

**Tugas Besar 1 IF3070**  
**Dasar Inteligensi Artifisial**  
**Pencarian Solusi Diagonal Magic Cube dengan Local Search**



Disusun oleh kelompok 25 :  
M. Fadhil Atha /18221003  
Nicholas Francis Aditjandra / 18221005  
M. Rivaldi Mahari /18222119  
Regan Adhiesta Mahendra /18222122

**PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2024**

## BAB I

### Deskripsi Persoalan

67	18	119	106	5
116	17	14	73	95
40	50	81	65	79
56	120	55	49	35
36	110	46	22	101

Diagonal Magic Cube adalah sebuah kubus yang terdiri dari angka-angka 1 hingga tanpa pengulangan, dengan  $n$  adalah panjang sisi kubus. Angka-angka dalam kubus disusun sedemikian rupa sehingga memenuhi beberapa properti. Beberapa syarat yang harus dipenuhi adalah:

- Terdapat satu angka yang disebut sebagai magic number dari kubus tersebut (magic number tidak harus berada dalam rentang angka 1 hingga  $n^3$ , dan juga bukan merupakan angka yang perlu dimasukkan ke dalam kubus)
- Jumlah angka-angka pada setiap baris harus sama dengan magic number.
- Jumlah angka-angka pada setiap kolom juga harus sama dengan magic number.
- Jumlah angka-angka pada setiap tiang harus sama dengan magic number
- Jumlah angka-angka pada setiap diagonal ruang kubus juga sama dengan magic number
- Jumlah angka-angka pada diagonal-diagonal di setiap potongan bidang dari kubus harus sama dengan magic number.

Sebagai contoh, untuk kubus berukuran  $5 \times 5 \times 5$ , ada 15 potongan bidang. Pada setiap potongan ini, diagonal-diagonal (dilingkari merah dalam ilustrasi) juga harus memenuhi syarat jumlah yang sama dengan magic number.

Tujuan dari akhir dari penyelesaian permasalahan ini adalah untuk menemukan solusi terbaik dari algoritma-algoritma mana yang paling optimal dalam menyelesaikan permasalahan tersebut. Kemudian mendapatkan jawaban perbandingan, kelebihan, dan kelemahan dari setiap algoritma.

## BAB II

### Pembahasan

#### A. Objective Function

Objective function digunakan untuk menentukan nilai dari suatu state pada *magic cube*, yaitu untuk menentukan seberapa dekat state tersebut dengan solusi yang diinginkan. Magic cube memiliki suatu magic number yang merupakan nilai yang harus dipenuhi untuk total nilai setiap baris, kolom, tiang, diagonal ruang, dan diagonal bidang. Magic number didapatkan dengan rumus sebagai berikut.

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

Pada Magic cube 5x5x5, objective function dirumuskan dari penjumlahan total nilai dari setiap baris, kolom, tiang, diagonal ruang, dan diagonal bidang yang telah dikurangi dengan magic number. Setelah dijumlahkan, diff dihitung sebagai selisih antara magic number ( $M(n)$ ) dengan hasil penjumlahan sebelumnya. Hasil yang diinginkan adalah diff = 109 yang berarti konfigurasi tersebut telah mencapai solusi optimal.

$$diff = \sum_{b=1}^5 SumB + \sum_{c=1}^5 SumK + \sum_{t=1}^5 SumT + SumDr + \sum_{k=1}^5 SumDb$$

- SumB = jumlah untuk setiap baris yang bergerak melalui lima kolom (indeks j) untuk menjumlahkan semua elemen dalam baris ke-r ( $MagicCube[r][j][k]$ )
- SumK = jumlah untuk setiap baris yang bergerak melalui lima baris (indeks i) untuk menjumlahkan semua elemen dalam kolom ke-c ( $MagicCube[i][c][k]$ )
- SumT = jumlah untuk setiap baris yang bergerak melalui lima baris (indeks i) untuk menjumlahkan semua elemen dalam kolom ke-t ( $MagicCube[i][j][t]$ )
- SumDr = jumlah untuk setiap diagonal ruang
- SumDb = jumlah untuk setiap diagonal bidang

Note : masing - masing Sum diatas dikurangi dengan magic number(315) dan dimutlakkan. Semakin mendekati 109 atau semakin nilainya maka akan semakin mendekati solusi optimal

#### B. Algoritma Local Search

Algoritma local search adalah algoritma yang mencari solusi dari suatu masalah dengan berpindah dari *start state* ke *neighboring states* tanpa menyimpan keseluruhan *path* yang telah diambil atau himpunan *states* yang telah dicapai. Local search memiliki tujuan untuk mencari puncak tertinggi (*global maximum*).

Namun, algoritma ini memiliki kelemahan yaitu berpotensi terjebak di puncak lokal (*local maximum*) daripada menemukan solusi optimal (*global maximum*).

#### a. Steepest Ascent Hill-Climbing

Algoritma search yang menyimpan state saat ini dan pada setiap iterasi akan berpindah ke neighboring state yang memiliki nilai objective function tertinggi. Algoritma ini akan berhenti saat mencapai puncak dimana tidak ada lagi neighbors yang memiliki nilai lebih tinggi daripada current state. Dalam penyelesaian magic cube proses pencarian dengan algoritma dapat dilakukan sebagai berikut.

Komponen-komponen fungsi :

- `objectivefunction(current_list)`, fungsi menghitung nilai objektif.
- `highestneighbor(current_list)`, fungsi yang mengembalikan solusi terbaik.
- `Max_iteration` dan `max_duration`, variabel yang mengatur jumlah iterasi dan durasi maksimum.

