																																																																																																									
78	122	122	76	20																																																																																																																																																																																																																																										
			54																																																																																																																																																																																																																																											

Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>----- Sisi ke- 1 -----</p> <pre>46 54 100 97 121 79 35 125 15 61 48 90 45 39 102 18 69 95 71 115 47 25 27 98 2 </pre> <p>----- Sisi ke- 2 -----</p> <pre>80 86 33 70 119 38 36 19 106 6 42 114 28 8 64 51 81 89 49 84 105 83 1 66 124 </pre> <p>----- Sisi ke- 3 -----</p> <pre>63 34 111 82 30 52 99 91 67 24 23 57 14 29 72 85 94 11 53 15 103 109 4 16 123 </pre> <p>----- Sisi ke- 4 -----</p> <pre>122 3 76 113 32 56 78 93 48 87 62 117 77 107 101 10 108 74 44 59 120 65 37 73 68 </pre> <p>----- Sisi ke- 5 -----</p> <pre>112 21 104 92 50 110 118 31 20 58 22 55 41 43 12 13 116 17 9 60 96 26 88 75 7 </pre>	<p>----- Sisi ke- 1 -----</p> <pre>46 79 48 18 47 80 38 42 51 105 63 52 23 85 103 122 56 62 10 120 112 110 22 13 96 </pre> <p>----- Sisi ke- 2 -----</p> <pre>54 35 90 69 25 86 36 114 81 83 34 99 57 94 109 3 78 117 108 65 21 118 55 116 26 </pre> <p>----- Sisi ke- 3 -----</p> <pre>100 125 45 95 27 33 19 28 89 1 111 91 14 11 4 76 93 77 74 37 104 31 41 17 88 </pre> <p>----- Sisi ke- 4 -----</p> <pre>97 5 39 71 98 70 106 8 49 66 82 67 29 53 16 113 48 107 44 73 92 20 43 9 75 </pre> <p>----- Sisi ke- 5 -----</p> <pre>121 61 102 115 2 119 6 64 84 124 30 24 72 15 123 32 87 101 59 68 50 58 12 60 7 </pre>	<p>----- Sisi ke- 1 -----</p> <pre>46 80 63 122 112 54 86 34 3 21 100 33 111 76 104 97 70 82 113 92 121 119 38 32 50 </pre> <p>----- Sisi ke- 2 -----</p> <pre>79 38 52 56 110 35 36 99 78 118 125 19 91 93 31 5 106 67 40 20 61 6 24 87 58 </pre> <p>----- Sisi ke- 3 -----</p> <pre>48 42 23 62 22 90 57 57 117 55 45 28 14 77 41 39 8 29 107 43 102 64 72 101 12 </pre> <p>----- Sisi ke- 4 -----</p> <pre>18 51 85 10 13 69 81 94 108 116 95 89 11 74 17 71 49 53 44 9 115 84 15 59 60 </pre> <p>----- Sisi ke- 5 -----</p> <pre>47 105 103 120 96 25 83 109 65 26 27 1 4 37 88 98 66 16 73 75 2 124 123 68 7 </pre>	<p>----- Sisi ke- 1 -----</p> <pre>64 94 55 19 105 75 117 63 76 7 57 14 92 122 40 13 117 47 55 53 9 80 64 67 36 </pre> <p>----- Sisi ke- 2 -----</p> <pre>83 63 114 103 13 14 97 40 86 98 62 12 50 75 41 61 80 20 114 100 105 55 15 67 65 </pre> <p>----- Sisi ke- 3 -----</p> <pre>76 22 46 84 54 31 102 98 79 3 86 76 30 21 23 27 79 109 118 110 76 11 94 18 125 </pre> <p>----- Sisi ke- 4 -----</p> <pre>76 50 81 82 56 70 43 64 5 109 79 76 100 13 125 122 66 101 4 27 20 41 53 78 58 </pre> <p>----- Sisi ke- 5 -----</p> <pre>82 105 102 71 70 81 52 45 71 23 50 104 63 112 97 78 39 56 54 49 54 37 34 51 28 </pre>	<p>----- Sisi ke- 1 -----</p> <pre>64 75 57 13 9 83 14 62 61 105 76 31 86 27 76 76 70 79 122 20 82 81 50 78 54 </pre> <p>----- Sisi ke- 2 -----</p> <pre>94 117 14 117 80 63 97 12 80 55 22 102 76 79 11 50 43 76 66 41 105 52 104 39 37 </pre> <p>----- Sisi ke- 3 -----</p> <pre>55 63 92 47 64 114 40 50 20 15 46 98 38 109 94 81 64 100 101 53 102 45 63 56 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre>19 76 122 55 67 103 86 75 114 67 84 79 21 118 18 82 5 13 4 78 71 71 112 54 51 </pre> <p>----- Sisi ke- 5 -----</p> <pre>105 7 40 53 36 13 98 41 100 65 54 3 23 110 125 56 109 125 27 58 70 23 97 40 28 </pre>	<p>----- Sisi ke- 1 -----</p> <pre>64 83 76 76 82 94 63 22 50 105 55 114 46 81 102 19 103 84 82 71 105 13 54 56 70 </pre> <p>----- Sisi ke- 2 -----</p> <pre>75 14 31 70 81 117 97 102 43 52 63 40 98 64 45 76 86 79 5 71 7 98 3 109 23 </pre> <p>----- Sisi ke- 3 -----</p> <pre>57 62 86 79 50 14 12 76 76 104 92 50 30 100 63 122 75 21 13 112 40 41 23 125 97 </pre> <p>----- Sisi ke- 4 -----</p> <pre>13 61 27 122 78 117 80 79 66 39 47 20 109 101 56 55 114 118 4 54 53 100 110 27 40 </pre> <p>----- Sisi ke- 5 -----</p> <pre>9 105 76 20 54 80 55 11 41 37 64 15 94 53 34 67 67 18 78 51 36 65 125 58 28 </pre>
Nilai Objective Function Awal: 6985			Nilai Objective Function Akhir: 4386 (Best: 4237)		

Plot Nilai Objective Function Minimum dan Rata-rata:

Plot Nilai Objective Function terhadap Jumlah Iterasi



Nilai Objective Function Terendah dari Populasi Terakhir: 4386

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5134.76

Nilai Objective Function Terbaik dari Seluruh Populasi: 4237

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 4344 ms

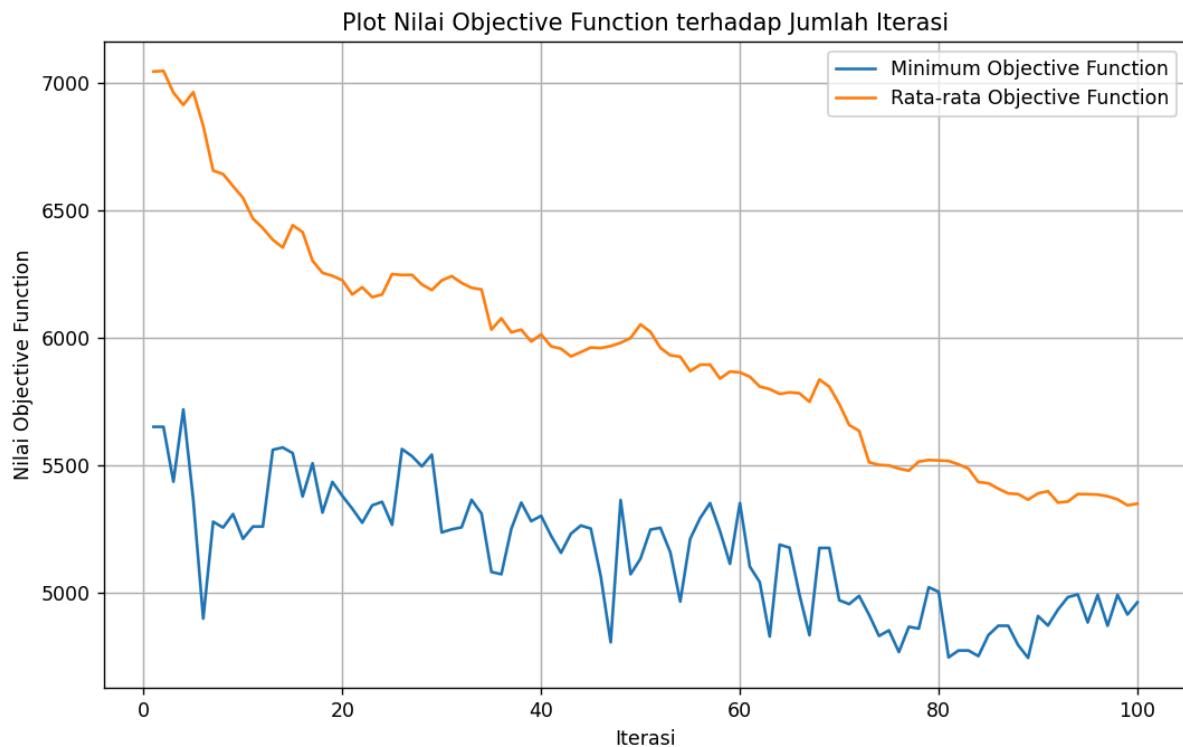
Percobaan 2

State Awal Cube		State Akhir Cube	
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X
----- Sisi ke- 1 ----- 45 21 4 79 124 74 93 59 28 29 105 36 58 119 85 81 116 98 118 3 90 71 15 10 62 ----- Sisi ke- 2 ----- 40 111 30 84 125 102 122 53 82 22 2 54 24 9 64 67 106 78 5 52 13 99 103 39 23 ----- Sisi ke- 3 ----- 60 43 63 57 17 97 76 121 113 26 89 27 12 58 68 19 42 65 72 80 18 117 88 83 11 ----- Sisi ke- 4 ----- 61 95 41 37 20 16 110 48 6 70 33 87 47 101 109 92 56 75 1 104 77 49 108 25 34 ----- Sisi ke- 5 ----- 51 94 14 32 120 114 35 86 123 38 115 46 66 91 55 107 100 112 73 44 7 8 69 96 31	----- Sisi ke- 1 ----- 45 74 105 81 90 40 102 2 67 13 60 97 89 19 18 61 16 33 92 77 51 114 115 107 7 ----- Sisi ke- 2 ----- 21 93 36 116 71 111 122 54 106 99 43 76 27 42 117 95 110 87 56 49 94 35 46 100 8 ----- Sisi ke- 3 ----- 4 59 58 98 15 30 53 24 78 103 63 121 12 65 88 41 48 47 75 108 14 86 66 112 69 ----- Sisi ke- 4 ----- 79 28 119 118 10 84 82 9 5 39 57 113 50 72 83 37 6 101 1 25 32 123 91 73 96 ----- Sisi ke- 5 ----- 124 29 85 3 62 125 22 64 52 23 17 26 68 80 11 20 70 109 104 34 120 38 55 44 31	----- Sisi ke- 1 ----- 45 48 9 70 79 21 111 43 95 94 4 30 63 41 14 79 84 57 37 32 124 125 17 20 120 ----- Sisi ke- 2 ----- 74 102 97 16 114 93 122 76 110 35 59 53 121 48 86 28 82 113 6 123 29 22 26 70 38 ----- Sisi ke- 3 ----- 105 2 89 33 115 36 52 27 87 46 58 24 12 47 66 119 9 50 101 91 85 64 68 109 55 ----- Sisi ke- 4 ----- 81 67 19 92 107 116 106 42 56 100 98 78 65 75 112 118 5 72 1 73 3 52 80 104 44 ----- Sisi ke- 5 ----- 90 13 18 77 7 71 99 117 49 8 15 103 88 108 69 10 39 83 25 96 62 23 11 34 31	----- Sisi ke- 1 ----- 94 53 89 108 23 89 50 61 45 28 108 26 88 61 125 11 116 57 18 64 94 25 116 93 28 ----- Sisi ke- 2 ----- 89 101 59 10 68 50 96 105 84 15 59 53 121 48 86 45 103 84 76 7 28 26 71 55 2 ----- Sisi ke- 3 ----- 48 39 84 69 106 101 96 44 103 26 14 57 39 24 98 48 18 82 95 71 47 84 125 68 15 ----- Sisi ke- 4 ----- 9 59 18 95 11 59 105 61 84 71 66 100 84 63 2 114 1 111 15 68 98 3 7 30 99 ----- Sisi ke- 5 ----- 89 40 80 57 116 84 44 39 82 125 18 61 84 111 7 40 43 18 9 25 16 107 111 45 107 ----- Sisi ke- 6 ----- 108 45 61 18 93 69 103 24 95 68 95 84 63 15 30 116 76 65 95 31 93 7 55 85 87 ----- Sisi ke- 7 ----- 23 28 125 64 28 106 26 98 71 15 11 71 2 68 99 39 55 46 20 111 116 2 57 66 53

Nilai Objective Function Awal: 7308

Nilai Objective Function Akhir: 4962
(Best: 4744)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 4962