Proses :

1. Kandidat awal solusi disimpan dalam `current_list` dan nilai solusi berdasarkan fungsi objektif dihitung dan disimpan dalam `current_value`
2. Algoritma melakukan iterasi untuk mencari solusi terbaik hingga mencapai jumlah iterasi maksimum (`max_iteration`) atau waktu maksimum yang telah ditetapkan (`max_duration`)
3. Pada setiap iterasi, waktu yang telah berlalu diperbarui. Jika waktu ini melebihi `max_duration`, iterasi akan dihentikan lebih awal.
  - a. `neighbor_list` diisi dengan tetangga terbaik dari `current_list`, yang diperoleh menggunakan `highestneighbor(current_list)`
  - b. Jika nilai dari tetangga ini lebih baik daripada `current_value`, solusi kemudian diperbarui menjadi tetangga ini, dan iterasi dilanjutkan.
  - c. Jika tidak ada tetangga yang lebih baik ditemukan, maka iterasi dihentikan, yang menunjukkan bahwa solusi sudah mencapai maksimum lokal.
4. Fungsi kemudian akan mengembalikan `current_list`, yang dianalisis sebagai solusi terbaik yang telah ditemukan dan berhenti ketika mencapai maksimum lokal atau batas waktu atau jumlah iterasi tercapai.

Fungsi `SteepestAscent_hill_climbing(list)`

```
def SteepestAscent_hill_climbing(list):
    current_list = list[:]
    current_value = objectivefunction(current_list)
    i = 0
    start = datetime.datetime.now()
    end = 0
    print(i, " : ", current_value)
    while i < max_iteration:
```

```

plotarray.append(current_value)
neighbor_list = highestneighbor(current_list)
if current_value < objectivefunction(neighbor_list):
    current_value = objectivefunction(neighbor_list)
    current_list = neighbor_list
    i += 1
    print(i, " : ", current_value)
else:
    break
end = (datetime.datetime.now() - start).seconds
if end >= max_duration:
    break
return current_list

```

### b. Hill-climbing with Sideways Move

Algoritma search yang menyimpan state saat ini dan pada setiap iterasi akan berpindah ke neighboring state yang memiliki nilai tertinggi. Perbedaan algoritma ini dengan steepest ascent hill-climbing adalah algoritma ini masih akan berjalan jika masih ada neighboring state yang memiliki nilai yang sama dengan state saat ini. Hal ini digunakan untuk menghindari terjebak di dataran (*plateau*) Algoritma ini baru akan berhenti ketika semua neighboring state nya bernilai lebih kecil. Dalam penyelesaian magic cube proses pencarian dapat dilakukan sebagai berikut.

Komponen-komponen fungsi:

- objectivefunction(current\_list), fungsi menghitung nilai objektif.
- highestneighbor(current\_list), fungsi yang mengembalikan solusi terbaik.
- Max\_sidewaymove, variabel yang mengatur jumlah maksimal toleransi solusi terbaik dengan nilai yang sama dengan nilai sekarang
- Max\_iteration dan max\_duration, variabel yang mengatur jumlah iterasi dan durasi maksimum.

Proses:

1. Kandidat awal solusi disimpan dalam current\_list dan nilai solusi berdasarkan fungsi objektif dihitung dan disimpan dalam current\_value
2. Jumlah toleransi solusi terbaik dengan nilai yang sama dengan current\_list disimpan dalam sidewaymove yang diinisialisasi dengan nilai 0
3. Algoritma melakukan iterasi untuk mencari solusi terbaik hingga mencapai jumlah iterasi maksimum (max\_iteration) atau waktu maksimum yang telah ditetapkan (max\_duration)

4. Pada setiap iterasi, waktu yang telah berlalu diperbarui. Jika waktu ini melebihi `max_duration`, iterasi akan dihentikan lebih awal.
  - a. `neighbor_list` diisi dengan tetangga terbaik dari `current_list`, yang diperoleh menggunakan `highestneighbor(current_list)`
  - b. Jika nilai dari tetangga ini lebih baik daripada `current_value`, solusi kemudian diperbarui menjadi tetangga ini, dan iterasi dilanjutkan.
  - c. Jika nilai dari tetangga ini sama dengan `current_value` selama `sidewaymove` lebih kecil dari `max_sidewaymove`, solusi kemudian diperbarui menjadi tetangga ini, `sidewaymove` ditambah 1, dan iterasi dilanjutkan,
  - d. Jika tidak ada tetangga yang lebih baik atau tetangga dengan nilai yang sama tetapi `sidewaymove` telah mencapai `max_sidewaymove` ditemukan, maka iterasi dihentikan, yang menunjukkan bahwa solusi sudah mencapai maksimum lokal.

Fungsi:

```
def SidewayMove_hill_climbing(list):
    current_list = list[:]
    current_value = objectivefunction(current_list)
    i = 0
    sidewaymove = 0
    start = datetime.datetime.now()
    end = 0
    print(i, " : ", current_value)
    while i < max_iteration:
        plotarray.append(current_value)
        neighbor_list = highestneighbor(current_list)
        if current_value < objectivefunction(neighbor_list):
            current_value = objectivefunction(neighbor_list)
            current_list = neighbor_list
            i += 1
            print(i, " : ", current_value)
        elif (current_value == objectivefunction(neighbor_list)) and (sidewaymove < max_sidewaymove):
            sidewaymove += 1
            current_value = objectivefunction(neighbor_list)
            current_list = neighbor_list
            i += 1
            print(i, " : ", current_value)
        else:
            break
    end = (datetime.datetime.now() - start).seconds
    if end >= max_duration:
        break
```

```
return current_list
```

### c. Random Restart Hill-Climbing

Algoritma search yang terdiri dari serangkaian hill-climbing yang masing-masing initial state nya acak. Ketika suatu hill-climbing gagal mencapai solusi optimal maka algoritma akan melakukan restart dari state random yang berbeda. Hal ini meningkatkan peluang untuk menemukan *global maximum* karena state awal yang dipilih secara acak dapat membantu keluar dari jebakan *local maximum*.