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5348.69

Nilai Objective Function Terbaik dari Seluruh Populasi: 4744

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 4545 ms

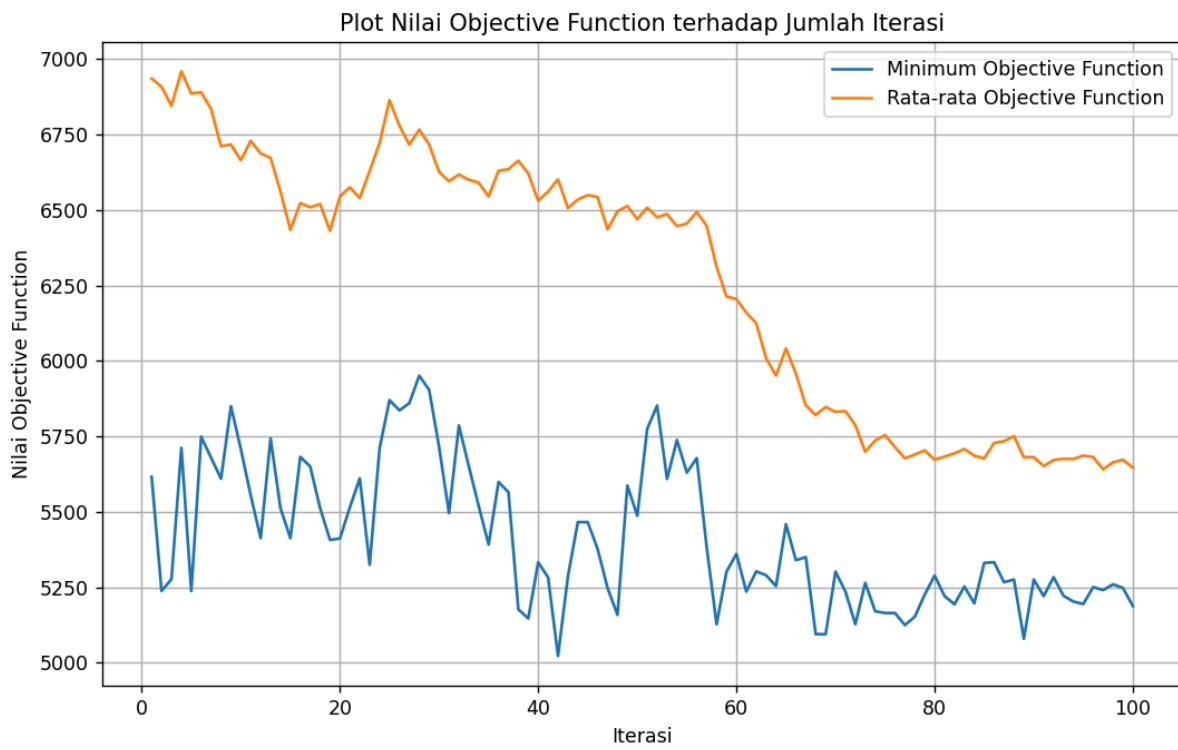
Percobaan 3

State Awal Cube

State Akhir Cube

<p>Sumbu X</p> <p>----- Sisi ke- 1 -----</p> <pre>46 57 47 108 64 120 18 22 38 17 92 103 84 68 67 102 21 106 104 39 66 48 53 117 23 </pre> <p>----- Sisi ke- 2 -----</p> <pre>116 89 107 16 123 12 49 76 118 85 1 69 3 35 37 28 122 121 20 2 93 51 56 125 91 </pre> <p>----- Sisi ke- 3 -----</p> <pre>25 50 77 115 97 114 61 43 95 112 78 36 99 113 124 189 4 63 55 54 31 88 110 13 81 </pre> <p>----- Sisi ke- 4 -----</p> <pre>5 32 27 26 87 75 65 10 59 24 185 86 101 68 80 96 15 8 41 70 71 42 9 58 119 </pre> <p>----- Sisi ke- 5 -----</p> <pre>98 90 40 38 62 74 7 94 34 19 82 73 29 111 14 6 33 52 72 79 83 100 44 45 11 </pre>	<p>Sumbu Y</p> <p>----- Sisi ke- 1 -----</p> <pre>46 120 92 102 66 116 12 1 28 93 25 114 78 109 31 5 75 105 96 71 98 74 82 6 83 </pre> <p>----- Sisi ke- 2 -----</p> <pre>57 18 103 21 48 89 49 69 122 51 58 61 36 4 88 32 65 86 15 42 90 7 73 33 100 </pre> <p>----- Sisi ke- 3 -----</p> <pre>47 22 84 106 53 187 76 5 121 56 77 43 99 63 110 27 10 101 8 9 40 94 29 52 44 </pre> <p>----- Sisi ke- 4 -----</p> <pre>108 30 60 104 117 16 118 35 20 125 115 95 113 55 13 26 59 68 41 58 38 34 111 72 45 </pre> <p>----- Sisi ke- 5 -----</p> <pre>64 17 67 39 23 123 85 37 2 91 97 112 124 54 81 87 24 80 70 119 62 19 14 79 11 </pre>	<p>Sumbu Z</p> <p>----- Sisi ke- 1 -----</p> <pre>46 116 25 5 98 57 89 50 32 98 47 107 77 27 40 108 16 115 26 38 64 123 97 87 62 </pre> <p>----- Sisi ke- 2 -----</p> <pre>120 12 114 75 74 18 49 61 65 7 22 76 43 10 94 30 118 95 59 34 17 85 112 24 19 </pre> <p>----- Sisi ke- 3 -----</p> <pre>92 1 78 105 82 103 69 36 86 73 84 3 99 101 29 68 35 113 68 111 67 37 124 80 14 </pre> <p>----- Sisi ke- 4 -----</p> <pre>102 28 109 96 6 21 122 4 15 33 106 121 63 8 52 104 20 55 41 72 39 2 54 70 79 </pre> <p>----- Sisi ke- 5 -----</p> <pre>66 93 31 71 83 48 51 88 42 100 53 56 110 9 44 117 125 13 58 45 23 91 81 119 11 </pre>	<p>Sumbu X</p> <p>----- Sisi ke- 1 -----</p> <pre>98 46 55 78 26 37 61 53 45 6 112 32 47 24 46 53 98 90 52 18 116 70 49 84 51 </pre> <p>----- Sisi ke- 2 -----</p> <pre>124 114 44 98 37 57 41 94 59 55 52 40 82 12 70 34 15 49 118 87 67 106 76 33 84 </pre> <p>----- Sisi ke- 3 -----</p> <pre>82 95 5 102 52 103 69 36 86 73 84 3 99 101 29 68 35 113 68 111 67 37 124 80 14 </pre> <p>----- Sisi ke- 4 -----</p> <pre>40 81 12 61 91 95 102 22 58 117 98 41 35 13 34 41 24 33 69 66 52 62 119 45 46 </pre> <p>----- Sisi ke- 5 -----</p> <pre>39 29 71 108 106 109 22 115 19 59 55 57 74 50 56 4 24 71 104 45 99 52 101 82 31 </pre>	<p>Sumbu Y</p> <p>----- Sisi ke- 1 -----</p> <pre>98 37 112 53 116 124 57 52 34 67 82 99 15 8 50 40 95 98 41 52 39 109 55 4 99 </pre> <p>----- Sisi ke- 2 -----</p> <pre>46 61 32 98 70 114 41 48 15 106 95 50 94 59 4 81 102 41 24 62 29 22 57 24 52 </pre> <p>----- Sisi ke- 3 -----</p> <pre>55 53 47 90 49 44 94 82 49 76 5 113 72 39 65 12 22 35 33 119 71 115 74 71 101 </pre> <p>----- Sisi ke- 4 -----</p> <pre>78 45 24 52 84 98 59 12 118 33 102 89 75 93 91 61 58 13 69 45 108 19 50 104 82 </pre> <p>----- Sisi ke- 5 -----</p> <pre>26 6 46 18 51 37 55 70 87 84 52 48 51 120 101 91 117 34 66 46 106 59 56 45 31 </pre>	<p>Sumbu Z</p> <p>----- Sisi ke- 1 -----</p> <pre>98 124 82 40 39 46 114 95 81 29 55 44 5 12 71 78 98 102 61 108 26 37 52 91 106 </pre> <p>----- Sisi ke- 2 -----</p> <pre>37 57 99 95 109 61 41 50 102 22 53 94 113 22 115 45 59 89 58 19 6 55 48 117 59 </pre> <p>----- Sisi ke- 3 -----</p> <pre>112 52 15 98 55 32 40 94 41 57 47 82 72 35 74 24 12 75 13 50 46 70 51 34 56 </pre> <p>----- Sisi ke- 4 -----</p> <pre>53 34 8 41 4 98 15 59 24 24 90 49 39 33 71 52 118 93 69 104 18 87 120 66 45 </pre> <p>----- Sisi ke- 5 -----</p> <pre>116 67 50 52 99 70 106 4 62 52 49 76 65 119 101 84 33 91 45 82 51 84 101 46 31 </pre>
Nilai Objective Function Awal: 6155	Nilai Objective Function Akhir: 5188 (Best: 5023)				

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5188

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5646.74

Nilai Objective Function Terbaik dari Seluruh Populasi: 5023

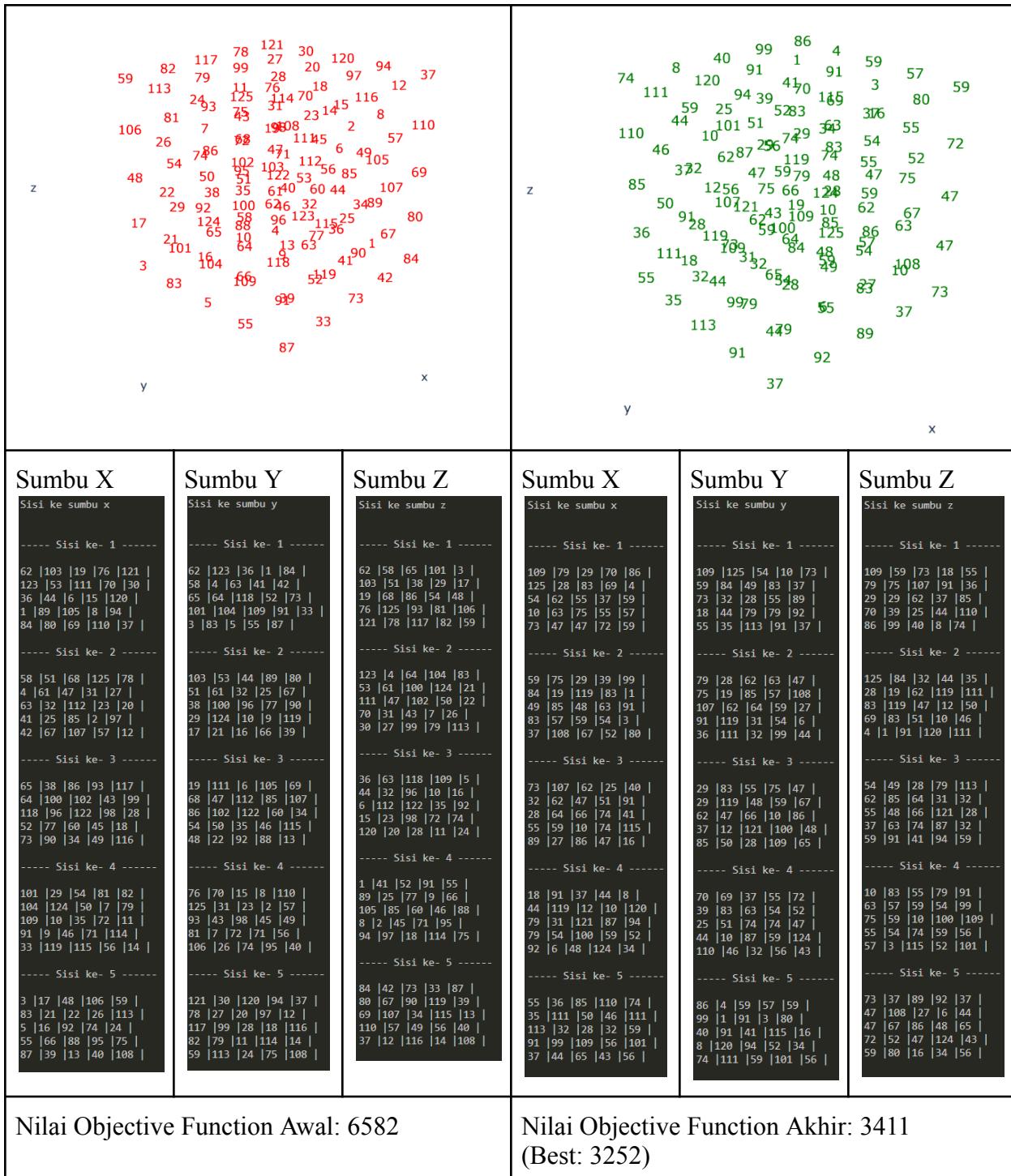
Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 4724 ms

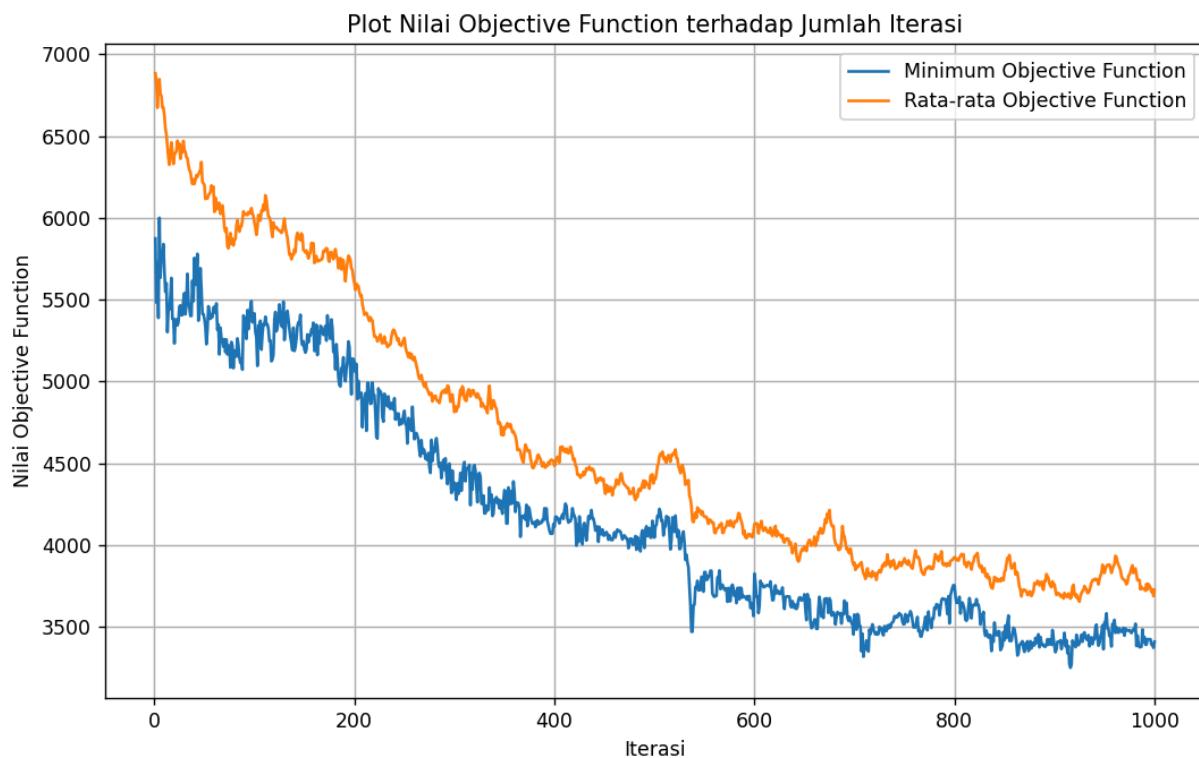
1.3 - Jumlah Maksimum Iterasi 1000

Percobaan 1

State Awal Cube	State Akhir Cube
-----------------	------------------



Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 3411

Nilai Objective Function Rata-rata dari Populasi Terakhir: 3727.21

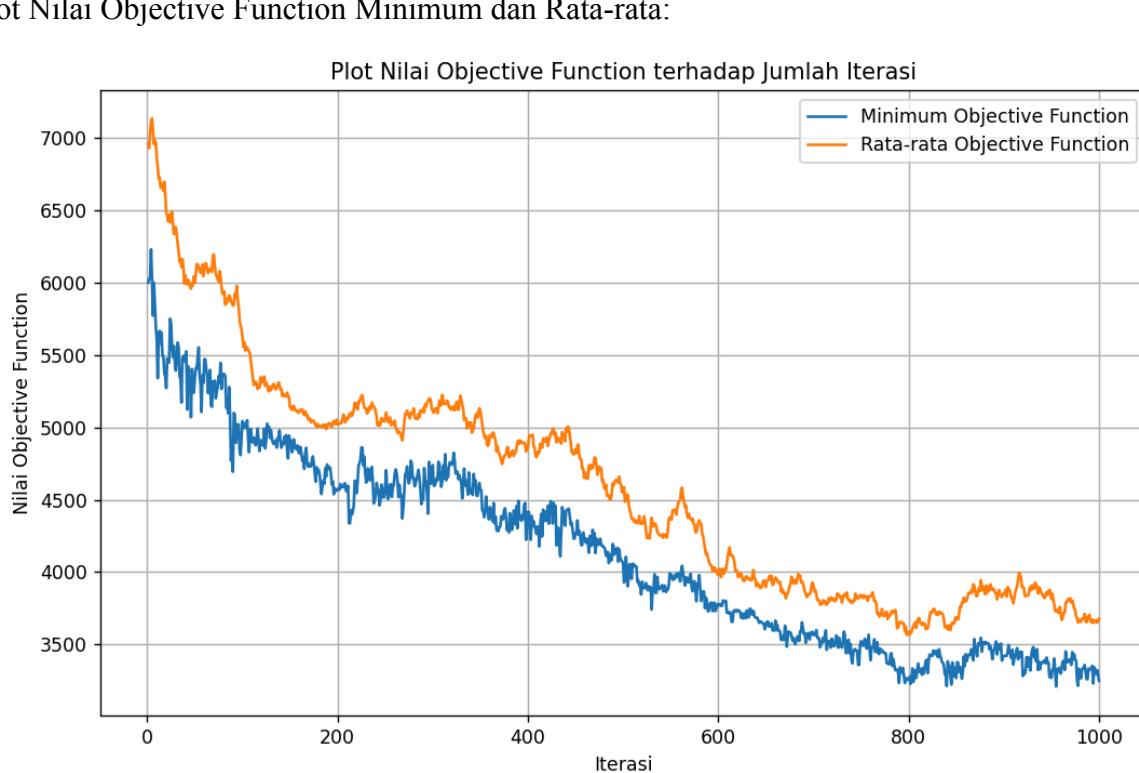
Nilai Objective Function Terbaik dari Seluruh Populasi: 3252

Banyak Iterasi yang Dilakukan: 1000

Durasi Proses Pencarian: 38074 ms

Percobaan 2

Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 ----- 109 68 40 12 85 108 59 110 97 21 70 94 92 6 8 30 25 2 90 87 48 91 96 115 56 ----- Sisi ke- 2 ----- 105 95 118 82 112 66 63 10 55 45 11 14 64 84 76 116 7 44 78 41 88 17 16 38 71 ----- Sisi ke- 3 ----- 37 27 113 22 31 98 99 194 79 100 120 106 53 34 29 1 58 73 49 42 47 3 50 102 43 ----- Sisi ke- 4 ----- 77 61 93 32 83 74 33 15 46 23 67 9 26 35 103 101 20 111 125 114 86 69 39 13 52 ----- Sisi ke- 5 ----- 4 15 28 123 121 107 88 81 117 65 122 19 18 89 119 54 57 24 124 72 36 51 62 75 60	----- Sisi ke- 1 ----- 109 108 70 30 48 105 66 11 116 180 37 98 120 1 147 77 74 67 101 86 4 107 122 54 36 ----- Sisi ke- 2 ----- 68 59 94 25 91 95 63 14 7 17 27 99 106 58 3 61 33 9 20 69 15 88 19 57 51 ----- Sisi ke- 3 ----- 40 110 92 2 96 118 10 64 44 16 113 104 53 73 50 93 5 26 111 39 28 81 18 24 62 ----- Sisi ke- 4 ----- 12 97 6 90 115 82 55 84 78 38 22 79 34 49 102 32 46 35 125 13 123 117 89 124 75 ----- Sisi ke- 5 ----- 85 21 8 87 56 112 45 76 41 71 31 100 29 42 43 83 23 103 114 52 121 65 119 72 60	----- Sisi ke- 1 ----- 109 105 37 77 4 68 95 27 61 15 40 118 113 93 28 12 82 22 32 123 85 112 31 83 121 ----- Sisi ke- 2 ----- 49 80 65 65 39 59 63 99 33 88 110 10 104 5 81 97 55 79 46 117 21 45 100 23 65 ----- Sisi ke- 3 ----- 70 11 120 67 122 94 14 106 9 19 92 64 53 26 18 6 84 34 35 89 8 76 29 103 119 ----- Sisi ke- 4 ----- 46 111 81 75 34 56 18 68 120 30 56 125 33 44 46 110 82 64 84 57 65 25 83 65 88 ----- Sisi ke- 5 ----- 90 32 125 11 62 49 116 65 80 46 72 59 21 43 108 58 56 46 112 81 89 114 20 73 30 ----- Sisi ke- 6 ----- 37 10 24 48 116 76 112 68 5 96 117 1 90 93 53 43 104 67 69 54 85 66 71 114 1 ----- Sisi ke- 7 ----- 30 54 64 27 65 49 104 25 68 120 46 56 56 110 65 90 49 72 58 89 37 76 117 43 85 ----- Sisi ke- 8 ----- 38 49 46 90 37 46 80 111 32 10 64 65 81 125 24 71 65 75 11 48 44 39 34 62 116 ----- Sisi ke- 9 ----- 54 104 56 49 76 18 26 18 116 112 63 52 68 65 68 67 3 120 80 5 37 102 30 46 96 ----- Sisi ke- 10 ----- 64 25 56 72 117 39 81 125 59 1 101 67 33 21 90 76 116 44 43 93 77 32 46 108 53 ----- Sisi ke- 11 ----- 27 68 110 58 43 112 2 82 56 104 55 62 64 46 67 64 39 84 112 69 76 59 57 81 54 ----- Sisi ke- 12 ----- 65 120 65 89 85 56 101 25 114 66 117 24 83 20 71 77 6 65 73 114 64 103 88 30 1	Sisi ke sumbu x ----- Sisi ke- 1 ----- 30 46 64 71 44 54 18 63 67 37 64 39 101 76 77 27 112 55 64 76 65 56 117 77 64 ----- Sisi ke- 2 ----- 46 18 39 112 56 80 26 81 2 101 111 18 125 82 25 32 116 59 56 114 10 112 1 104 66 ----- Sisi ke- 3 ----- 64 63 101 55 117 55 52 67 62 24 81 68 33 64 83 125 65 21 46 20 24 68 90 67 71 ----- Sisi ke- 4 ----- 71 67 76 64 77 65 3 116 39 6 75 120 44 84 65 11 80 43 112 73 48 5 93 69 114 ----- Sisi ke- 5 ----- 44 37 77 76 64 39 102 32 59 103 34 30 46 57 88 62 46 108 81 30 116 96 53 54 1 ----- Sisi ke- 6 ----- 37 10 24 48 116 76 112 68 5 96 117 1 90 93 53 43 104 67 69 54 85 66 71 114 1 ----- Sisi ke- 7 ----- 30 54 64 27 65 49 104 25 68 120 46 56 56 110 65 90 49 72 58 89 37 76 117 43 85 ----- Sisi ke- 8 ----- 38 49 46 90 37 46 80 111 32 10 64 65 81 125 24 71 65 75 11 48 44 39 34 62 116 ----- Sisi ke- 9 ----- 54 104 56 49 76 18 26 18 116 112 63 52 68 65 68 67 3 120 80 5 37 102 30 46 96 ----- Sisi ke- 10 ----- 64 25 56 72 117 39 81 125 59 1 101 67 33 21 90 76 116 44 43 93 77 32 46 108 53 ----- Sisi ke- 11 ----- 27 68 110 58 43 112 2 82 56 104 55 62 64 46 67 64 39 84 112 69 76 59 57 81 54 ----- Sisi ke- 12 ----- 65 120 65 89 85 56 101 25 114 66 117 24 83 20 71 77 6 65 73 114 64 103 88 30 1	Nilai Objective Function Awal: 6756 Nilai Objective Function Akhir: 3249 (Best: 3209)	Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 3249