Dalam penyelesaian magic cube proses pencarian dengan algoritma ini sama dengan Steepest Ascent Hill-Climbing, random restart hill-climbing merupakan kumpulan iterasi dari algoritma steepest ascent hill-climbing dengan tujuan diff yang sama dengan 109. Perbedaan 2 algoritma ini adalah steepest ascent hill-climbing penghentiannya belum tentu mencapai *global maximum* sedangkan algoritma random restart hill-climbing penghentiannya sampai dengan *global maximum*.

Komponen-komponen fungsi:

- initial(), fungsi untuk menghasilkan solusi awal acak.
- objectivefunction(list), fungsi untuk menghitung nilai objektif
- Steepestascent\_hill\_climbing, sidewaymove\_hill\_climbing, dan stochastic\_hill\_climbing
- Display\_3d\_cube
- Plotarray, array untuk menyimpan dan memplot skor selama proses.

Proses :

1. Fungsi dimulai dengan mencatat waktu mulai (start) dan inisialisasi perhitungan restart ke 0. Algoritma ini akan mengulangi proses hingga jumlah maksimum restart(max\_restart) tercapai.
2. Setiap kali restart:
  - a. Sebuah solusi awal yang diacak (initialList) dihasilkan menggunakan fungsi initial().
  - b. Nilai objektif dari solusi awal ini dihitung menggunakan objectivefunction(initialList).
  - c. Solusi awal akan disimpan dalam finalList sebagai solusi terbaik yang ditemukan pada restart ini.
3. Berdasarkan nilai variabel hill\_climbing, fungsi kemudian akan memilih metode hill climbing yang akan digunakan:
  - a. 0 : steepestascent\_hill\_climbing

- b. 1 : sidewaymove\_hill\_climbing
- c. 2 : stochastic\_hill\_climbing

Solusi akhir dari metode yang telah dipilih akan disimpan dalam tempList. Jika nilai tempList lebih baik daripada nilai finalList, maka finalList diperbarui menjadi tempList.

4. Setelah semua restart selesai, nilai objektif dari finalList dihitung dan dicetak sebagai skor akhir. Kemudian durasi total dari seluruh proses dicetak.
5. Fungsi membuat plot untuk menunjukkan perkembangan skor selama proses dengan data dalam plotarray.

```
def randomrestart_hill_climbing():
    start = datetime.datetime.now()
    restart = 0

    while restart <= max_restart:
        print("Restart ke : ", restart)
        initialList = initial()
        a = objectivefunction(initialList)
        print("CURRENT SCORE " + str(a))
        print(initialList)
        plotrestart.append(initialList)

        finalList = initialList[:]
        if hill_climbing == 0:
            tempList = steepestascent_hill_climbing(initialList)
        elif hill_climbing == 1:
            tempList = sidewaymove_hill_climbing(initialList)
        elif hill_climbing == 2:
            tempList = stochastic_hill_climbing(initialList)
        print("Total iterasi : ", i)
        plotiterasi.append(i)
        restart += 1
        if objectivefunction(tempList) > objectivefunction(finalList):
            finalList = tempList[:]

    print("Plot iterasi per Restart")
    print(plotiterasi)
    print("Plot Restart")
    print(plotrestart)
    b = objectivefunction(finalList)
    print("FINAL SCORE " + str(b))
    print(finalList)
```



```

end = (datetime.datetime.now() - start)
print("Total waktu : ", end)

initial_array = np.array(initialList).reshape((5, 5, 5))
display_3d_cube(initial_array, 2)
final_array = np.array(finalList).reshape((5, 5, 5))
display_3d_cube(final_array, 1)
plt.plot(plotarray)
plt.show()

```

#### d. Stochastic Hill-Climbing

Algoritma search yang menyimpan state saat ini dan pada setiap iterasi akan berpindah ke neighboring state yang lebih tinggi secara acak. Algoritma ini akan berhenti setelah melakukan iterasi sebanyak nilai yang telah ditentukan sebelumnya.

Dalam penyelesaian magic cube proses pencarian dengan algoritma ini sama seperti algoritma Steepest Ascent Hill-Climbing hanya saja pada algoritma stochastic hill-climbing namun pada setiap iterasi, algoritma ini akan memilih satu tetangga secara acak. Jika nilai dari tetangga tersebut lebih buruk dibandingkan kondisi saat ini (*current state*), algoritma akan mengabaikannya dan melanjutkan pencarian ke neighbor lain. Namun, jika nilai neighbor tersebut lebih baik dibandingkan kondisi saat ini, algoritma akan berpindah ke tetangga tersebut dan terus melanjutkan proses pencarian. Proses ini berulang sampai tidak ada neighbor lain yang lebih baik. Algoritma akan berhenti ketika sudah melewati durasi. Berbeda dari *Steepest Ascent Hill-Climbing*, yang memeriksa semua neighbor sebelum memilih yang terbaik, *stochastic hill-climbing* hanya berfokus pada evaluasi acak dan melanjutkan pencarian jika menemukan solusi yang lebih baik.

Komponen:

- `objectivefunction(current_list)`, fungsi menghitung nilai objektif.
- `neighbor(current_list)`, fungsi yang mengembalikan state tetangga secara acak
- `Max_duration`, variabel yang mengatur durasi maksimum.

Proses:

1. Kandidat awal solusi disimpan dalam `current_list` dan nilai solusi berdasarkan fungsi objektif dihitung dan disimpan dalam `current_value`

2. Algoritma melakukan iterasi untuk mencari solusi terbaik hingga mencapai waktu maksimum yang telah ditetapkan (`max_duration`)
3. Pada setiap iterasi, waktu yang telah berlalu diperbarui. Jika waktu ini melebihi `max_duration`, iterasi akan dihentikan lebih awal.
  - a. Akan dicari `neighbor_list` yang merupakan state tetangganya secara acak dengan nilai yang akan disimpan ke dalam `neighbor_value`
  - b. Jika nilai dari tetangga ini lebih baik daripada `current_value`, solusi kemudian diperbarui menjadi tetangga ini, dan iterasi dilanjutkan.
  - c. Jika nilai dari tetangga ini kurang dari atau sama dengan `current_value` iterasi akan diulang,

Fungsi:

```
def stochastic_hill_climbing(list):  
    global i  
    i = 0  
    current_list = list[:]  
    current_value = objectivefunction(current_list)  
    i = 0  
    start = datetime.datetime.now()  
    end = 0  
    while True:  
        plotarray.append(current_value)  
        neighbor_list = neighbor(current_list)  
        neighbor_value = objectivefunction(neighbor_list)  
        if neighbor_value > current_value:  
            current_list = neighbor_list  
            current_value = objectivefunction(current_list)  
        end = (datetime.datetime.now() - start).seconds  
        i += 1  
        print(i, " : ", current_value)  
  
        if end >= max_duration:  
            break  
    return current_list
```

#### e. Simulated Annealing

Algoritma search yang menyimpan state saat ini dan pada setiap iterasi akan berpindah ke neighboring state yang memiliki nilai yang lebih tinggi. Algoritma ini memperbolehkan perpindahan ke neighbor state yang lebih buruk tapi frekuensi perpindahan ke state yang lebih buruk ini awalnya akan jarang dan semakin sering terutama apabila terjebak di suatu state. Neighbor state yang lebih kecil akan dipilih sesuai dengan probabilitas  $e^{\Delta E/T}$  dengan T adalah “temperatur” yang perlahan berkurang dan  $\Delta E = \text{neighbor.value} - \text{current.value}$ .

- Jika neighbor lebih baik ( $\Delta E > 0$ ) maka pindahkan current state ke neighbor tersebut.
- Jika neighbor lebih buruk ( $\Delta E < 0$ ) maka probabilitas untuk pindah adalah  $e^{\Delta E/T}$

Dalam penyelesaian magic cube proses pencarian dengan algoritma ini hampir sama dengan algoritma Stochastic Hill-Climbing, algoritma ini dapat memilih neighboring state yang memiliki nilai lebih besar, namun masih dapat melanjutkan pencarian dengan probabilitas tertentu.

Proses:

1. Fungsi menerima input berupa list yang dimasukan ke dalam current\_list, kemudian nilai objektif dari solusi awal dimasukan ke dalam current\_value yang dihitung menggunakan objectivefunction(current\_list) dan temprature dimasukan ke dalam max\_temprature.
2. Jika (while true) terpenuhi maka loop akan terus berjalan sampai kondisi penghetian berupa waktu dari batas durasi max\_duration.
3. Pada setiap iterasi, algoritma memilih tetangga acak(neighbor\_list) dari current\_list dan menghitung nilai objektifnya(neighbor\_value)
4. Jika solusi yang diberikan dari tetangga lebih baik (neighbor\_value > current\_value), maka nilai dari tetangga akan diterima tanpa syarat, dan setelah itu current\_list serta current\_value diperbarui dengan solusi dari tetangga tersebut.
5. Jika neighbor\_value < current\_value maka algoritma akan memutuskan apakah solusi ini diterima dengan perhitungan probabilitas annealing melalui langkah.
6. Kemudian nilai acak chance akan dicari dengan nilai antara 0 dan 1 (sampai 5 angka dibelakang koma) yang akan dibandingkan dengan probabilitas annealing. Jika chance lebih besar dari annealing, solusi lebih buruk akan diterima dimana current\_list serta current\_value diperbarui dengan solusi dari tetangga tersebut, kemudian stuck akan dikonfirmasi dengan bertambah nilai.
7. Apabila chance lebih kecil dari annealing, temperature akan turun untuk meningkatkan peluang menerima solusi buruk terutama ketika mengalami stuck.
8. Jika waktu sudah mencapai max\_duration, loop akan dihentikan.
9. Fungsi akan mengembalikan solusi akhir (current\_list) beserta nilai akhirnya (current\_value)

Fungsi :

```

def Simulated_Annealing(list):
    global stuck
    global i
    i = 0
    current_list = list[:]
    current_value = objectivefunction(current_list)

    start = datetime.datetime.now()
    end = 0
    while True:
        print(i, " : ", current_value)
        plotarray.append(current_value)
        neighbor_list = neighbor(current_list)
        neighbor_value = objectivefunction(neighbor_list)

        if neighbor_value > current_value:
            current_list = neighbor_list
            current_value = objectivefunction(current_list)
        elif neighbor_value < current_value:
            if temperature != 0:
                annealing = math.exp(
                    (neighbor_value - current_value) / temperature)
                temperaturplot.append(annealing)
                temperaturploti.append(i)
                chance = random.randint(0, 100000)
                if chance != 0:
                    chance = chance/100000
                if chance >= annealing:
                    current_list = neighbor_list
                    current_value = objectivefunction(current_list)
                    temperature = max_temperature
                    stuck += 1
                else :
                    temperature -= 1
            else:
                truth = 1
        i += 1
    end = (datetime.datetime.now() - start).seconds
    if end >= max_duration:

```

```
break
```

```
print(i, " : ", current_value)
```

```
return current_list
```

## f. Genetic Algorithm

Algoritma ini mengambil metapora dari seleksi natural dalam biologi. Dari suatu populasi individuals (states), algoritma ini akan mencari individu yang paling fit (highest value) untuk memproduksi anak (successor states) yang akan menciptakan generasi baru. Setiap individu merupakan sebuah string yang terdiri sequence of number dari seperti DNA. Genetic algorithm dibagi menjadi 3 tahap :

1. Selection : Proses untuk menyeleksi individu yang akan dijadikan sebagai parents generasi berikutnya.
2. Crossover : Memilih crossover point secara acak untuk memisahkan string dari masing-masing parents lalu menggabungkan kembali untuk membentuk anak yang baru, satu dengan bagian awal parent 1 dan bagian akhir parent 2, dan yang lainnya sisanya.
3. Mutation : Secara acak mengubah elemen algoritma genetic yang sudah di crossover untuk membuat variasi yang lebih besar dari populasi

Dalam penyelesaian magic cube proses pencarian dengan algoritma ini dengan beberapa tahapan:

1. Populasi awal diisi dengan state acak dari magic cube
2. Solusi yang paling mendekati  $\text{diff} = 109$  akan dipilih sebagai individu terbaik, lalu proses crossover dan mutasi state baru hingga ditemukan solusi terbaik.