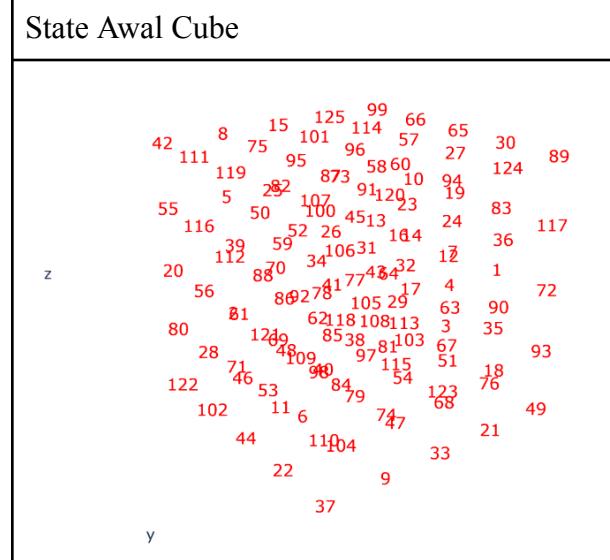
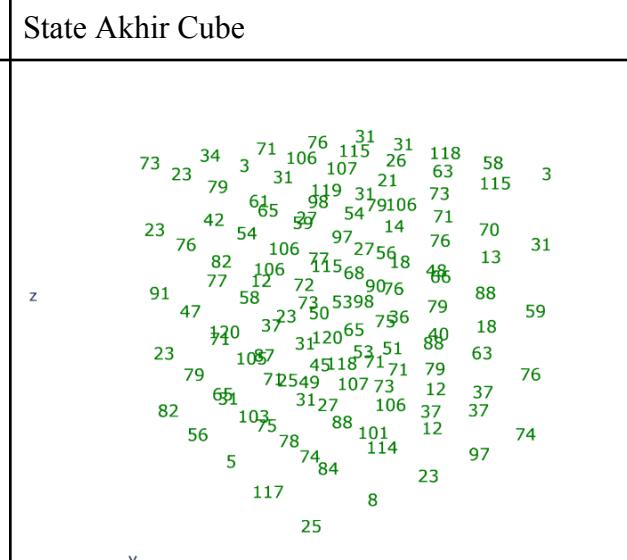
Nilai Objective Function Rata-rata dari Populasi Terakhir: 3676.13

Nilai Objective Function Terbaik dari Seluruh Populasi: 3209

Banyak Iterasi yang Dilakukan: 1000

Durasi Proses Pencarian: 37709 ms

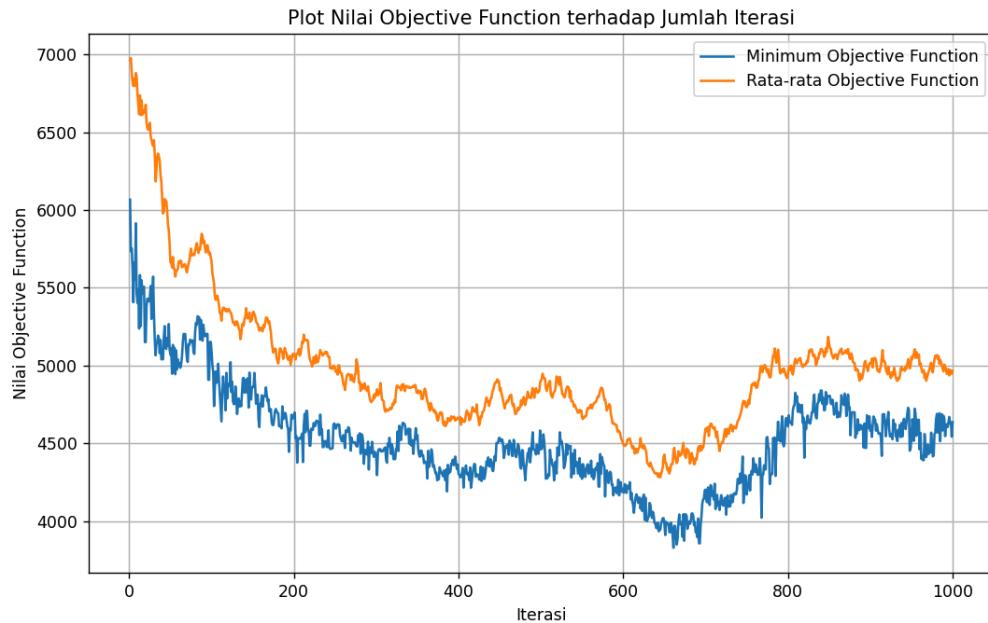
Percobaan 3

State Awal Cube	State Akhir Cube				
					
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>108 43 13 58 99 103 17 14 10 66 51 63 7 19 65 76 35 1 83 30 49 93 72 117 89 </pre> <p>----- Sisi ke- 2 -----</p> <pre>85 41 26 87 125 97 105 31 91 114 54 113 32 23 57 68 67 4 24 27 21 18 90 36 124 </pre> <p>----- Sisi ke- 3 -----</p> <pre>48 86 59 82 15 98 62 34 107 101 79 38 77 45 96 47 115 29 16 68 33 123 3 12 94 </pre> <p>----- Sisi ke- 4 -----</p> <pre>71 2 112 5 8 53 121 88 58 75 6 109 92 52 95 104 84 118 106 73 9 74 81 64 120 </pre> <p>----- Sisi ke- 5 -----</p> <pre>122 80 20 55 42 102 28 56 116 111 44 46 61 39 119 22 11 69 70 25 37 110 40 78 100 </pre> <p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>108 103 51 76 49 85 97 54 68 21 48 98 79 47 33 71 53 6 104 9 122 102 44 22 37 </pre> <p>----- Sisi ke- 2 -----</p> <pre>43 17 63 35 93 41 105 113 67 18 86 62 38 115 123 2 121 109 84 74 80 28 46 11 110 </pre> <p>----- Sisi ke- 3 -----</p> <pre>13 14 7 1 72 26 31 32 4 90 59 34 77 29 3 112 88 92 118 81 20 56 61 69 40 </pre> <p>----- Sisi ke- 4 -----</p> <pre>58 10 19 83 117 87 91 23 24 36 82 107 45 16 12 5 58 52 106 64 55 116 39 70 78 </pre> <p>----- Sisi ke- 5 -----</p> <pre>49 21 33 9 37 93 18 123 74 110 72 90 3 81 40 117 36 12 64 78 42 111 119 25 100 </pre> <p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>108 85 48 71 122 43 41 86 2 80 13 26 59 112 20 58 87 82 5 55 99 125 15 8 42 </pre> <p>----- Sisi ke- 2 -----</p> <pre>103 97 98 53 102 17 105 62 121 28 14 31 34 88 56 10 91 107 50 116 66 114 101 75 111 </pre> <p>----- Sisi ke- 3 -----</p> <pre>51 54 79 6 44 63 113 38 109 46 7 32 177 92 61 88 118 53 97 107 114 73 75 56 21 65 57 96 95 119 </pre> <p>----- Sisi ke- 4 -----</p> <pre>76 68 47 104 22 35 67 115 84 11 1 4 29 118 69 83 24 16 106 70 30 27 60 73 25 </pre> <p>----- Sisi ke- 5 -----</p> <pre>49 21 33 9 37 93 18 123 74 110 72 90 3 81 40 117 36 12 64 78 42 111 119 25 100 </pre> <p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>53 98 27 31 31 71 36 18 106 31 12 40 66 71 118 37 63 88 70 58 74 76 59 31 3 </pre> <p>----- Sisi ke- 2 -----</p> <pre>45 50 77 98 76 17 65 68 54 115 106 51 76 14 26 12 79 79 76 63 97 37 18 13 115 </pre> <p>----- Sisi ke- 3 -----</p> <pre>71 37 106 65 71 63 113 38 109 46 7 32 177 92 61 88 118 53 97 107 114 73 75 56 21 23 37 88 48 73 </pre> <p>----- Sisi ke- 4 -----</p> <pre>65 71 77 42 34 103 105 58 54 3 1 4 29 118 69 83 24 16 106 70 8 101 71 90 79 </pre> <p>----- Sisi ke- 5 -----</p> <pre>31 106 71 70 31 98 54 14 76 13 65 59 97 56 48 42 54 106 115 90 23 76 82 12 73 </pre> <p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>53 71 12 37 74 45 107 106 12 97 71 31 88 114 23 65 103 78 84 8 82 56 5 117 25 </pre> <p>----- Sisi ke- 2 -----</p> <pre>98 36 40 63 76 50 65 51 79 37 37 31 118 73 37 71 105 25 27 101 23 79 31 75 74 </pre> <p>----- Sisi ke- 3 -----</p> <pre>71 107 31 103 56 36 65 31 105 79 18 68 72 58 47 106 54 59 54 76 31 115 106 3 23 </pre> <p>----- Sisi ke- 4 -----</p> <pre>12 186 88 78 5 40 51 118 25 31 66 76 53 23 120 71 14 97 106 82 118 26 107 31 79 </pre> <p>----- Sisi ke- 5 -----</p> <pre>37 12 114 84 117 63 79 73 27 75 88 79 75 120 87 70 76 56 115 12 58 63 21 119 61 </pre> <p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>53 45 71 65 82 98 58 37 71 23 27 77 106 77 91 31 98 65 42 23 31 76 71 34 73 </pre> <p>----- Sisi ke- 2 -----</p> <pre>71 107 31 103 56 36 65 31 105 79 18 68 72 58 47 106 54 59 54 76 31 115 106 3 23 </pre> <p>----- Sisi ke- 3 -----</p> <pre>12 186 88 78 5 40 51 118 25 31 66 76 53 23 120 71 14 97 106 82 118 26 107 31 79 </pre> <p>----- Sisi ke- 4 -----</p> <pre>37 12 114 84 117 63 79 73 27 75 88 79 75 120 87 70 76 56 115 12 58 63 21 119 61 </pre> <p>----- Sisi ke- 5 -----</p> <pre>74 97 23 8 25 76 37 37 101 74 59 18 88 71 49 31 13 48 99 73 3 115 73 79 27 </pre>					

Nilai Objective Function Awal: 7826

Nilai Objective Function Akhir: 4637
(Best: 3829)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 4637

Nilai Objective Function Rata-rata dari Populasi Terakhir: 4965.78

Nilai Objective Function Terbaik dari Seluruh Populasi: 3829

Banyak Iterasi yang Dilakukan: 1000

Durasi Proses Pencarian: 38152 ms

2. Variasi Kontrol Jumlah Iterasi:

Variasi ini akan menetapkan jumlah maksimum iterasi sebesar 100 sebagai kontrol. Lalu, akan diberikan jumlah populasi menjadi 3 jenis, yakni sebesar 10, 100, dan 1000. Berikut adalah hasil percobaan menggunakan metode Genetic Algorithm dengan variansi jumlah populasi sebagai kontrol yang ditetapkan:

2.1 - Jumlah Populasi 10

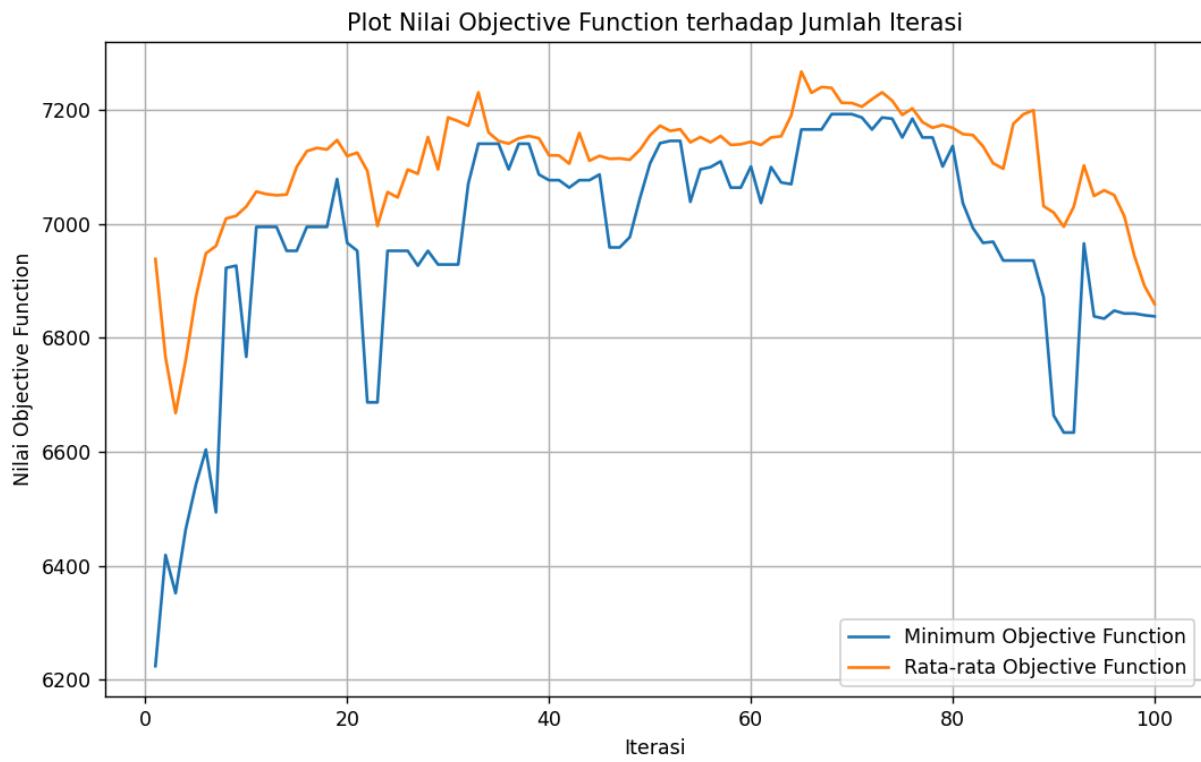
Percobaan 1

State Awal Cube

State Akhir Cube

<table border="1"> <thead> <tr> <th colspan="2">Sumbu X</th> <th colspan="2">Sumbu Y</th> <th colspan="2">Sumbu Z</th> </tr> </thead> <tbody> <tr> <td colspan="2"><i>Sisi ke sumbu x</i></td> <td colspan="2"><i>Sisi ke sumbu y</i></td> <td colspan="2"><i>Sisi ke sumbu z</i></td> </tr> <tr> <td colspan="6"> <p>----- Sisi ke- 1 -----</p> <pre>3 21 88 6 97 11 89 61 122 64 48 94 62 82 8 43 107 56 47 123 113 12 124 90 108 </pre> <p>----- Sisi ke- 2 -----</p> <pre>31 58 77 41 98 81 45 15 18 25 65 16 19 1 92 71 13 24 84 120 20 109 72 63 121 </pre> <p>----- Sisi ke- 3 -----</p> <pre>112 95 111 87 96 86 101 125 83 66 102 60 29 73 74 14 91 46 10 37 27 93 50 69 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre>67 51 85 32 53 49 78 79 22 55 33 54 4 76 36 104 105 42 57 118 35 119 23 75 110 </pre> <p>----- Sisi ke- 5 -----</p> <pre>117 39 80 26 2 5 17 70 116 99 44 103 38 114 68 59 30 52 115 40 106 9 7 28 100 </pre> </td></tr> <tr> <td colspan="2"><i>Sisi ke sumbu x</i></td> <td colspan="2"><i>Sisi ke sumbu y</i></td> <td colspan="2"><i>Sisi ke sumbu z</i></td> </tr> </tbody> </table>	Sumbu X		Sumbu Y		Sumbu Z		<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>		<p>----- Sisi ke- 1 -----</p> <pre>3 21 88 6 97 11 89 61 122 64 48 94 62 82 8 43 107 56 47 123 113 12 124 90 108 </pre> <p>----- Sisi ke- 2 -----</p> <pre>31 58 77 41 98 81 45 15 18 25 65 16 19 1 92 71 13 24 84 120 20 109 72 63 121 </pre> <p>----- Sisi ke- 3 -----</p> <pre>112 95 111 87 96 86 101 125 83 66 102 60 29 73 74 14 91 46 10 37 27 93 50 69 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre>67 51 85 32 53 49 78 79 22 55 33 54 4 76 36 104 105 42 57 118 35 119 23 75 110 </pre> <p>----- Sisi ke- 5 -----</p> <pre>117 39 80 26 2 5 17 70 116 99 44 103 38 114 68 59 30 52 115 40 106 9 7 28 100 </pre>						<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>		<table border="1"> <thead> <tr> <th colspan="2">Sumbu X</th> <th colspan="2">Sumbu Y</th> <th colspan="2">Sumbu Z</th> </tr> </thead> <tbody> <tr> <td colspan="2"><i>Sisi ke sumbu x</i></td> <td colspan="2"><i>Sisi ke sumbu y</i></td> <td colspan="2"><i>Sisi ke sumbu z</i></td> </tr> <tr> <td colspan="6"> <p>----- Sisi ke- 1 -----</p> <pre>3 31 112 67 117 21 58 95 51 39 112 86 102 14 27 67 49 33 104 35 117 5 44 59 106 </pre> <p>----- Sisi ke- 2 -----</p> <pre>11 81 86 49 5 58 45 16 13 109 95 101 60 91 93 51 78 54 105 119 39 17 103 30 9 </pre> <p>----- Sisi ke- 3 -----</p> <pre>88 61 62 56 124 77 15 19 24 72 111 125 29 46 59 85 79 4 42 23 80 78 38 52 7 </pre> <p>----- Sisi ke- 4 -----</p> <pre>6 122 82 47 99 41 18 1 84 63 87 83 73 18 69 32 22 76 57 75 26 116 114 115 28 </pre> <p>----- Sisi ke- 5 -----</p> <pre>43 71 14 104 59 107 13 91 105 30 56 24 46 42 52 47 84 10 57 115 123 120 37 118 40 </pre> </td></tr> <tr> <td colspan="2"><i>Sisi ke sumbu x</i></td> <td colspan="2"><i>Sisi ke sumbu y</i></td> <td colspan="2"><i>Sisi ke sumbu z</i></td> </tr> </tbody> </table>	Sumbu X		Sumbu Y		Sumbu Z		<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>		<p>----- Sisi ke- 1 -----</p> <pre>3 31 112 67 117 21 58 95 51 39 112 86 102 14 27 67 49 33 104 35 117 5 44 59 106 </pre> <p>----- Sisi ke- 2 -----</p> <pre>11 81 86 49 5 58 45 16 13 109 95 101 60 91 93 51 78 54 105 119 39 17 103 30 9 </pre> <p>----- Sisi ke- 3 -----</p> <pre>88 61 62 56 124 77 15 19 24 72 111 125 29 46 59 85 79 4 42 23 80 78 38 52 7 </pre> <p>----- Sisi ke- 4 -----</p> <pre>6 122 82 47 99 41 18 1 84 63 87 83 73 18 69 32 22 76 57 75 26 116 114 115 28 </pre> <p>----- Sisi ke- 5 -----</p> <pre>43 71 14 104 59 107 13 91 105 30 56 24 46 42 52 47 84 10 57 115 123 120 37 118 40 </pre>						<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>	
Sumbu X		Sumbu Y		Sumbu Z																																													
<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>																																													
<p>----- Sisi ke- 1 -----</p> <pre>3 21 88 6 97 11 89 61 122 64 48 94 62 82 8 43 107 56 47 123 113 12 124 90 108 </pre> <p>----- Sisi ke- 2 -----</p> <pre>31 58 77 41 98 81 45 15 18 25 65 16 19 1 92 71 13 24 84 120 20 109 72 63 121 </pre> <p>----- Sisi ke- 3 -----</p> <pre>112 95 111 87 96 86 101 125 83 66 102 60 29 73 74 14 91 46 10 37 27 93 50 69 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre>67 51 85 32 53 49 78 79 22 55 33 54 4 76 36 104 105 42 57 118 35 119 23 75 110 </pre> <p>----- Sisi ke- 5 -----</p> <pre>117 39 80 26 2 5 17 70 116 99 44 103 38 114 68 59 30 52 115 40 106 9 7 28 100 </pre>																																																	
<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>																																													
Sumbu X		Sumbu Y		Sumbu Z																																													
<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>																																													
<p>----- Sisi ke- 1 -----</p> <pre>3 31 112 67 117 21 58 95 51 39 112 86 102 14 27 67 49 33 104 35 117 5 44 59 106 </pre> <p>----- Sisi ke- 2 -----</p> <pre>11 81 86 49 5 58 45 16 13 109 95 101 60 91 93 51 78 54 105 119 39 17 103 30 9 </pre> <p>----- Sisi ke- 3 -----</p> <pre>88 61 62 56 124 77 15 19 24 72 111 125 29 46 59 85 79 4 42 23 80 78 38 52 7 </pre> <p>----- Sisi ke- 4 -----</p> <pre>6 122 82 47 99 41 18 1 84 63 87 83 73 18 69 32 22 76 57 75 26 116 114 115 28 </pre> <p>----- Sisi ke- 5 -----</p> <pre>43 71 14 104 59 107 13 91 105 30 56 24 46 42 52 47 84 10 57 115 123 120 37 118 40 </pre>																																																	
<i>Sisi ke sumbu x</i>		<i>Sisi ke sumbu y</i>		<i>Sisi ke sumbu z</i>																																													
Nilai Objective Function Awal: 7418	Nilai Objective Function Akhir: 6838 (Best: 6224)																																																