Proses:

1. Fungsi menerima input berupa jumlah populasi awal dan jumlah iterasi
2. Algoritma berjalan hingga batas iterasi(max\_iteration) mencapai batas maksimal(max\_duration)
3. Pada setiap iterasi, seluruh populasi dilakukan seleksi lalu setiap seleksi di lakukan crossover satu sama lain
4. Jika array memiliki angka yang sama maka akan dilakukan mutasi untuk mengubah angka tersebut
5. Hasil akan di masukan kedalam list untuk menjadi populasi yang baru

Fungsi :

```

def geneticAlgorithm():
    jumlahPopulasi = int(input("jumlahPopulasi = ")) * 2
    jumlahIterasi = int(input("jumlahIterasi = "))
    start = datetime.datetime.now()

    highest_offspring = -1
    plotarray = []
    highest_offspring_array = []

    populationList, probability = population(jumlahPopulasi)
    initialList = max(populationList, key=objectivefunction)

    for _ in range(jumlahIterasi):
        parents = selection(jumlahPopulasi, populationList, probability)
        offspringarray = []

        for x in range(0, len(parents) // 2):
            offspring1, offspring2 = crossover(parents[x], parents[x + len(parents) // 2])
            offspringarray.extend([offspring1, offspring2])

        nextoffspring = [mutation(off) for off in offspringarray]
        populationList, probability = nextPopulation(nextoffspring)

        highest_offspring_in_iteration = max(nextoffspring, key=objectivefunction)
        highest_fitness = objectivefunction(highest_offspring_in_iteration)

        if highest_fitness > highest_offspring:
            highest_offspring = highest_fitness
            highest_offspring_array = highest_offspring_in_iteration

        plotarray.append(highest_fitness)

    print("Nilai Objective function =", highest_offspring)
    print("Jumlah populasi =", jumlahPopulasi)
    print("Banyak iterasi =", jumlahIterasi)
    print("Durasi proses pencarian =", (datetime.datetime.now() - start))

    print(initialList)
    display_3d_cube(np.array(initialList).reshape((5, 5, 5)), end=2)
    print(highest_offspring_array)
    display_3d_cube(np.array(highest_offspring_array).reshape((5, 5, 5)), end=1)

    plt.plot(plotarray)
    plt.title("Objective Function vs. Iterations")
    plt.show()

```

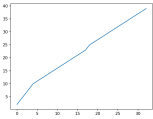
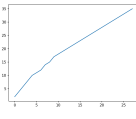
### C. Eksperimen dan Analisis

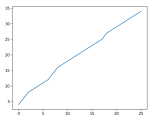
Pada bagian ini dilakukan eksperimen dengan melakukan 3 kali percobaan pada setiap algoritma yang telah dibuat.

#### a. Steepest Ascent Hill-Climbing

Pada bagian ini ditunjukkan hasil dari percobaan eksperimen Steepest Ascent Hill-Climbing yang telah dilakukan.

Percobaan ke-	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Banyak iterasi hingga berhenti	Durasi

1	[31, 44, 82, 58, 19, 62, 67, 68, 105, 123, 17, 112, 6, 9, 60, 47, 15, 11, 23, 90, 97, 26, 116, 49, 61, 98, 91, 32, 111, 100, 106, 28, 75, 53, 35, 101, 48, 85, 50, 43, 46, 38, 83, 40, 69, 84, 10, 99, 45, 80, 42, 2, 55, 5, 25, 37, 74, 66, 54, 96, 30, 108, 13, 119, 51, 63, 1, 115, 41, 102, 88, 79, 121, 18, 39, 95, 86, 33, 14, 72, 120, 93, 118, 89, 78, 113, 36, 71, 7, 87, 8, 107, 114, 52, 21, 12, 64, 77, 20, 103, 122, 16, 110, 65, 22, 70, 92, 29, 57, 34, 125, 124, 27, 56, 104, 117, 94, 109, 76, 59, 73, 4, 81, 3, 24]	[92, 7, 82, 91, 19, 62, 109, 59, 105, 123, 17, 112, 6, 9, 60, 47, 15, 11, 61, 90, 97, 26, 122, 49, 23, 98, 58, 32, 111, 100, 106, 28, 75, 10, 35, 101, 48, 85, 50, 31, 30, 38, 83, 99, 69, 84, 53, 40, 45, 80, 74, 107, 55, 5, 25, 44, 42, 66, 54, 96, 46, 86, 119, 13, 51, 63, 1, 110, 39, 102, 88, 79, 121, 18, 41, 95, 108, 33, 114, 117, 120, 93, 118, 89, 78, 113, 36, 71, 8, 87, 37, 2, 16, 52, 29, 12, 76, 77, 20, 4, 22, 72, 115, 65, 94, 124, 43, 21, 57, 70, 125, 3, 27, 56, 104, 14, 116, 67, 64, 68, 34, 81, 103, 73, 24]	39		33	0:01:37.001204
2	[45, 3, 57, 6, 2, 44, 42, 43, 12, 98, 109, 90, 99, 49, 84, 104, 4, 27, 101, 75, 10, 1, 92, 123, 114, 119, 72, 108, 61, 67, 21, 40, 47, 73, 122, 112, 24, 41, 11, 48, 55, 96, 35,	[45, 3, 57, 6, 2, 44, 42, 43, 86, 98, 109, 90, 99, 49, 97, 104, 4, 24, 15, 75, 10, 102, 92, 123, 114, 119, 72, 108, 61, 112, 21,	35		28	0:02:01.468347