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 6838

Nilai Objective Function Rata-rata dari Populasi Terakhir: 6859.6

Nilai Objective Function Terbaik dari Seluruh Populasi: 6224

Banyak Iterasi yang Dilakukan: 100

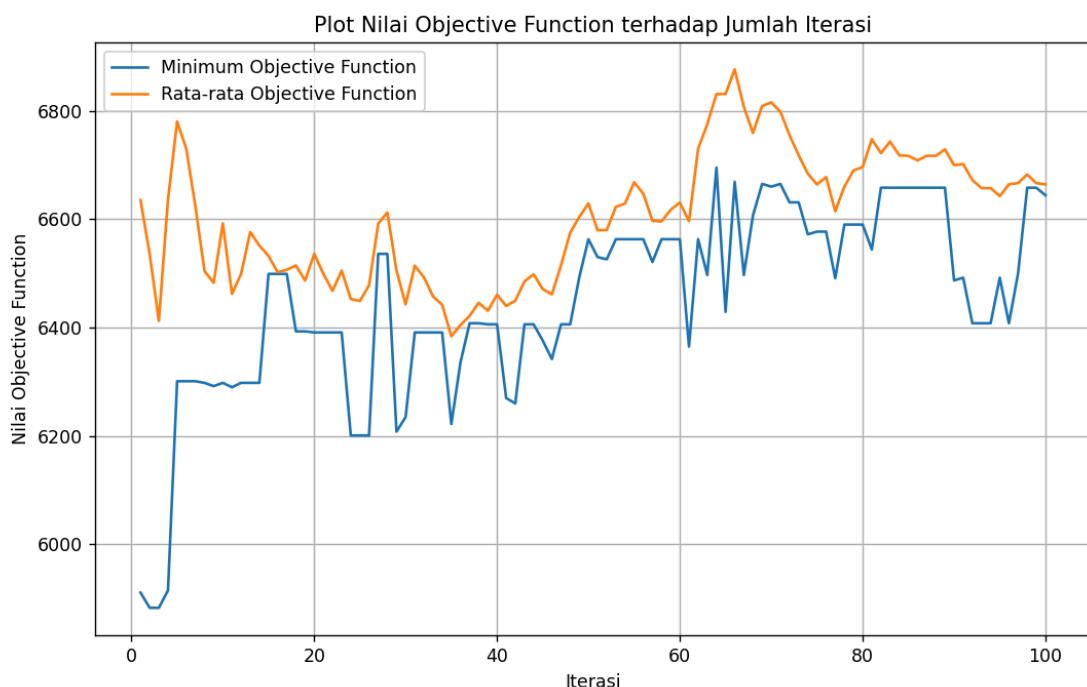
Durasi Proses Pencarian: 407 ms

Percobaan 2

State Awal Cube												State Akhir Cube													
40	6	117	86	112	115	59	12	16	108	116	70	4	55	55	6	42	28	101	69	23	21				
100	97	4	107	50	113	10	13	110	130	121	27	101	93	98	97	20	83	89	85	87					
18	101	61	33	49	113	10	14	78	75	123	26	84	88	509	4450	53	11	38	62	66	32				
3	82	24	10	104	104	78	36	62	3	28	45	90	90	41	34	58	310	494	84	16					
14	154	995	59	22	73	72	109	66	560	8751	68	43	56	115	120	2571	494	624	31	5					
125	58	39	45	103	83	20	69	32	80	81	25	95	27	1036	45	184	246	100	113	112	14				
52	64	23	105	83	20	69	33	109	109	44	88	77	74	144	8	12510	57	86	103						
47	29	18	105	893	31	17	19	89	85				107	309	81	35	626	17	95						
21													58	121	59	18	60	113	19	75					
19													61	80	111	94	102	114	72	47					
													13	1186	73	76	39	92							

Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 ----- 2 120 7 13 115 11 51 63 30 59 25 68 26 121 108 88 95 45 27 116 85 77 9 119 70 ----- Sisi ke- 2 ----- 55 105 104 1 112 89 72 36 110 12 79 57 76 75 16 74 71 90 123 5 89 22 43 84 94 ----- Sisi ke- 3 ----- 35 56 24 107 86 83 66 48 49 124 53 32 37 78 46 67 102 109 62 111 31 44 96 28 91 ----- Sisi ke- 4 ----- 39 15 82 97 117 23 54 99 61 114 118 103 122 10 50 93 69 60 98 113 17 106 81 87 92 ----- Sisi ke- 5 ----- 125 14 18 100 40 52 58 3 101 6 47 64 34 38 4 21 29 42 65 33 19 8 20 73 41	----- Sisi ke- 1 ----- 2 11 25 88 85 55 80 79 74 89 35 83 53 67 31 39 23 118 93 17 125 52 47 21 19 ----- Sisi ke- 2 ----- 120 51 68 95 77 105 72 57 71 22 56 66 32 102 44 15 54 103 69 106 14 58 64 29 8 ----- Sisi ke- 3 ----- 7 63 26 45 9 104 36 76 90 43 24 48 37 109 96 82 99 122 68 81 18 3 34 42 20 ----- Sisi ke- 4 ----- 13 30 121 27 119 1 110 75 123 84 107 49 78 62 28 97 61 10 98 87 100 101 38 65 73 ----- Sisi ke- 5 ----- 115 59 108 116 70 112 12 16 5 94 86 124 46 111 91 117 114 50 113 92 40 6 4 33 41	----- Sisi ke- 1 ----- 2 55 35 39 125 120 105 56 15 14 7 104 24 82 18 13 1 107 97 100 115 112 86 117 40 ----- Sisi ke- 2 ----- 11 80 83 23 52 51 72 66 54 58 63 36 48 99 3 30 110 49 61 101 59 12 124 114 6 ----- Sisi ke- 3 ----- 59 185 115 51 55 60 8 120 54 97 114 35 118 3 20 26 76 37 122 34 121 75 78 10 38 108 16 46 50 4 ----- Sisi ke- 4 ----- 121 14 12 88 55 80 30 71 41 98 94 18 36 34 7 45 90 109 60 42 27 123 62 98 65 116 5 111 113 33 ----- Sisi ke- 5 ----- 58 74 52 15 4 61 107 27 90 104 13 2 24 56 93 77 22 44 106 8 9 43 96 81 20 119 84 28 87 73 70 94 91 92 41	----- Sisi ke- 1 ----- 10 64 110 53 42 26 82 114 38 28 95 22 67 62 101 63 103 31 66 69 10 14 5 32 21 ----- Sisi ke- 2 ----- 81 1 58 44 6 19 77 71 29 83 72 57 46 116 89 40 17 100 84 85 92 112 113 16 23 ----- Sisi ke- 3 ----- 110 114 67 31 5 58 71 46 100 113 115 120 118 122 86 12 71 36 125 61 52 27 24 9 113 ----- Sisi ke- 4 ----- 53 38 62 66 32 44 29 116 84 16 51 54 3 49 124 88 41 34 25 22 15 90 56 48 45 ----- Sisi ke- 5 ----- 42 28 101 69 21 6 83 89 85 23 55 97 20 11 87 55 98 7 58 81 4 104 93 109 108	----- Sisi ke- 1 ----- 10 81 59 121 58 64 1 105 14 174 110 58 115 12 52 53 44 51 88 15 42 6 55 55 4 ----- Sisi ke- 2 ----- 26 19 60 80 61 82 77 8 30 107 114 71 120 71 127 38 29 54 41 190 28 83 97 98 104 ----- Sisi ke- 3 ----- 95 72 114 94 13 22 57 35 18 2 67 46 118 36 24 62 116 3 34 56 101 89 20 7 93 ----- Sisi ke- 4 ----- 63 40 76 106 37 103 17 75 102 111 31 100 122 125 9 66 84 49 25 48 69 85 11 50 109 ----- Sisi ke- 5 ----- 10 92 39 96 79 14 112 47 73 119 5 113 86 61 113 32 16 124 22 45 21 23 87 81 108	
Nilai Objective Function Awal: 7176				Nilai Objective Function Akhir: 6644 (best 5883)	

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 6644

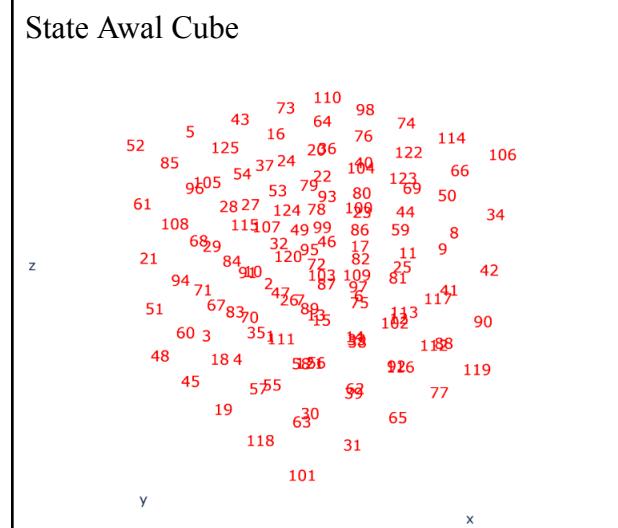
Nilai Objective Function Rata-rata dari Populasi Terakhir: 6664.2

Nilai Objective Function Terbaik dari Seluruh Populasi: 5883

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 402 ms

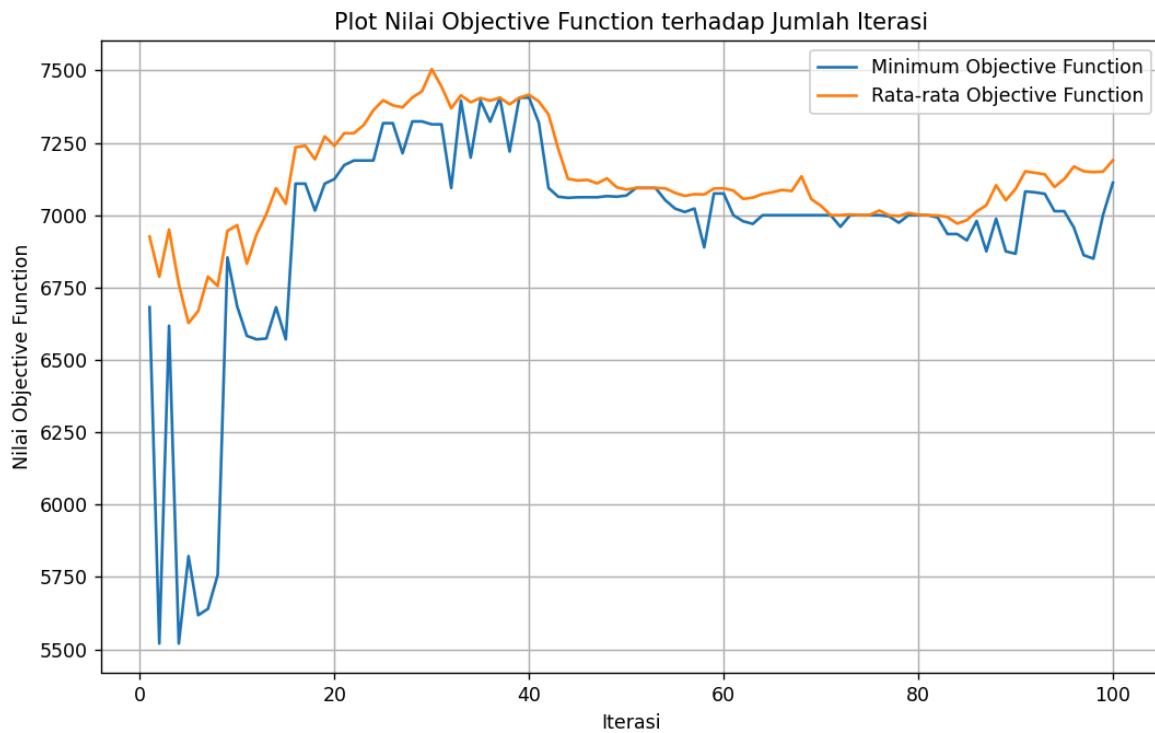
Percobaan 3

State Awal Cube		State Akhir Cube			
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
 <p>Sisi ke sumbu x</p> <pre> ----- Sisi ke- 1 ----- 87 46 93 36 110 75 82 23 40 98 102 81 59 123 74 112 117 9 50 114 119 90 42 34 106 ----- Sisi ke- 2 ----- 26 120 124 24 73 15 103 99 22 64 38 6 17 80 76 92 12 25 44 122 77 88 41 8 66 ----- Sisi ke- 3 ----- 70 91 115 54 43 111 47 32 53 16 56 13 72 78 20 62 33 97 86 104 65 116 113 11 69 ----- Sisi ke- 4 ----- 3 71 68 96 5 4 83 84 28 125 55 1 2 107 37 36 121 89 95 79 31 39 14 109 100 ----- Sisi ke- 5 ----- 48 51 21 61 52 45 60 94 108 85 19 18 67 29 105 118 57 35 10 27 101 63 58 7 49 </pre> <p>Sisi ke sumbu y</p> <pre> ----- Sisi ke- 1 ----- 87 75 102 112 119 26 15 38 92 77 70 111 56 62 65 3 4 55 30 31 48 45 19 118 101 ----- Sisi ke- 2 ----- 46 82 81 117 90 120 103 6 12 88 91 47 13 33 116 71 83 1 121 39 51 68 18 57 63 ----- Sisi ke- 3 ----- 93 23 59 9 42 124 99 17 25 41 115 32 72 97 113 68 84 2 89 14 21 94 67 35 58 ----- Sisi ke- 4 ----- 36 49 123 50 34 24 22 88 44 8 54 53 78 86 11 96 28 107 95 109 61 108 29 10 7 ----- Sisi ke- 5 ----- 110 98 74 114 106 73 64 76 122 66 43 16 20 104 69 5 125 37 79 100 52 85 105 27 49 </pre> <p>Sisi ke sumbu z</p> <pre> ----- Sisi ke- 1 ----- 52 5 43 73 110 98 85 125 16 64 74 74 61 96 54 37 24 104 108 28 27 124 78 109 21 68 9 115 07 44 51 94 71 67 83 70 60 3 35 111 113 48 18 4 55 58 56 45 19 63 30 65 118 101 101 101 101 ----- Sisi ke- 2 ----- 125 107 2 63 82 34 52 81 14 50 123 19 32 67 17 65 15 70 74 82 15 124 125 73 2 43 83 98 908 23 25 157 925 68 6 20 50 590 33 60 51 60 11 115 145 26 12 100 36 64 58 8 30 31 31 142 91 16 60 ----- Sisi ke- 3 ----- 104 107 2 63 82 34 52 81 14 50 123 19 32 67 17 65 15 70 74 82 15 124 125 73 2 43 83 98 908 23 25 157 925 68 6 20 50 590 33 60 51 60 11 115 145 26 12 100 36 64 58 8 30 31 31 142 91 16 60 ----- Sisi ke- 4 ----- 125 107 2 63 82 34 52 81 14 50 123 19 32 67 17 65 15 70 74 82 15 124 125 73 2 43 83 98 908 23 25 157 925 68 6 20 50 590 33 60 51 60 11 115 145 26 12 100 36 64 58 8 30 31 31 142 91 16 60 ----- Sisi ke- 5 ----- 125 107 2 63 82 34 52 81 14 50 123 19 32 67 17 65 15 70 74 82 15 124 125 73 2 43 83 98 908 23 25 157 925 68 6 20 50 590 33 60 51 60 11 115 145 26 12 100 36 64 58 8 30 31 31 142 91 16 60 </pre>					

Nilai Objective Function Awal: 7570

Nilai Objective Function Akhir: 7112
(Best: 5520)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 7112

Nilai Objective Function Rata-rata dari Populasi Terakhir: 7189.7

Nilai Objective Function Terbaik dari Seluruh Populasi: 5520

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 402 ms

2.2 - Jumlah Populasi 100

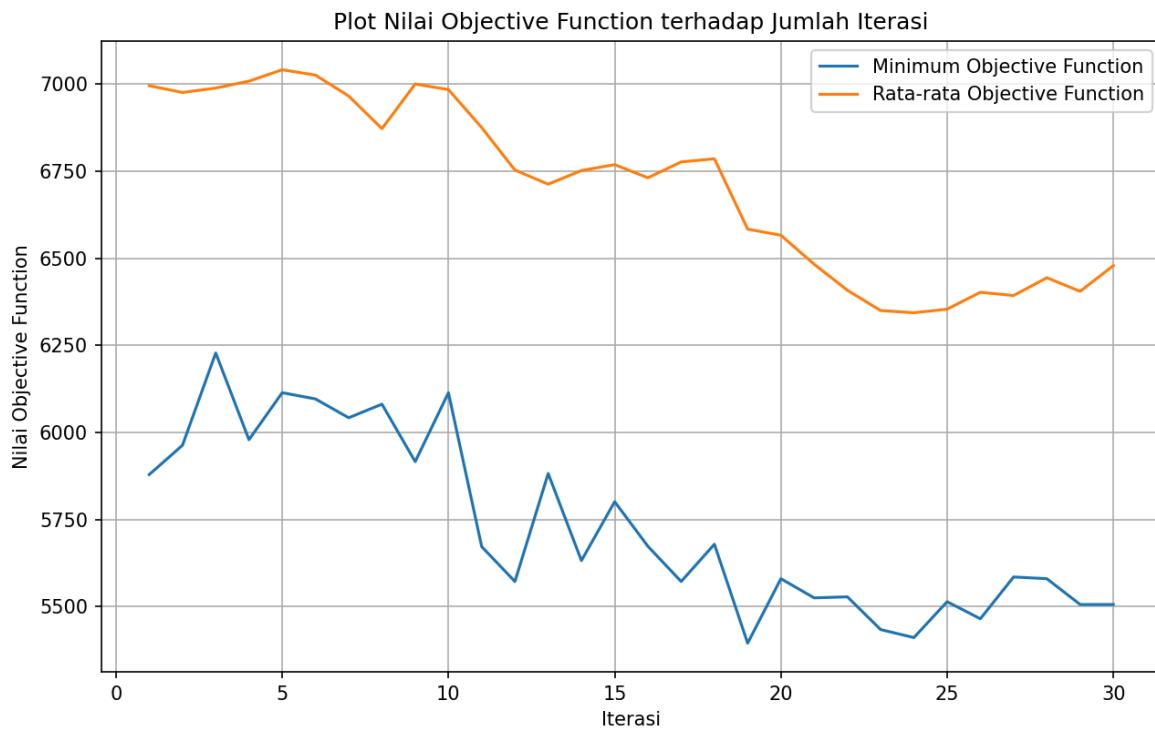
Percobaan 1

State Awal Cube

State Akhir Cube

Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>----- Sisi ke- 1 -----</p> 33 28 50 66 67 57 19 60 112 40 3 85 97 39 1 54 49 20 102 32 38 42 108 115 23 <p>----- Sisi ke- 2 -----</p> 46 103 34 120 12 56 52 37 106 35 121 51 74 96 110 24 70 75 118 82 99 69 114 78 10 <p>----- Sisi ke- 3 -----</p> 8 71 5 104 30 92 62 22 125 116 83 44 48 95 88 36 65 94 27 122 31 58 59 91 14 <p>----- Sisi ke- 4 -----</p> 73 76 17 101 81 7 80 64 107 16 119 90 84 72 45 13 87 29 93 63 6 98 86 79 11 <p>----- Sisi ke- 5 -----</p> 111 123 55 18 43 105 25 15 41 68 113 117 4 77 21 53 109 61 9 47 26 2 124 100 89	<p>----- Sisi ke- 1 -----</p> 33 57 3 54 38 46 56 121 24 99 8 92 83 36 31 73 7 119 13 6 111 105 113 53 26 <p>----- Sisi ke- 2 -----</p> 28 19 85 49 42 103 52 51 70 69 71 62 44 65 58 76 80 90 87 98 123 25 117 109 2 <p>----- Sisi ke- 3 -----</p> 50 60 97 20 108 34 37 74 75 114 5 22 48 94 59 17 64 84 29 86 55 15 4 61 124 <p>----- Sisi ke- 4 -----</p> 66 112 39 102 115 120 106 96 118 78 104 125 95 27 91 101 107 72 93 79 18 41 77 9 100 <p>----- Sisi ke- 5 -----</p> 67 40 1 32 23 12 35 110 82 10 30 116 88 122 14 81 16 45 63 11 43 68 21 47 89	<p>----- Sisi ke- 1 -----</p> 33 46 8 73 111 28 103 71 76 123 50 34 5 17 55 66 120 104 101 18 67 12 30 81 43 <p>----- Sisi ke- 2 -----</p> 57 56 92 7 105 19 52 62 80 25 60 37 22 64 15 112 106 125 107 41 40 35 116 16 68 <p>----- Sisi ke- 3 -----</p> 3 121 83 119 113 85 51 44 90 117 97 74 48 84 4 39 96 95 72 77 1 110 88 45 21 <p>----- Sisi ke- 4 -----</p> 54 24 36 13 53 49 79 65 87 109 20 75 94 29 61 102 118 127 93 9 32 82 122 63 47 <p>----- Sisi ke- 5 -----</p> 38 99 31 6 26 42 69 58 98 12 108 114 59 86 124 115 78 91 79 100 23 10 14 11 89	<p>----- Sisi ke- 1 -----</p> 5 70 91 111 51 98 124 88 25 120 9 22 88 21 103 89 58 68 33 37 47 65 13 99 90 <p>----- Sisi ke- 2 -----</p> 59 42 5 22 86 54 29 114 41 65 58 114 63 72 15 112 41 67 44 104 31 65 75 48 51 <p>----- Sisi ke- 3 -----</p> 91 97 88 68 13 5 40 63 67 75 102 36 86 46 89 111 68 120 60 75 64 84 49 91 124 <p>----- Sisi ke- 4 -----</p> 111 25 21 33 99 22 46 72 44 48 26 39 36 78 122 113 62 105 83 31 25 38 95 69 103 <p>----- Sisi ke- 5 -----</p> 51 120 103 37 90 86 61 15 104 51 82 101 49 124 34 45 94 65 88 113 50 122 26 73 57	<p>----- Sisi ke- 1 -----</p> 5 59 27 35 91 70 42 63 14 33 91 5 102 111 64 111 22 26 113 25 51 86 82 45 50 <p>----- Sisi ke- 2 -----</p> 98 54 94 72 115 124 29 56 42 14 97 40 36 68 84 25 46 39 62 38 120 61 101 94 122 <p>----- Sisi ke- 3 -----</p> 9 58 110 37 39 22 114 53 46 23 88 63 86 120 49 21 72 36 105 95 103 15 49 65 26 <p>----- Sisi ke- 4 -----</p> 89 112 32 121 55 58 41 29 114 84 68 67 46 60 91 33 44 78 83 69 37 104 124 88 73 <p>----- Sisi ke- 5 -----</p> 47 31 66 76 36 65 65 116 62 1 13 75 80 75 124 99 48 122 31 103 90 51 34 113 57	
Nilai Objective Function Awal: 7293			Nilai Objective Function Akhir: 5506(Best: 5395)		

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5506

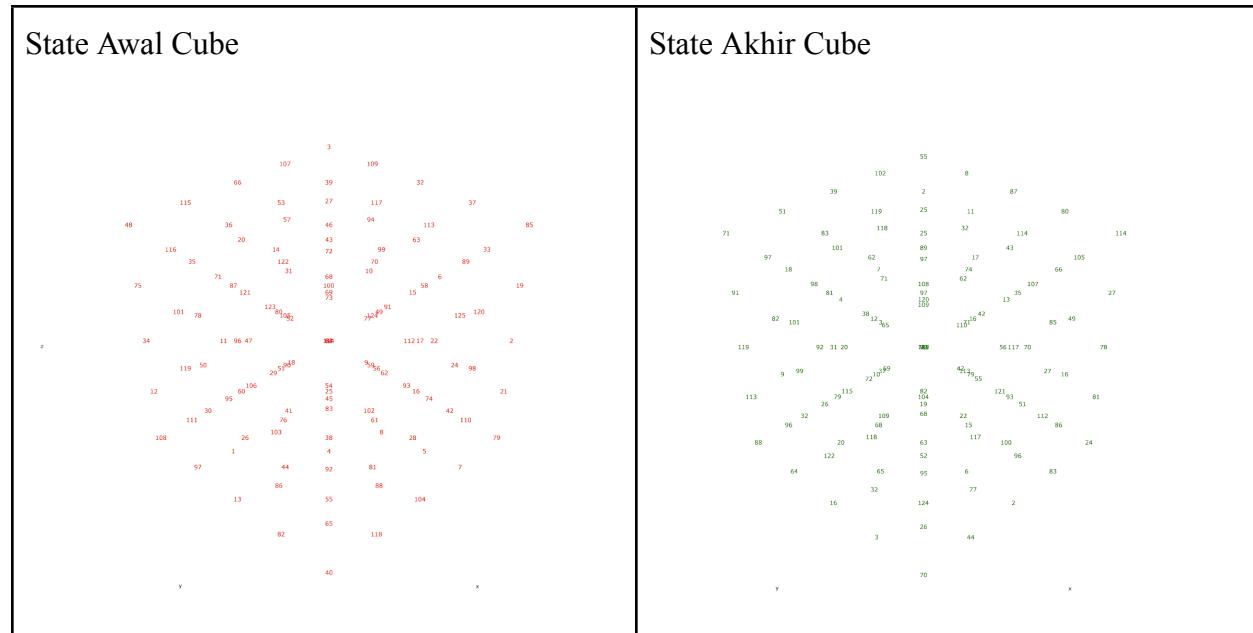
Nilai Objective Function Rata-rata dari Populasi Terakhir: 6478.75

Nilai Objective Function Terbaik dari Seluruh Populasi: 5395

Banyak Iterasi yang Dilakukan: 30

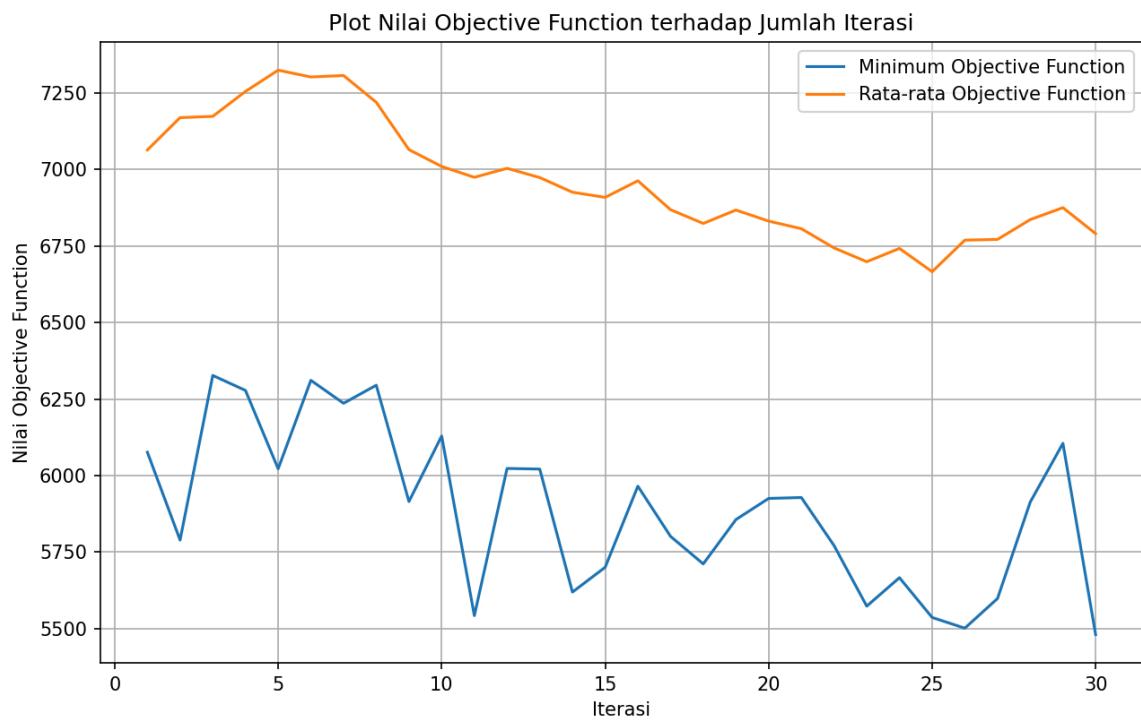
Durasi Proses Pencarian: 1.0425224304199219 ms

Percobaan 2



Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>----- Sisi ke- 1 -----</p> 84 73 72 27 3 9 77 10 94 109 93 112 15 63 32 42 24 125 89 37 79 21 2 19 85 <p>----- Sisi ke- 2 -----</p> 18 52 31 57 107 54 114 69 43 39 102 59 124 70 117 28 16 17 58 113 7 110 98 120 33 <p>----- Sisi ke- 3 -----</p> 106 47 121 28 66 41 90 105 122 53 38 25 64 100 46 81 61 56 49 99 104 5 74 22 6 <p>----- Sisi ke- 4 -----</p> 30 50 78 35 115 26 60 96 87 36 44 76 51 86 14 55 4 45 23 68 118 88 8 62 91 <p>----- Sisi ke- 5 -----</p> 108 12 34 75 48 97 11 119 181 116 13 1 95 11 71 82 8 103 29 123 40 65 99 83 67	<p>----- Sisi ke- 1 -----</p> 84 9 93 42 79 18 54 102 28 7 106 41 38 81 104 38 26 44 55 118 108 97 13 82 40 <p>----- Sisi ke- 2 -----</p> 73 77 112 24 21 52 114 59 16 110 47 90 25 61 5 58 68 76 4 88 12 111 1 86 65 <p>----- Sisi ke- 3 -----</p> 72 10 15 125 2 31 69 124 17 98 121 105 64 56 74 78 96 51 45 8 34 119 95 103 92 <p>----- Sisi ke- 4 -----</p> 27 94 63 89 19 57 43 70 58 120 28 122 100 49 22 35 87 80 23 62 75 181 11 29 83 <p>----- Sisi ke- 5 -----</p> 3 109 32 37 85 107 39 117 113 33 66 53 46 99 6 115 36 14 68 91 48 116 71 123 67	<p>----- Sisi ke- 1 -----</p> 84 18 106 38 108 73 52 47 50 12 72 31 121 78 34 27 57 28 35 75 3 107 66 115 48 <p>----- Sisi ke- 2 -----</p> 9 54 41 26 97 77 114 98 60 111 10 69 105 96 119 94 43 122 87 101 109 39 53 36 116 <p>----- Sisi ke- 3 -----</p> 93 102 38 44 13 112 59 25 76 1 15 124 64 51 95 63 70 100 88 11 32 117 46 14 71 <p>----- Sisi ke- 4 -----</p> 42 28 81 55 82 24 16 61 4 86 125 17 56 45 103 89 58 49 23 29 37 113 99 68 123 <p>----- Sisi ke- 5 -----</p> 79 7 104 118 40 21 118 5 88 65 2 98 74 8 92 19 126 22 62 83 85 33 6 91 67	<p>----- Sisi ke- 1 -----</p> 91 109 8 17 104 77 27 86 43 73 117 58 122 66 35 53 55 85 66 80 46 58 54 76 37 <p>----- Sisi ke- 2 -----</p> 69 65 71 118 182 82 98 128 89 2 22 113 71 74 11 38 69 67 62 48 124 26 34 11 119 <p>----- Sisi ke- 3 -----</p> 102 12 189 13 68 76 97 14 14 93 19 23 25 39 52 59 23 71 42 73 54 98 12 111 50 <p>----- Sisi ke- 4 -----</p> 17 19 33 18 51 20 79 31 81 83 65 68 10 12 62 124 52 79 119 65 93 123 184 101 112 <p>----- Sisi ke- 5 -----</p> 31 52 93 12 79 128 62 4 119 78 77 104 26 92 98 3 32 118 72 38 70 26 95 68 9	<p>----- Sisi ke- 1 -----</p> 91 77 117 53 46 69 82 22 38 124 102 76 19 59 54 17 20 65 124 93 31 120 77 3 70 <p>----- Sisi ke- 2 -----</p> 109 27 58 55 58 65 90 113 69 26 12 97 23 23 98 19 79 68 52 123 52 62 104 32 26 <p>----- Sisi ke- 3 -----</p> 117 22 19 65 77 58 113 23 68 184 122 71 25 18 26 66 74 39 12 92 35 11 52 62 98 <p>----- Sisi ke- 4 -----</p> 53 38 59 124 3 55 69 23 52 32 85 67 71 70 118 66 62 42 119 72 80 48 73 65 38 <p>----- Sisi ke- 5 -----</p> 46 124 54 93 70 58 26 98 123 26 54 34 12 104 95 76 11 111 101 68 37 119 58 112 9	
Nilai Objective Function Awal: 6360			Nilai Objective Function Akhir: 5480(Best: 5480)		

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5480

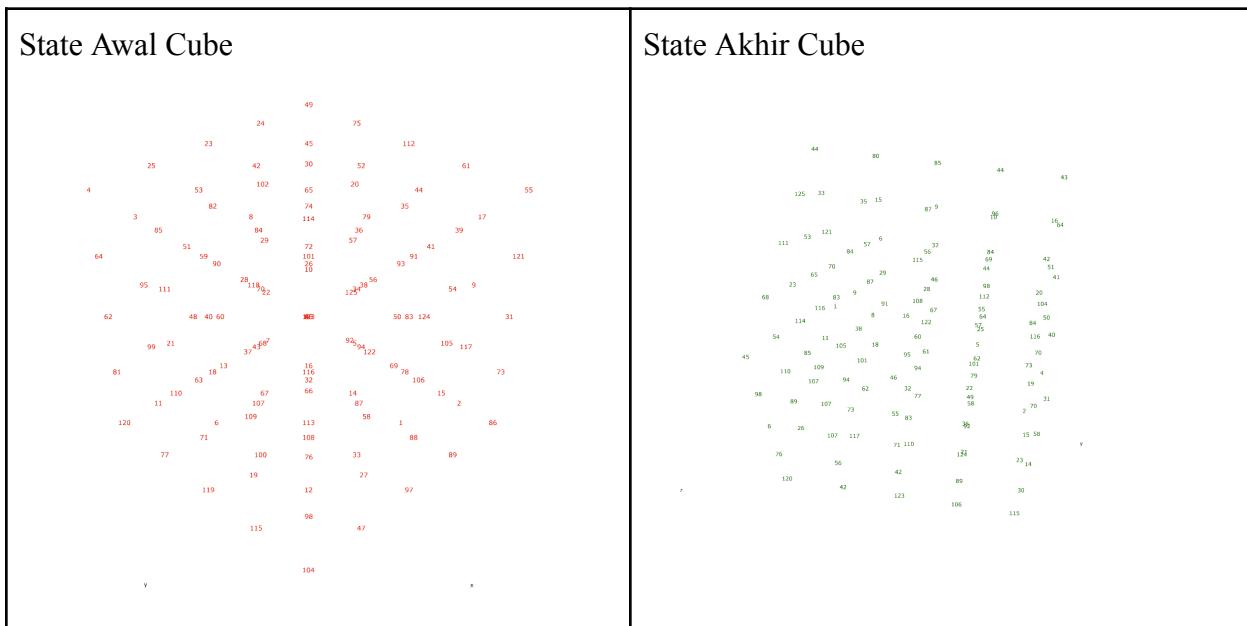
Nilai Objective Function Rata-rata dari Populasi Terakhir: 6790.36