	14, 74, 53, 54, 25, 5, 28, 23, 117, 80, 66, 77, 7, 70, 118, 50, 91, 124, 32, 58, 9, 89, 76, 15, 86, 106, 20, 29, 52, 94, 34, 107, 100, 85, 97, 116, 83, 102, 19, 51, 37, 115, 81, 111, 103, 26, 13, 88, 36, 69, 64, 105, 93, 65, 113, 60, 121, 22, 18, 30, 31, 8, 59, 63, 82, 33, 71, 95, 62, 110, 79, 68, 56, 87, 78, 125, 39, 16, 38, 120, 46, 17]	73, 47, 40, 122, 67, 54, 100, 11, 48, 55, 96, 35, 14, 115, 53, 1, 25, 5, 9, 23, 117, 32, 66, 77, 63, 70, 118, 50, 91, 124, 80, 58, 60, 89, 76, 101, 12, 106, 20, 29, 52, 94, 33, 107, 113, 105, 103, 116, 83, 27, 19, 51, 16, 74, 81, 111, 84, 26, 13, 88, 36, 69, 64, 85, 28, 65, 8, 93, 121, 110, 18, 30, 31, 41, 59, 7, 56, 34, 82, 38, 62, 22, 125, 68, 71, 78, 87, 79, 39, 37, 95, 120, 46, 17]				
3	[120, 31, 106, 36, 90, 71, 102, 47, 18, 59, 57, 99, 1, 108, 2, 26, 80, 52, 67, 37, 125, 77, 19, 6, 122, 60, 28, 81, 23, 14, 88, 27, 11, 15, 33, 91, 113, 4, 111, 58, 13, 46, 123, 41, 63, 53, 121, 35, 110, 24, 20, 87, 25, 124, 54, 95, 86, 17, 117, 107, 98, 85, 45, 48, 44, 8, 40, 10, 118, 93, 22, 100, 34, 114, 72, 116, 3, 104, 38, 61, 119, 78, 32, 5, 83,	[120, 31, 106, 36, 90, 71, 27, 47, 18, 22, 105, 99, 1, 108, 2, 61, 81, 52, 67, 37, 125, 77, 19, 6, 122, 102, 28, 113, 58, 14, 88, 60, 11, 15, 33, 91, 80, 87, 111, 23, 13, 46, 123, 41, 26, 53, 121, 24, 92, 25, 20, 4, 35, 124, 54, 57, 86, 110, 117, 107, 98,	34		26	0:01:15.71356 3

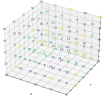
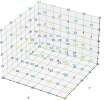
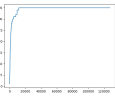


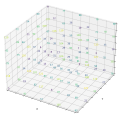
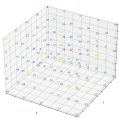
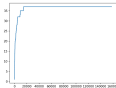
	96, 79, 109, 92, 101, 50, 62, 43, 7, 89, 30, 21, 112, 12, 105, 49, 29, 9, 42, 70, 68, 64, 75, 82, 94, 115, 74, 73, 39, 76, 66, 69, 55, 84, 65, 103, 97, 16, 51, 56]	85, 45, 48, 44, 8, 40, 56, 118, 93, 59, 100, 34, 50, 72, 116, 3, 95, 38, 63, 30, 78, 32, 5, 83, 17, 79, 109, 9, 101, 114, 62, 43, 7, 89, 66, 21, 112, 12, 104, 49, 29, 96, 42, 94, 69, 64, 75, 84, 70, 115, 74, 73, 39, 76, 119, 51, 55, 82, 65, 103, 97, 16, 68, 10]				
--	---	---	--	--	--	--



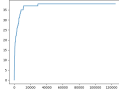
Dapat disimpulkan dari ketiga eksperimen bahwa Steepest Ascent Hill-Climbing: membutuhkan banyak waktu untuk mendapat iterasi berikutnya, sering melakukan lompatan nilai yang cukup banyak (lebih dari 1), dan mudah terjebak di local maksimum

b. Stochastic Hill-Climbing

Pada bagian ini ditunjukan hasil dari percobaan eksperimen Stochastic Hill-Climbing yang telah dilakukan.

Percobaan ke-	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Banyak iterasi hingga berhenti	Durasi
1	 [28, 25, 41, 49, 18, 93, 16, 54, 119, 15, 27, 64, 69, 86, 95, 83, 118, 20, 97, 59, 4, 8, 62, 112, 35, 125, 56, 37, 52, 45, 19, 44, 50, 85, 21, 32, 6, 78, 109, 122, 55, 36, 14, 100, 84,	 [28, 25, 17, 49, 18, 93, 16, 54, 119, 115, 27, 64, 69, 9, 10, 83, 118, 1, 97, 59, 112, 92, 62, 100, 105, 125, 56, 37, 52, 45, 8,	35		127613	0:02:00.003106