Nilai Objective Function Terbaik dari Seluruh Populasi: 5480

Banyak Iterasi yang Dilakukan: 30

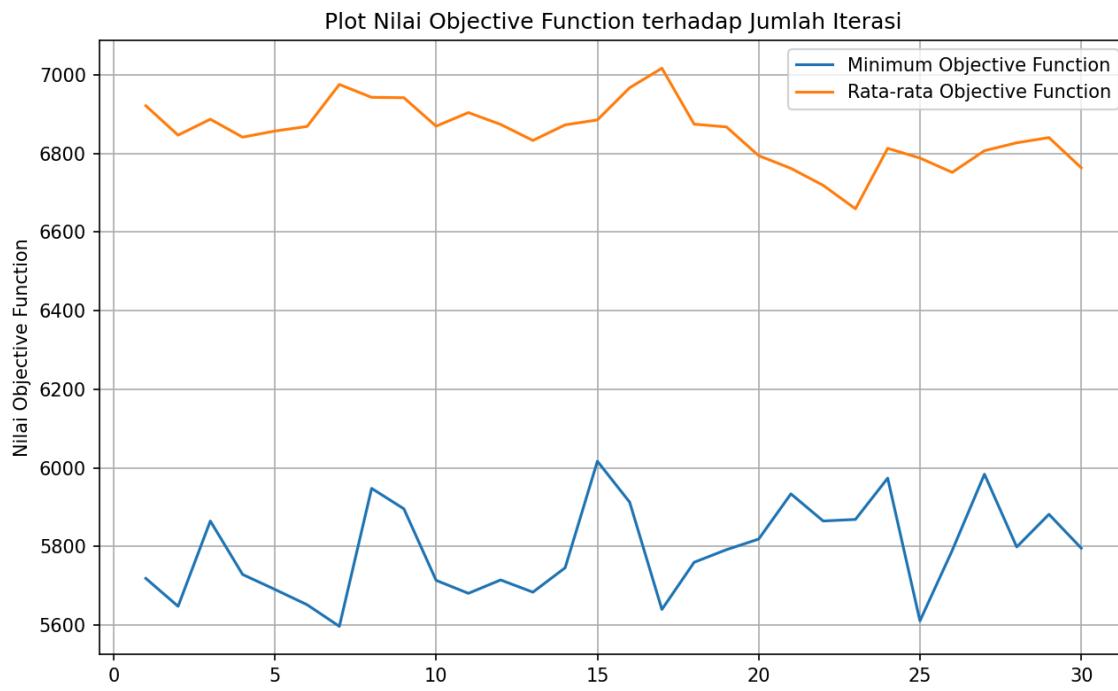
Durasi Proses Pencarian: 1.0670976638793945 ms

Percobaan 3



Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>----- Sisi ke- 1 -----</p> 46 19 114 30 49 92 125 57 20 75 69 50 93 35 112 15 105 54 39 61 86 73 31 121 55 <p>----- Sisi ke- 2 -----</p> 7 22 29 102 24 16 80 26 74 45 14 5 34 36 52 1 78 83 91 44 89 2 117 9 17 <p>----- Sisi ke- 3 -----</p> 13 60 90 82 23 67 68 70 84 42 113 116 96 101 65 33 87 94 38 79 97 88 106 124 41 <p>----- Sisi ke- 4 -----</p> 110 21 111 85 25 6 18 40 59 53 100 107 43 118 8 12 108 32 123 72 47 27 58 122 56 <p>----- Sisi ke- 5 -----</p> 120 81 62 64 4 77 11 99 95 3 119 71 63 48 51 115 19 109 37 28 104 98 76 66 103	<p>----- Sisi ke- 1 -----</p> 46 92 69 15 86 7 16 14 1 89 13 67 113 33 97 110 6 100 12 47 120 77 119 115 104 <p>----- Sisi ke- 2 -----</p> 10 125 58 105 73 22 80 5 78 2 60 68 116 87 88 21 18 107 108 27 81 11 71 19 98 <p>----- Sisi ke- 3 -----</p> 114 57 93 54 31 29 26 34 83 117 90 70 96 94 106 111 40 43 32 58 62 99 63 109 76 <p>----- Sisi ke- 4 -----</p> 30 20 35 39 121 102 74 36 91 9 82 84 101 38 124 85 59 118 123 122 64 95 48 37 66 <p>----- Sisi ke- 5 -----</p> 49 75 112 61 55 24 45 52 44 17 23 42 65 79 41 25 53 8 72 56 4 3 51 28 103	<p>----- Sisi ke- 1 -----</p> 46 7 13 110 120 18 22 68 21 81 114 29 98 111 62 30 102 82 85 64 49 24 23 25 4 <p>----- Sisi ke- 2 -----</p> 92 16 67 6 77 125 80 68 18 11 57 26 70 40 99 20 74 84 59 95 75 45 42 53 3 <p>----- Sisi ke- 3 -----</p> 69 14 113 100 119 58 5 116 107 71 93 34 96 43 63 35 36 101 118 148 112 52 65 8 51 <p>----- Sisi ke- 4 -----</p> 15 1 33 12 115 105 78 87 108 19 54 83 94 32 109 39 91 38 123 37 61 44 79 72 28 <p>----- Sisi ke- 5 -----</p> 86 89 97 47 104 73 2 88 27 98 31 117 106 58 76 121 9 124 122 66 55 17 41 56 103	<p>----- Sisi ke- 1 -----</p> 48 115 56 123 114 29 106 92 1 89 97 54 81 68 23 83 88 64 73 63 43 121 27 78 23 <p>----- Sisi ke- 2 -----</p> 95 29 97 83 43 95 81 18 53 103 20 15 64 33 74 109 31 30 121 110 7 103 61 110 36 <p>----- Sisi ke- 3 -----</p> 48 95 20 109 7 115 7 119 86 112 56 27 121 21 45 123 5 65 94 51 114 12 107 73 50 <p>----- Sisi ke- 4 -----</p> 29 81 15 31 103 106 58 108 79 13 92 55 56 114 111 1 42 16 77 68 89 82 116 86 81 <p>----- Sisi ke- 5 -----</p> 97 18 64 30 61 54 114 88 56 14 81 72 70 2 106 60 14 64 89 125 23 117 29 54 28	<p>Nilai Objective Function Awal: 6939</p> <p>Nilai Objective Function Akhir: 5796(Best: 5597)</p>	

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 5796

Nilai Objective Function Rata-rata dari Populasi Terakhir: 6763.69

Nilai Objective Function Terbaik dari Seluruh Populasi: 5597

Banyak Iterasi yang Dilakukan: 30

Durasi Proses Pencarian: 1.054060935974121 ms

2.3 - Jumlah Populasi 1000

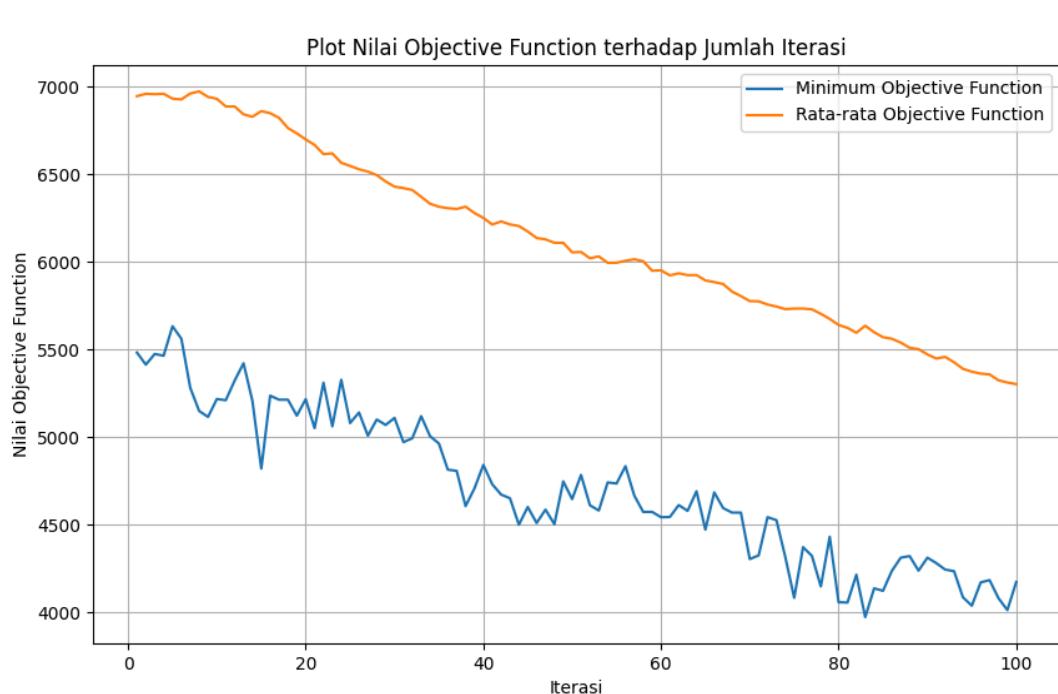
Percobaan 1

State Awal Cube			State Akhir Cube		
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
State Awal Cube: A 3D grid of numbers representing the initial state of a cube. The numbers are colored red, indicating they are part of the current population. 	State Akhir Cube: A 3D grid of numbers representing the final state of the cube after 30 iterations. The numbers are colored green, indicating they are part of the final population. 				

Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z	Sisi ke sumbu x	Sisi ke sumbu y	Sisi ke sumbu z
----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----	----- Sisi ke- 1 -----
37 52 125 72 45 64 109 111 83 25 22 17 66 15 47 63 96 118 69 7 35 26 106 90 80	37 64 22 63 35 117 91 14 87 88 3 123 82 21 31 67 29 181 51 78 34 27 18 93 71	37 117 3 67 34 52 103 110 28 9 125 59 73 60 68 72 46 43 99 54 45 84 95 75 19	57 35 31 49 60 53 112 102 74 17 69 76 59 123 22 74 77 78 8 77 59 71 93 64 64	57 53 69 74 59 88 75 25 67 91 97 11 91 79 77 43 11 77 38 112 49 123 27 58 96	57 88 97 43 49 35 29 80 113 45 31 109 21 58 94 49 46 45 78 33 60 53 35 43 65
----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----	----- Sisi ke- 2 -----
117 103 59 46 84 91 13 94 104 57 14 33 63 89 76 87 16 124 55 81 88 79 36 56 115	52 109 17 96 26 103 13 33 16 79 110 24 2 78 48 28 39 105 11 32 9 85 113 122 38	64 91 123 29 27 109 13 24 30 85 111 94 108 20 62 83 104 100 74 44 25 57 98 23 112	88 29 109 46 53 75 87 68 16 77 83 104 100 74 44 67 85 20 57 73 91 36 86 57 40	35 112 76 77 71 29 87 90 85 36 80 65 52 122 87 113 46 13 77 102 45 58 54 36 79	53 75 11 11 123 112 87 65 46 58 102 68 105 30 70 74 16 37 1 99 17 77 85 73 49
----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----	----- Sisi ke- 3 -----
3 110 73 43 95 123 24 108 100 98 82 2 53 92 107 21 78 65 77 4 31 48 41 114 86	125 111 66 118 106 59 94 61 124 36 73 108 53 65 41 60 20 12 121 49 68 62 58 39 10	22 14 82 101 18 17 33 2 105 113 66 61 53 12 58 15 89 92 8 116 47 76 107 5 40	97 80 21 45 35 11 65 105 37 85 91 52 58 94 75 79 122 68 70 37 77 87 31 59 71	31 102 59 78 93 109 68 46 20 86 21 105 58 68 31 50 30 90 69 78 94 70 55 95 15	69 25 91 77 27 76 90 52 13 54 59 46 58 90 55 123 21 94 51 80 22 71 75 100 35
----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----	----- Sisi ke- 4 -----
67 28 60 99 75 29 30 20 74 23 101 105 12 18 5 51 11 121 1 58 70 32 49 97 6	72 83 15 69 90 46 104 89 55 56 43 100 92 77 114 99 74 18 1 97 54 44 116 102 42	63 87 21 51 93 96 16 78 11 122 118 124 65 121 39 69 55 77 1 102 7 81 4 58 120	43 113 50 78 43 11 46 30 1 73 77 13 90 51 100 38 77 69 59 76 112 102 78 93 76	49 74 123 8 64 46 16 21 57 57 45 37 94 70 59 78 1 51 59 93 33 99 80 42 59	74 67 79 38 58 77 85 122 77 36 78 20 66 69 95 8 57 70 59 42 77 73 37 76 64
----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----	----- Sisi ke- 5 -----
34 9 68 54 19 27 85 62 44 112 18 113 50 116 140 93 122 39 102 120 71 38 10 42 119	45 25 147 7 80 84 57 176 83 115 95 98 187 4 86 75 23 5 58 6 19 112 40 120 119	35 88 31 70 71 26 79 48 32 38 186 36 41 49 18 90 56 114 97 42 80 115 86 6 119	49 45 94 33 65 123 58 70 99 49 27 54 55 80 35 58 36 95 42 64 96 79 15 59 48	60 17 22 77 64 53 77 71 73 40 35 85 75 37 71 43 73 100 76 76 65 49 35 64 48	59 91 77 112 96 71 36 87 102 79 93 86 31 78 15 64 57 59 93 59 64 40 71 76 48

Nilai Objective Function Awal: 6820

Nilai Objective Function Akhir: 4172 (Best: 4121)