	120, 110, 79, 103, 75, 72, 96, 90, 77, 70, 46, 104, 117, 48, 99, 22, 66, 43, 51, 65, 57, 38, 11, 12, 88, 68, 42, 106, 13, 7, 71, 73, 114, 23, 58, 87, 121, 105, 2, 53, 26, 24, 1, 123, 74, 47, 107, 108, 5, 9, 89, 102, 67, 82, 61, 39, 33, 60, 80, 29, 81, 101, 63, 98, 92, 91, 111, 34, 94, 115, 113, 3, 124, 30, 10, 17, 76, 40, 31, 116]	44, 107, 96, 35, 32, 6, 78, 77, 122, 55, 99, 14, 82, 84, 120, 3, 79, 38, 75, 34, 85, 90, 109, 70, 46, 104, 65, 48, 36, 22, 19, 43, 51, 117, 67, 114, 11, 12, 88, 68, 42, 106, 95, 4, 89, 116, 29, 23, 58, 87, 121, 26, 2, 53, 21, 94, 20, 123, 57, 47, 50, 108, 24, 86, 71, 102, 74, 7, 61, 39, 33, 60, 80, 103, 81, 30, 63, 98, 111, 91, 66, 72, 5, 15, 113, 110, 124, 101, 13, 41, 76, 40, 31, 73]				
2	 [5, 118, 79, 82, 110, 91, 19, 34, 120, 64, 57, 117, 122, 25, 4, 105, 87, 111, 42, 17, 63, 69, 85, 41, 80, 96, 97, 75, 89, 72, 62, 102, 53, 37, 94, 6, 61, 47, 8, 14, 48, 59, 46, 29, 36, 21, 125, 40, 99, 30, 124, 15, 115, 11, 54, 83, 13,	 [49, 23, 5, 72, 110, 91, 19, 15, 59, 64, 47, 117, 122, 25, 4, 105, 87, 111, 115, 57, 63, 69, 62, 41, 80, 96, 29, 75, 89, 82, 85, 13, 61, 20, 94, 102, 53, 79, 8, 73, 16, 120,	37		157818	0:02:00.00365 3

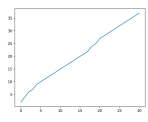
	18, 33, 109, 73, 56, 106, 104, 119, 44, 70, 81, 7, 100, 95, 114, 58, 3, 52, 45, 24, 113, 88, 60, 49, 31, 39, 116, 12, 43, 51, 90, 50, 26, 108, 112, 86, 84, 22, 27, 20, 55, 35, 92, 1, 74, 67, 107, 9, 98, 28, 32, 65, 77, 2, 38, 103, 10, 66, 23, 121, 123, 71, 78, 76, 101, 16, 93, 68]	46, 97, 36, 21, 125, 40, 99, 30, 124, 34, 92, 11, 54, 22, 6, 116, 33, 109, 14, 56, 106, 104, 119, 103, 70, 35, 7, 100, 52, 114, 58, 112, 95, 45, 24, 98, 88, 60, 17, 31, 74, 18, 71, 118, 51, 10, 26, 50, 108, 3, 86, 84, 66, 27, 43, 107, 81, 42, 1, 39, 67, 55, 9, 113, 28, 32, 65, 77, 2, 38, 44, 90, 83, 123, 121, 37, 12, 78, 76, 101, 48, 93, 68]				
3	 <p>[98, 37, 115, 84, 15, 89, 66, 125, 102, 47, 94, 44, 118, 35, 38, 51, 77, 105, 21, 34, 88, 49, 103, 20, 117, 54, 29, 19, 46, 11, 22, 1, 39, 53, 2, 73, 65, 14, 76, 45, 69, 13, 18, 26, 107, 33, 59, 99, 48, 68, 56, 85, 90, 79, 62, 55, 28, 83, 124, 95, 9, 58, 109, 100, 30, 93, 31, 61, 7, 106, 82, 116, 63, 64, 78, 32, 43, 92, 121, 8,</p>	 <p>[64, 37, 115, 84, 15, 89, 79, 125, 113, 86, 69, 96, 77, 32, 121, 104, 118, 111, 81, 34, 88, 49, 103, 16, 59, 102, 21, 19, 46, 8, 22, 1, 116, 53, 2, 73, 44, 14, 76, 108, 94, 13, 18, 68, 107, 24, 117, 99, 48, 27, 56, 98, 90, 66,</p>	38		124742	0:02:00.004926

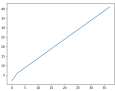
	114, 120, 40, 25, 91, 111, 12, 16, 6, 75, 119, 113, 74, 110, 86, 71, 23, 72, 27, 57, 10, 42, 108, 17, 96, 70, 50, 123, 67, 41, 24, 52, 60, 101, 4, 80, 112, 3, 36, 104, 122, 87, 97, 5, 81]	123, 20, 65, 35, 57, 95, 9, 58, 109, 100, 30, 93, 31, 61, 7, 17, 82, 39, 63, 85, 78, 83, 26, 92, 112, 11, 114, 120, 40, 25, 91, 52, 12, 55, 6, 42, 119, 72, 74, 3, 47, 71, 23, 54, 43, 124, 10, 75, 45, 106, 28, 70, 50, 62, 67, 41, 33, 105, 60, 101, 29, 80, 38, 51, 36, 110, 122, 87, 97, 5, 4]				
--	--	--	--	--	--	--

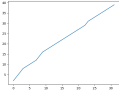
Dapat disimpulkan dari ketiga eksperimen bahwa Stochastic Hill-Climbing pada awal state dapat meningkat dengan cepat dengan melakukan banyak iterasi tetapi semakin lama nilai objektif function akan susah untuk meningkat karena keacakan pemilihan tetangga.

c. Hill-Climbing with Sideways Move

Pada bagian ini ditunjukkan hasil dari percobaan eksperimen Hill-Climbing with Sideways Move yang telah dilakukan.