Nilai Objective Function Terendah dari Populasi Terakhir: 4173

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5301.815

Nilai Objective Function Terbaik dari Seluruh Populasi: 3973

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 90367 ms

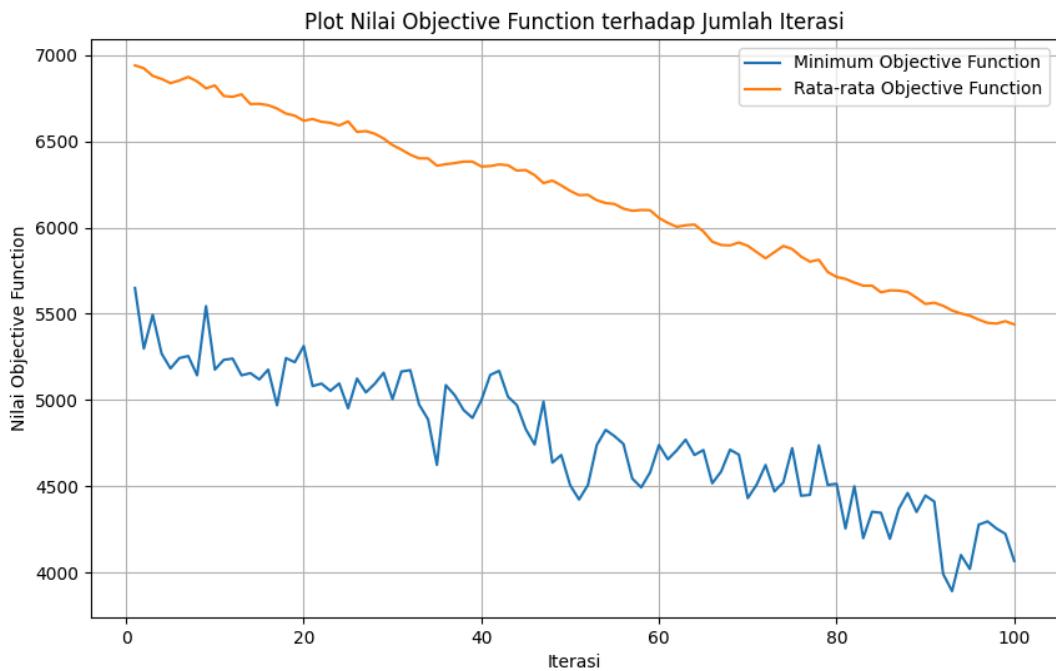
Percobaan 2

State Awal Cube		State Akhir Cube			
Sumbu X	Sumbu Y	Sumbu Z	Sumbu X	Sumbu Y	Sumbu Z
<p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>30 50 65 104 21 45 49 101 72 33 78 40 114 31 61 14 19 41 29 32 97 96 6 77 24 </pre> <p>----- Sisi ke- 2 -----</p> <pre>80 68 93 85 108 44 4 100 54 39 102 107 2 48 10 117 47 113 84 1 109 91 5 83 27 </pre> <p>----- Sisi ke- 3 -----</p> <pre>37 105 23 125 52 103 55 17 57 124 86 118 66 51 98 18 94 81 74 106 7 121 92 98 112 </pre> <p>----- Sisi ke- 4 -----</p> <pre>87 88 26 116 111 56 70 64 69 36 62 28 15 99 89 42 28 73 67 115 59 11 123 120 46 </pre> <p>----- Sisi ke- 5 -----</p> <pre>119 8 16 95 13 9 63 53 68 22 75 38 25 3 43 58 79 71 122 76 110 12 34 82 35 </pre> <p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>30 45 78 14 97 80 44 102 117 109 37 103 86 18 7 87 56 62 42 59 119 9 75 58 110 </pre> <p>----- Sisi ke- 2 -----</p> <pre>50 49 48 19 96 68 4 107 47 91 105 55 118 94 121 88 70 20 28 11 8 63 58 79 12 </pre> <p>----- Sisi ke- 3 -----</p> <pre>65 101 114 41 6 93 100 2 113 5 23 17 66 81 92 26 16 45 73 123 16 53 25 71 34 </pre> <p>----- Sisi ke- 4 -----</p> <pre>104 72 31 29 77 85 54 48 84 83 125 57 51 74 98 116 69 99 67 120 99 60 3 122 82 </pre> <p>----- Sisi ke- 5 -----</p> <pre>21 33 61 32 24 108 39 10 1 27 52 124 90 106 112 111 36 89 115 46 13 22 43 76 35 </pre> <p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>30 80 37 87 119 50 68 105 88 8 65 93 23 26 16 104 85 125 116 95 21 108 52 111 13 </pre> <p>----- Sisi ke- 2 -----</p> <pre>45 44 103 56 9 49 4 55 70 63 101 100 17 64 53 72 54 57 69 68 33 39 124 36 22 </pre> <p>----- Sisi ke- 3 -----</p> <pre>78 102 86 62 75 40 107 118 20 38 114 2 66 15 25 51 48 51 99 3 61 10 90 89 43 </pre> <p>----- Sisi ke- 4 -----</p> <pre>14 117 18 42 58 19 47 94 28 79 41 113 81 73 71 29 84 74 67 122 32 1 106 115 76 </pre> <p>----- Sisi ke- 5 -----</p> <pre>97 109 7 59 110 96 91 121 11 12 6 5 92 123 34 77 83 98 120 82 24 27 112 46 35 </pre> <p>Sisi ke sumbu x</p> <p>----- Sisi ke- 1 -----</p> <pre>86 54 35 21 57 93 78 92 37 50 73 59 52 43 38 14 124 30 81 72 75 57 54 102 62 </pre> <p>----- Sisi ke- 2 -----</p> <pre>18 56 53 103 40 108 83 97 25 116 47 54 92 60 83 72 64 10 102 15 98 64 56 53 51 </pre> <p>----- Sisi ke- 3 -----</p> <pre>77 80 57 10 33 40 107 118 20 38 114 2 66 15 25 51 48 51 99 3 2 121 72 57 94 </pre> <p>----- Sisi ke- 4 -----</p> <pre>21 37 43 81 102 103 25 60 102 53 10 113 36 98 57 110 70 74 66 66 6 89 111 40 121 </pre> <p>----- Sisi ke- 5 -----</p> <pre>57 50 38 72 62 49 116 83 15 51 33 65 40 89 94 50 23 89 93 63 65 47 118 102 17 </pre> <p>Sisi ke sumbu y</p> <p>----- Sisi ke- 1 -----</p> <pre>86 93 73 14 75 18 108 47 72 98 77 21 105 99 2 76 25 86 90 63 47 73 62 64 74 </pre> <p>----- Sisi ke- 2 -----</p> <pre>54 78 59 124 57 56 83 64 64 64 80 24 88 28 121 3 60 78 55 75 74 96 60 63 42 </pre> <p>----- Sisi ke- 3 -----</p> <pre>35 92 52 30 54 53 97 92 10 56 57 42 60 83 72 84 79 53 77 61 113 16 79 35 25 </pre> <p>----- Sisi ke- 4 -----</p> <pre>21 37 43 81 102 103 25 60 102 53 10 113 36 98 57 110 70 74 66 66 6 89 111 40 121 </pre> <p>----- Sisi ke- 5 -----</p> <pre>75 98 2 63 74 57 64 121 55 42 54 56 72 61 25 102 53 57 66 121 62 51 94 63 17 </pre> <p>Sisi ke sumbu z</p> <p>----- Sisi ke- 1 -----</p> <pre>86 18 77 76 47 54 56 80 3 74 35 53 57 84 113 21 103 10 119 6 57 40 33 50 65 </pre> <p>----- Sisi ke- 2 -----</p> <pre>93 108 21 25 73 78 83 24 60 96 92 97 42 79 16 37 25 113 70 89 50 116 65 23 47 </pre> <p>----- Sisi ke- 3 -----</p> <pre>73 47 105 86 62 59 54 88 57 60 52 92 60 53 79 43 60 36 74 111 38 83 40 89 118 </pre> <p>----- Sisi ke- 4 -----</p> <pre>14 72 99 99 64 124 64 28 8 63 30 10 83 77 35 81 102 98 66 40 72 15 89 93 102 </pre> <p>----- Sisi ke- 5 -----</p>					

Nilai Objective Function Awal: 6895

Nilai Objective Function Akhir: 4067 (Best: 3892)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 4067

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5438.287

Nilai Objective Function Terbaik dari Seluruh Populasi: 2892

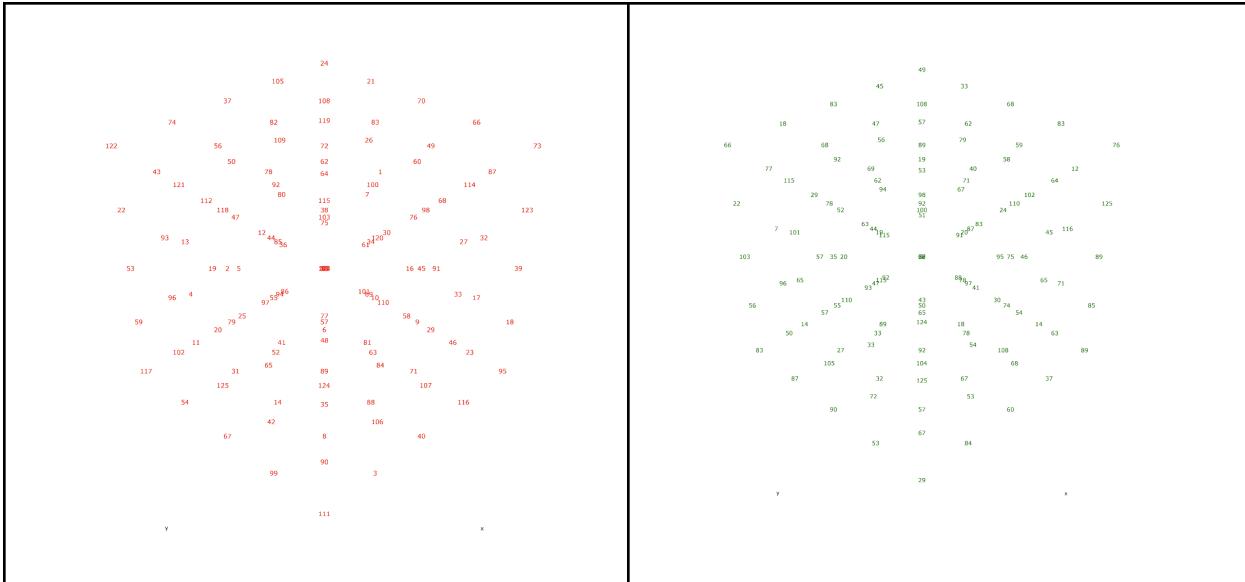
Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 89006 ms

Percobaan 3

State Awal Cube

State Akhir Cube



Sumbu X

Sumbu Y

Sumbu Z

Sumbu X

Sumbu Y

Sumbu Z

----- Sisi ke- 1 -----

51 | 75 | 64 | 119 | 24 |
101 | 61 | 7 | 26 | 21 |
58 | 16 | 76 | 68 | 70 |
46 | 33 | 27 | 114 | 66 |
95 | 18 | 39 | 123 | 73 |

----- Sisi ke- 2 -----

86 | 36 | 80 | 109 | 105 |
77 | 15 | 103 | 62 | 108 |
81 | 69 | 34 | 100 | 83 |
71 | 9 | 45 | 98 | 49 |
116 | 23 | 17 | 32 | 87 |

----- Sisi ke- 3 -----

25 | 5 | 47 | 50 | 37 |
41 | 94 | 85 | 92 | 82 |
89 | 57 | 113 | 38 | 72 |
88 | 63 | 10 | 120 | 1 |
40 | 107 | 29 | 91 | 68 |

----- Sisi ke- 4 -----

11 | 4 | 13 | 121 | 74 |
31 | 79 | 2 | 118 | 56 |
14 | 52 | 55 | 44 | 78 |
8 | 124 | 6 | 28 | 115 |
3 | 106 | 84 | 110 | 30 |

----- Sisi ke- 5 -----

117 | 59 | 53 | 22 | 122 |
54 | 102 | 96 | 93 | 43 |
67 | 125 | 20 | 19 | 112 |
99 | 42 | 65 | 97 | 12 |
111 | 90 | 35 | 48 | 104 |

----- Sisi ke- 1 -----

51 | 101 | 58 | 46 | 95 |
86 | 77 | 81 | 71 | 116 |
25 | 41 | 89 | 88 | 48 |
11 | 31 | 14 | 8 | 3 |
117 | 54 | 67 | 99 | 111 |

----- Sisi ke- 2 -----

75 | 61 | 16 | 33 | 18 |
36 | 15 | 69 | 9 | 23 |
5 | 94 | 57 | 63 | 107 |
4 | 79 | 52 | 124 | 106 |
59 | 102 | 125 | 42 | 90 |

----- Sisi ke- 3 -----

64 | 7 | 76 | 27 | 39 |
80 | 103 | 34 | 45 | 17 |
47 | 85 | 113 | 10 | 29 |
13 | 2 | 55 | 6 | 84 |
53 | 96 | 20 | 65 | 35 |

----- Sisi ke- 4 -----

119 | 26 | 60 | 114 | 123 |
109 | 62 | 100 | 98 | 32 |
50 | 92 | 38 | 120 | 91 |
121 | 118 | 44 | 28 | 110 |
22 | 93 | 19 | 97 | 48 |

----- Sisi ke- 5 -----

24 | 21 | 70 | 66 | 73 |
105 | 108 | 83 | 49 | 87 |
37 | 82 | 72 | 1 | 68 |
74 | 56 | 78 | 115 | 30 |
122 | 43 | 112 | 12 | 104 |

----- Sisi ke- 1 -----

51 | 101 | 58 | 46 | 95 |
86 | 77 | 81 | 71 | 116 |
25 | 41 | 89 | 88 | 48 |
11 | 31 | 14 | 8 | 3 |
117 | 54 | 67 | 99 | 111 |

----- Sisi ke- 2 -----

75 | 61 | 16 | 33 | 18 |
36 | 15 | 69 | 9 | 23 |
5 | 94 | 57 | 63 | 107 |
4 | 79 | 52 | 124 | 106 |
59 | 102 | 125 | 42 | 90 |

----- Sisi ke- 3 -----

64 | 7 | 76 | 27 | 39 |
80 | 103 | 34 | 45 | 17 |
47 | 85 | 113 | 10 | 29 |
13 | 2 | 55 | 6 | 84 |
53 | 96 | 20 | 65 | 35 |

----- Sisi ke- 4 -----

119 | 26 | 60 | 114 | 123 |
109 | 62 | 100 | 98 | 32 |
50 | 92 | 38 | 120 | 91 |
121 | 118 | 44 | 28 | 110 |
22 | 93 | 19 | 97 | 48 |

----- Sisi ke- 5 -----

24 | 21 | 70 | 66 | 73 |
105 | 108 | 83 | 49 | 87 |
37 | 82 | 72 | 1 | 68 |
74 | 56 | 78 | 115 | 30 |
122 | 43 | 112 | 12 | 104 |

----- Sisi ke- 1 -----

26 | 64 | 53 | 56 | 49 |
88 | 91 | 67 | 79 | 54 |
30 | 95 | 24 | 58 | 116 |
83 | 50 | 75 | 97 | 16 |
95 | 85 | 89 | 33 | 74 |

----- Sisi ke- 2 -----

92 | 112 | 45 | 77 | 94 |
39 | 58 | 82 | 71 | 108 |
30 | 77 | 107 | 48 | 3 |
68 | 46 | 34 | 88 | 72 |
41 | 100 | 76 | 78 | 19 |

----- Sisi ke- 3 -----

53 | 100 | 49 | 83 | 29 |
100 | 41 | 46 | 11 | 105 |
59 | 74 | 66 | 49 | 60 |
53 | 48 | 49 | 87 | 40 |
47 | 68 | 54 | 46 | 100 |

----- Sisi ke- 4 -----

23 | 30 | 96 | 62 | 77 |
98 | 64 | 9 | 109 | 41 |
40 | 1 | 29 | 95 | 101 |
43 | 97 | 118 | 56 | 1 |
37 | 64 | 61 | 41 | 113 |

----- Sisi ke- 5 -----

85 | 58 | 100 | 38 | 84 |
87 | 50 | 96 | 60 | 77 |
82 | 43 | 36 | 82 | 29 |
49 | 72 | 33 | 83 | 63 |
28 | 40 | 117 | 88 | 45 |

----- Sisi ke- 1 -----

26 | 88 | 38 | 83 | 95 |
92 | 39 | 30 | 68 | 41 |
53 | 100 | 59 | 53 | 47 |
23 | 98 | 40 | 43 | 37 |
85 | 87 | 82 | 49 | 28 |

----- Sisi ke- 2 -----

64 | 91 | 95 | 50 | 85 |
112 | 58 | 77 | 46 | 100 |
100 | 41 | 74 | 48 | 68 |
30 | 64 | 1 | 97 | 64 |
58 | 50 | 43 | 72 | 40 |

----- Sisi ke- 3 -----

53 | 67 | 24 | 75 | 89 |
45 | 82 | 107 | 34 | 76 |
49 | 46 | 66 | 49 | 54 |
96 | 9 | 29 | 118 | 61 |
100 | 96 | 36 | 33 | 117 |

----- Sisi ke- 4 -----

56 | 79 | 58 | 97 | 33 |
77 | 71 | 48 | 88 | 78 |
83 | 11 | 49 | 87 | 46 |
62 | 109 | 95 | 56 | 41 |
38 | 60 | 82 | 83 | 88 |

----- Sisi ke- 5 -----

49 | 54 | 116 | 16 | 74 |
94 | 108 | 3 | 72 | 19 |
29 | 105 | 68 | 40 | 100 |
77 | 41 | 101 | 1 | 113 |
84 | 77 | 29 | 63 | 45 |

----- Sisi ke- 1 -----

26 | 92 | 53 | 23 | 85 |
64 | 112 | 100 | 30 | 58 |
53 | 45 | 49 | 96 | 100 |
56 | 77 | 83 | 62 | 38 |
49 | 94 | 29 | 77 | 84 |

----- Sisi ke- 2 -----

88 | 39 | 100 | 98 | 87 |
91 | 50 | 41 | 64 | 50 |
67 | 82 | 46 | 9 | 96 |
79 | 71 | 11 | 109 | 60 |
54 | 108 | 105 | 41 | 77 |

----- Sisi ke- 3 -----

30 | 30 | 59 | 40 | 82 |
95 | 77 | 74 | 1 | 43 |
24 | 107 | 66 | 29 | 36 |
58 | 48 | 49 | 95 | 82 |
116 | 3 | 60 | 101 | 29 |

----- Sisi ke- 4 -----

83 | 68 | 53 | 43 | 49 |
50 | 46 | 48 | 97 | 72 |
75 | 34 | 49 | 118 | 33 |
97 | 88 | 87 | 56 | 83 |
16 | 72 | 40 | 1 | 63 |

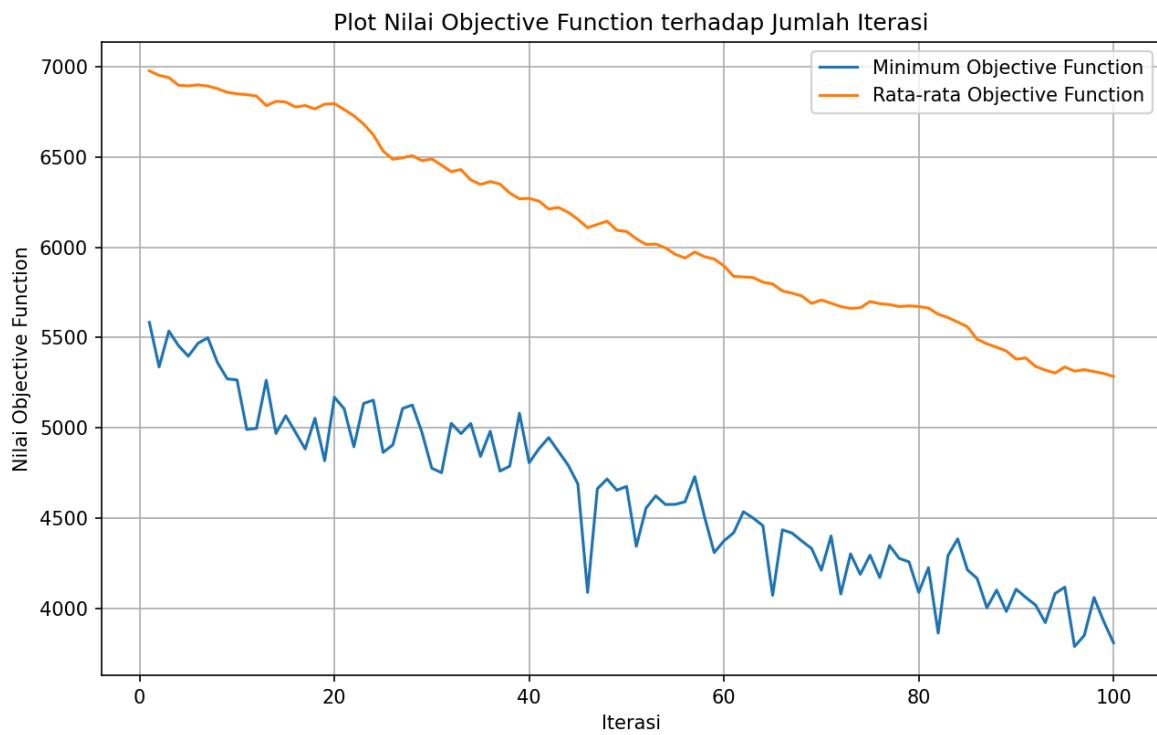
----- Sisi ke- 5 -----

95 | 41 | 47 | 37 | 28 |
85 | 100 | 68 | 64 | 40 |
89 | 76 | 54 | 61 | 117 |
33 | 78 | 46 | 41 | 88 |
74 | 19 | 100 | 113 | 45 |

Nilai Objective Function Awal: 7256

Nilai Objective Function Akhir: 3813(Best: 3792)

Plot Nilai Objective Function Minimum dan Rata-rata:



Nilai Objective Function Terendah dari Populasi Terakhir: 3813

Nilai Objective Function Rata-rata dari Populasi Terakhir: 5285.038

Nilai Objective Function Terbaik dari Seluruh Populasi: 3792

Banyak Iterasi yang Dilakukan: 100

Durasi Proses Pencarian: 88.065908432006836 ms

III.2 Analisis

1. Steepest Ascent Hill-climbing

Algoritma *Steepest Ascent Hill-climbing* ini bekerja dengan membandingkan *state*-nya sekarang dengan *neighbor state* disekitarnya, dan hanya akan berpindah *state* ketika *objective function* dari *state* tersebut lebih baik daripada *state*-nya sekarang. Oleh karena itu, jumlah yang diperlukan untuk mencapai *objective function* seringkali bermacam-macam dikarenakan *state* awal yang berbeda. Algoritma ini juga lebih sering mencapai *local optima* karena ketika disekitarnya tidak ada lagi *state* yang lebih baik nilainya, maka algoritma ini akan berhenti bekerja. Sehingga akan sulit untuk menemukan *global optima* untuk fungsi yang kompleks dan memiliki banyak *local optima*. Oleh karena itu, dibandingkan dengan algoritma lain yang memiliki cara menghindari terjebak pada *local optima*, algoritma ini kurang optimal untuk digunakan.

Jika dibandingkan dengan algoritma *hill-climbing* lainnya, durasi pencarian *steepest ascent* cenderung lebih cepat dibandingkan dengan *sideways move* dan *random restart* karena

steepest ascent akan langsung berhenti ketika disekitarnya tidak ada lagi *state* dengan nilai lebih tinggi. Untuk algoritma lain, tergantung dengan banyaknya iterasi / suhu / keturunan yang diinginkan, durasi pencarian dapat beragam. Hasil akhir dari algoritma ini akan selalu konsisten jika *state* awal kubus sama, namun karena *state* awal kubus acak, durasi dan *objective function* yang didapatkan juga beragam. Dari eksperimen yang dilakukan, algoritma ini bekerja cukup baik dengan rata-rata *objective function* sekitar +-600 (nilai maks *objective function* = 0).

2. Hill-climbing with Sideways Move

Seperti algoritma *Steepest Ascent Hill-climbing*, algoritma *sideways move* bekerja dengan membandingkan *state*-nya sekarang dengan *neighbor state* disekitarnya, dan akan berpindah *state* ketika *objective function* dari *state* tersebut lebih baik daripada *state*-nya sekarang, dengan tambahan kemampuan untuk pindah ke *state* yang memiliki *objective function* sama. Sehingga jika dibandingkan dengan *steepest ascent* algoritma ini mempunyai cara untuk menghindari jebakan *local optima* dengan melakukan *sideways move*, dengan begitu, kemungkinan mencapai *global optima* akan lebih besar, walau tetap ada batasan dalam eksplorasi karena hanya akan menerima solusi dengan nilai sama.

Jika dibandingkan dengan algoritma *hill climbing* lainnya, durasi pencarian *sideways move* cenderung lebih lama dibandingkan *steepest ascent* karena *sideways move* tidak langsung berhenti ketika mencapai *local optima*. Algoritma ini diizinkan untuk bergerak pada *state* dengan nilai yang sama, hingga batas jumlah gerakan yang ditentukan, sebelum memutuskan untuk berhenti. Karena itu, durasi pencarian juga bergantung pada jumlah iterasi maksimal yang diizinkan untuk berpindah dalam level yang sama (*sideways*). Jika iterasi *sideways* dibatasi sedikit, waktu pencarian akan lebih cepat tetapi kemungkinan besar algoritma berhenti di *local optima*. Namun, jika batas iterasi *sideways* lebih besar, waktu pencarian akan bertambah, dan peluang mencapai solusi lebih optimal meningkat. Hasil akhir dari algoritma ini akan selalu konsisten jika *state* awal kubus sama, namun karena *state* awal kubus acak, durasi dan *objective function* yang didapatkan juga beragam. Dari eksperimen yang dilakukan, algoritma ini bekerja cukup baik dengan rata-rata *objective function* sekitar +-700 (nilai maks *objective function* = 0).

3. Random Restart Hill-climbing

Random Restart Hill Climbing memiliki cara kerja seperti *steepest ascent hill-climbing* pada awalnya, dengan perbedaan ketika algoritma ini mencapai *local optima* dan tidak ada *state* di sekitarnya dengan nilai yang lebih baik, algoritma akan melakukan restart dari *state* baru. Proses ini berulang hingga batas iterasi yang telah ditentukan atau hingga *objective function* = 0 tercapai, apabila algoritma ini tidak menemukan *global optima* sampai batas iterasi habis, maka akan memiliki *state* dengan *objective function* paling baik. *Random restart* dapat mendekati *global optima* lebih baik dibandingkan algoritma *steepest ascent* atau *sideways move*, karena memungkinkan algoritma untuk terus melakukan pencarian baru saat mencapai optimum lokal.

Jika dibandingkan dengan algoritma *hill climbing* lainnya, durasi pencarian *random restart* akan semakin lama seiring jumlah iterasi pengulangan yang diinginkan. Namun, semakin sering pengulangan dilakukan maka peluang untuk mendapatkan *state* terbaik semakin besar. Sayangnya, semakin banyak pengulangan dilakukan maka durasi pencarian juga akan bertambah. Dari eksperimen yang dilakukan, algoritma ini memberikan hasil yang memuaskan dengan rata-rata *objective function* sekitar +500 (nilai maks *objective function* = 0).

4. Stochastic Hill-climbing

Stochastic Hill Climbing bekerja dengan memilih secara acak salah satu *neighbor state* dan berpindah ke *state* tersebut jika *objective function*-nya lebih tinggi. Dalam setiap iterasi, tidak semua *neighbor state* diperiksa untuk menemukan yang terbaik, tetapi dipilih secara acak. Sehingga algoritma ini dapat menghindari jebakan *local optima* dan mendekati *global optima* (faktor keberuntungan) karena tidak berfokus pada pencarian dengan nilai tertinggi di setiap langkah.

Untuk hasil pencarian dan durasi dari algoritma ini bergantung kepada jumlah iterasi yang ingin dilakukan. Semakin banyak iterasi yang dilakukan maka hasil pencarinya akan semakin baik, namun durasi pencarian tentunya akan meningkat. Kegunaan *Stochastic hill-climbing* sendiri adalah menawarkan durasi yang singkat dan hasil yang cukup baik, dapat dilihat pada eksperimen dimana algoritma ini hanya membutuhkan waktu sekitar 1 detik untuk menghasilkan solusi dengan *objective function* rata-rata +1000.

5. Simulated Annealing

Hasil eksperimen dari algoritma *simulated annealing* sangatlah beragam, ada yang menghasilkan *objective function* kurang baik (4581), ada juga yang menghasilkan *objective function* memuaskan (534). Sehingga dapat kita lihat bahwa ada banyak faktor yang menentukan *objective function* yang akan dihasilkan algoritma ini. Seperti suhu awal dan koefisien penurunan suhu yang akan menentukan banyaknya iterasi yang akan dilakukan, juga batas probabilitas yang akan menentukan penerimaan *state* bernilai lebih rendah. Kedua hal itu bukan hanya mempengaruhi *objective function* tetapi juga durasi pencarian, karena semakin banyaknya iterasi yang dilakukan maka akan semakin lama juga durasi pencarian. Sehingga perlunya disesuaikan konfigurasi input pada algoritma untuk mencapai hasil yang diinginkan.

6. Genetic Algorithm

Hasil eksperimen dari algoritma *genetic algorithm* sangat bergantung dari parameter yang dimilikinya yaitu jumlah populasi dan jumlah iterasi maksimal. Jumlah populasi mempengaruhi keragaman *parent* yang akan berdampak pada pembentukan *child*. Populasi yang terbatas akan menyebabkan kemungkinan individu sama terpilih dalam *spin wheel* meningkat. Jika individu sama terpilih menjadi *parent* dan jika individu tersebut terpilih untuk melakukan *crossover*, maka *child* yang dihasilkan akan sama dengan *parentnya* dan seterusnya untuk iterasi berikutnya.

Namun dengan mutasi, meski dengan peluang yang sangat kecil, hal tersebut dapat diatasi sedikit dengan melakukan pertukaran secara acak.

Jumlah Iterasi memengaruhi pencarian nilai objektif. Jika jumlah iterasi tidak diiringi dengan keragaman populasi maka iterasi yang banyak pun akan menjadi tidak efektif karena setiap iterasi hanya mengulangi pemilihan individu yang sama. Namun, ketika populasi beragam, maka jumlah iterasi yang banyak akan meningkatkan peluang dalam mendapatkan nilai objektif yang lebih baik.

Secara keseluruhan algoritma ini merupakan algoritma yang lambat karena banyak proses yang harus dilalui dalam satu iterasi dan hasil akhir dari algoritma ini pun tidak pasti tergantung dari hasil *random*. Baik dari metode *crossover* maupun mutasi yang dilakukan, semuanya bergantung dari hasil pengacakan.

Perbandingan

1. Kedekatan Hasil Pencarian dengan Global Optima

Dari seluruh eksperimen yang dilakukan, kami mendapatkan bahwa algoritma *random restart hill climbing* memberikan nilai *objective function* paling mendekati 0, dengan rata-rata hasil *objective function* sekitar 500 (berdasarkan hasil plot), angka ini masih dapat ditingkatkan seiring naiknya jumlah iterasi pengulangan yang ingin dilakukan. Untuk algoritma dengan ketepatan paling rendah adalah *genetic algorithm*, dapat dilihat dari semua hasil eksperimen *genetic algorithm* memiliki *objective function* akhir sekitar 4000 sampai 5000, walaupun sudah memakan durasi yang lama dalam proses pencarinya. Terdapat juga algoritma *simulated annealing* yang memiliki hasil beragam sesuai dengan konfigurasi input yang diinginkan, yaitu berdasarkan banyaknya iterasi (menentukan suhu awal dan koefisien penurunan suhu) dan juga penentuan batas probabilitas.

2. Durasi Proses Pencarian

Untuk durasi proses pencarian tiap-tiap algoritma, Penghasil solusi dengan durasi tercepat pada eksperimen yang kami lakukan adalah algoritma *stochastic hill climbing* dengan durasi proses pencarian hanya sekitar 1 detik untuk 5000 kali iterasi. Disusul oleh *Steepest Ascent hill-climbing* dengan durasi proses pencarian rata-rata dibawah 100 detik. Alasan algoritma *Stochastic* memiliki durasi yang sangat singkat adalah karena algoritma ini tidak melakukan pencarian *best neighbor* seperti algoritma *hill climbing* lainnya, melainkan hanya melakukan pertukaran random dengan *neighbor* disekitarnya tanpa melihat nilai *neighbor* tersebut. Untuk *steepest ascent*, algoritma ini memiliki durasi yang

lebih singkat dibandingkan algoritma seperti *sideways move* dan *random restart* karena *steepest ascent* tidak mampu untuk berpindah ke *state* dengan nilai yang lebih rendah / sama darinya, dibandingkan dengan *random restart* yang melakukan pengulangan iterasi dari awal atau *sideways move* yang mampu berpindah ke nilai yang sama.

3. Konsistensi Hasil Akhir

Untuk algoritma *Hill Climbing*, selain *Stochastic* kami mendapatkan hasil yang cukup konsisten untuk setiap percobaan yang kami lakukan untuk masing-masing algoritma. Meski dengan *state* awal yang berbeda, hasil akhir yang didapat memang berbeda akan tetapi dalam rentang yang tidak jauh sehingga dapat disimpulkan bahwa hasil yang didapat cukup konsisten. Hal tersebut dikarenakan algoritma ini selalu mencari *state* tetannga terbaik yang mungkin, terutama untuk *random restart* yang akan selalu mengambil solusi terbaik dari percobaan-percobaan yang dilakukan. Sedangkan untuk Simulated Annealing dan Genetic Algorithm, dan Stochastic, karena *neighbor state* *digenerate* secara acak, maka hasil yang didapat tidak konsisten. Selain itu konsistensi pada ketiga algoritma tersebut juga dipengaruhi oleh banyaknya iterasi yang dilakukan.

4. Pengaruh Banyak Iterasi dan Jumlah Populasi Genetic Algorithm

Semakin sedikit populasi, semakin besar peluang individu yang terpilih menjadi *parent* merupakan individu yang sama karena metode *spin wheel* menggunakan jumlah populasi sebagai basis pemilihan *parent* sehingga semakin sedikit kandidat *parent*, semakin besar peluang terpilihnya indiividu yang sama untuk menjadi *parent*. Hal tersebut mengakibatkan *child* hasil crossover akan memberikan hasil yang sama dengan *parentnya* dan bisa menyebabkan ulangan terus menerus seperti gambar berikut

```
Pada iterasi ke 3, Nilai Objective dari child adalah: [6677, 6525, 6679]
Pada iterasi ke 3, Best Objective : 6145
Pada iterasi ke 4, Nilai Objective dari child adalah: [6525, 6525, 6525]
Pada iterasi ke 4, Best Objective : 6145
Pada iterasi ke 5, Nilai Objective dari child adalah: [6525, 6525, 6525]
Pada iterasi ke 5, Best Objective : 6145
Pada iterasi ke 6, Nilai Objective dari child adalah: [6525, 6525, 6525]
Pada iterasi ke 6, Best Objective : 6145
Pada iterasi ke 7, Nilai Objective dari child adalah: [6525, 6525, 6525]
Pada iterasi ke 7, Best Objective : 6145
Pada iterasi ke 8, Nilai Objective dari child adalah: [6525, 6525, 6277]
Pada iterasi ke 8, Best Objective : 6145
Pada iterasi ke 9, Nilai Objective dari child adalah: [6625, 6525, 6541]
Pada iterasi ke 9, Best Objective : 6145
```

Untuk bisa dapat keluar dari perulangan yang sama tersebut, Genetic Algorithm memiliki fungsi mutasi yang bisa melakukan pertukaran secara acak. Namun, karena peluang untuk terjadinya mutasi kecil, maka mutasi pun jarang terjadi. Sebaliknya semakin besar populasi, semakin banyak kemungkinan *parent* untuk *crossover* sehingga individu yang dihasilkan bisa beragam serta peluang untuk mendapatkan nilai objektif yang lebih baik semakin tinggi.

Pengaruh banyaknya iterasi terhadap hasil akhir yaitu semakin banyak iterasi, akan meningkatkan kemungkinan algoritma untuk mendapatkan nilai objektif yang lebih baik karena individu dengan *crossover* kedua individu diharapkan akan menghasilkan *child* yang memiliki nilai objektif yang lebih baik. Oleh karena itu, ketika dilakukan iterasi yang lebih banyak, peluang untuk mencapai nilai objektif yang lebih baik pun semakin tinggi.

IV. Kesimpulan dan Saran

IV.1 Kesimpulan

Dari Tugas Besar ini, ada beberapa hal yang dapat disimpulkan diantaranya :

Algoritma yang dihasilkan dari tugas besar kali ini yaitu algoritma *local search* yang terdiri dari Hill Climbing, Simulated Annealing dan Genetic Algorithm. Algoritma tersebut digunakan dalam mencari nilai objektif terbaik yang berupa suatu *state* pada kubus yang paling memenuhi persyaratan *magic cube* yaitu penjumlahan setiap garisnya sejumlah *magic number*.

Dari hasil eksperimen yang kami lakukan, algoritma yang kami buat bisa melakukan perubahan *state* yang membuat nilai objektif semakin baik, akan tetapi dari percobaan yang kami lakukan, kami belum pernah mencapai kondisi global optimum. Namun, hasil yang kami dapatkan sudah cukup baik dalam mengurangi nilai objektif (mengurangi *error*) dari *state* awal. Algoritma yang dapat memberikan nilai objektif terbaik tanpa mempedulikan banyaknya iterasi adalah **Simulated Annealing**. Alasannya karena terdapat beberapa algoritma untuk menghindari lokal optimum (seperti pindah ke *state* yang lebih buruk dengan probabilitas tertentu). Walaupun begitu, karena banyaknya iterasi dan durasi yang dibutuhkan untuk menghasilkan nilai objektif terbaik algoritma Simulated Annealing kurang cocok jika membutuhkan solusi yang lebih cepat tanpa memerlukan hasil paling optimal. Karena itu, algoritma Steepest Ascent Hill Climbing dapat dipilih sebagai algoritma “*all rounder*” baik dari segi durasi, iterasi, juga kedekatannya dengan *global optima*, karena algoritma hanya akan berpindah ke *state* dengan nilai lebih baik, sehingga mengurangi jumlah iterasi dan durasi tanpa mengurangi *value* dari *state* yang ditempati.

Untuk durasi pencarian tercepat yaitu Stochastic Hill Climbing jika iterasi yang dilakukan sejumlah iterasi yang dilakukan pada algoritma Hill Climbing lainnya(+100). Hal ini dikarenakan *neighbor state* dari algoritma ini di-*generate* secara acak dan tidak membangkitkan semua *successor*. Sedangkan untuk yang terlama yaitu Random Restart Hill Climbing karena algoritma ini membangkitkan semua *successor* dalam mencari *best neighbor state* serta ketika menemukan lokal optima, algoritma ini akan melakukan *restart*. *Local search* bermanfaat dalam memberikan solusi yang berfokus pada efisiensi karena hanya mencari di sekitar ruang solusi. Selain itu algoritma ini tergolong lebih cepat dibandingkan algoritma *classical search* dan lebih mudah diimplementasikan.

IV.2 Saran

- Gunakan bahasa pemrograman yang tepat agar dapat menjalankan kode dengan lebih cepat. Kami merekomendasikan bahasa pemrograman yang cepat agar ketika mengeksekusi algoritma Hill Climbing yang membangkitkan semua suksesor dapat berjalan lebih cepat.
- Dalam pembuatan *magic cube*, gunakan struktur matrix 3D agar lebih mempermudah pendefinisian *magic cube*. Ketika struktur kubus hanya didefinisikan dalam array,

bentuknya tidak akan terlalu terbayang dan akan cukup sulit dalam menentukan *objective function*.

- Usahakan visualisasi dalam bentuk 3D sehingga pengguna dapat lebih terbayang. Hal tersebut memungkinkan pengguna untuk bisa mengetahui tata letak dan struktur dari kubus (termasuk letak elemen-elemennya) dengan lebih jelas.
- Pilih *schedule* yang lebih *simple* (bisa dilakukan dengan pengurangan atau pengalian terhadap *coolingRate*). Tujuannya agar lebih mudah dilakukan analisis untuk setiap iterasi.
- Buat plot dari nilai objektif suatu *state* untuk setiap iterasi agar dapat lebih mudah diamati dan dianalisis serta ditarik kesimpulan dari algoritma yang dijalankan.

V. Pembagian Tugas

NIM	Nama	Kegiatan
18222022	Louis Ferdyo Gunawan	<ol style="list-style-type: none"> 1. Membuat steepest 2. Membuat visualisasi 2 Dimensi 3. Membuat laporan bagian eksperimen & analisis pada hasil eksperimen 4. Membuat kesimpulan
18222028	Erwan Poltak Halomoan	<ol style="list-style-type: none"> 1. Membuat Visualisasi 3 Dimensi 2. Membuat Stochastic Algorithm 3. Melakukan debug code (terutama pembentukan cube, state, dan objective function) 4. Membantu testing dan laporan bagian Genetic Algorithm
18222064	Hartanto Luwis	<ol style="list-style-type: none"> 1. Membuat fungsi-fungsi dasar spt makeCube, random, dst 2. Membuat Simulated Annealing 3. Membuat Genetic Algorithm 4. Membuat detail fungsi dan penjelasannya di laporan 5. Membuat kesimpulan n saran
18222066	Ammar Naufal	<ol style="list-style-type: none"> 1. Membuat Sideways Move 2. Membuat Random Restart 3. Membuat laporan bagian hasil eksperimen, membantu kesimpulan dan saran, serta merapikan laporan secara keseluruhan 4. Membantu debug dan revisi typography kode

VI. Referensi

- [Magic Cube - Wolfram Mathworld](#)
- [Features of the magic cube - Magisch vierkant](#)
- [Perfect Magic Cubes \(trump.de\)](#)
- [Magic cube - Wikipedia](#)
- [Local Search Algorithm in Artificial Intelligence - GeeksforGeeks](#)
- [The Relation Between Complete and Incomplete Search](#)