Percobaan ke-	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Banyak iterasi hingga berhenti (Max_sid emove : 5)	Durasi
1	[107, 14, 77, 11, 110, 85, 109, 17, 5, 52, 95, 16, 42, 22, 116, 53, 96, 51, 118, 56, 114, 105,	[107, 14, 77, 11, 110, 85, 109, 17, 5, 52, 95, 16, 96, 22, 116,	37		36	0:01:41.8094 48

	24, 44, 65, 39, 100, 60, 73, 101, 79, 61, 27, 20, 37, 49, 119, 28, 38, 19, 30, 35, 91, 94, 23, 88, 89, 99, 98, 59, 12, 46, 62, 36, 15, 125, 123, 25, 112, 6, 82, 121, 41, 117, 45, 69, 106, 122, 84, 115, 66, 83, 29, 113, 48, 8, 97, 54, 103, 120, 86, 92, 1, 70, 18, 40, 58, 33, 75, 67, 43, 72, 90, 102, 87, 10, 74, 7, 9, 71, 124, 81, 80, 78, 50, 64, 93, 63, 32, 104, 68, 108, 34, 2, 13, 31, 26, 4, 111, 76, 47, 57, 55, 21, 3]	53, 26, 51, 118, 67, 114, 105, 24, 7, 65, 56, 100, 60, 25, 101, 79, 23, 27, 20, 37, 49, 44, 83, 38, 19, 121, 35, 84, 94, 99, 39, 89, 61, 98, 59, 12, 46, 62, 36, 15, 1, 123, 73, 112, 6, 82, 30, 41, 117, 45, 69, 106, 122, 91, 115, 66, 57, 31, 113, 48, 8, 97, 54, 50, 34, 86, 92, 125, 70, 18, 2, 81, 33, 75, 124, 43, 3, 90, 111, 68, 42, 74, 119, 9, 71, 21, 58, 103, 78, 55, 64, 93, 40, 108, 10, 87, 32, 120, 63, 13, 29, 104, 4, 76, 102, 47, 28, 80, 88, 72]				
2	[112, 71, 72, 15, 40, 109, 96, 14, 36, 34, 84, 79, 6, 38, 119, 83, 44, 16, 47, 90, 23, 49, 10, 68, 20, 31, 54, 92, 124, 81, 2, 103, 75, 97, 61, 120, 99, 58, 60, 110, 73, 43, 108,	[117, 71, 72, 15, 40, 109, 96, 14, 43, 34, 84, 79, 6, 38, 108, 83, 27, 68, 47, 90, 13, 49, 10, 16, 20, 115, 54, 59,	41		40	0:02:02.3820 39

	114, 4, 80, 77, 116, 117, 18, 55, 67, 39, 9, 37, 27, 22, 85, 7, 121, 21, 5, 11, 107, 111, 101, 42, 70, 76, 86, 78, 88, 100, 104, 125, 13, 59, 74, 8, 98, 24, 113, 56, 93, 63, 118, 94, 41, 122, 66, 26, 32, 89, 87, 30, 1, 57, 45, 29, 3, 35, 25, 95, 106, 91, 115, 52, 82, 12, 19, 17, 123, 48, 50, 105, 51, 28, 62, 46, 53, 64, 33, 65, 69, 102]	85, 81, 76, 32, 75, 97, 61, 120, 99, 95, 60, 110, 73, 36, 119, 55, 4, 125, 77, 116, 39, 18, 94, 67, 46, 2, 37, 44, 22, 88, 121, 7, 87, 5, 11, 45, 111, 3, 42, 70, 114, 86, 112, 28, 100, 33, 74, 63, 98, 80, 8, 66, 107, 113, 56, 93, 89, 118, 9, 17, 122, 29, 26, 103, 92, 21, 30, 1, 57, 24, 23, 101, 35, 25, 58, 106, 91, 31, 52, 82, 12, 124, 41, 123, 48, 50, 53, 51, 19, 62, 78, 105, 64, 104, 65, 69, 102]				
3	[91, 47, 17, 68, 71, 65, 89, 13, 73, 49, 38, 19, 4, 34, 61, 5, 105, 74, 93, 82, 115, 59, 56, 24, 1, 41, 62, 125, 33, 31, 30, 32, 87, 98, 102, 9, 101, 50, 118, 20, 100, 16, 7, 108, 124, 60, 23, 97, 77, 39, 8, 69, 70, 81, 58, 3, 29, 90, 112, 6, 64, 111, 114, 120, 122, 116, 45,	[91, 47, 17, 68, 50, 65, 89, 13, 1, 49, 39, 19, 4, 34, 61, 5, 101, 74, 93, 82, 115, 59, 14, 119, 73, 116, 62, 125, 106, 31, 30, 32, 87, 64, 102, 9, 97, 71, 118, 20, 100, 16, 7, 3, 124,	40		32	0:02:00.689005

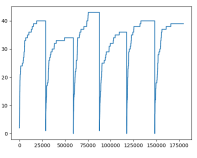
	106, 109, 40, 25, 18, 2, 84, 42, 104, 11, 79, 22, 99, 66, 51, 27, 83, 113, 37, 14, 88, 75, 36, 52, 86, 28, 44, 48, 46, 53, 76, 119, 95, 54, 78, 92, 57, 67, 10, 107, 96, 12, 123, 103, 80, 21, 26, 63, 35, 43, 117, 121, 94, 15, 55, 85, 110, 72]	60, 23, 117, 77, 38, 109, 69, 70, 123, 105, 108, 36, 90, 75, 6, 98, 111, 120, 25, 122, 37, 81, 33, 8, 40, 114, 18, 2, 84, 42, 104, 79, 11, 22, 99, 24, 51, 29, 83, 113, 41, 56, 88, 112, 27, 52, 76, 95, 44, 48, 94, 53, 86, 54, 28, 66, 78, 92, 12, 67, 10, 107, 96, 57, 45, 103, 80, 21, 26, 85, 121, 58, 43, 35, 46, 15, 55, 63, 110, 72]				
--	---	--	--	--	--	--

Dapat disimpulkan dari ketiga eksperimen bahwa Hill-Climbing with Sideways Move dapat meningkatkan nilai objektif function dengan cara yang mirip dengan Steepest Ascent Hill-Climbing tetapi dapat sideways move agar tidak terjebak local maksimum meskipun kemungkinan terjebak pada dua state dengan nilai sama.

d. Random Restart Hill-Climbing

Pada bagian ini ditunjukkan hasil dari percobaan eksperimen Random Restart Hill-Climbing yang telah dilakukan.

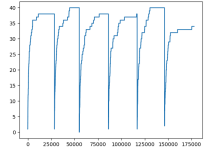
Percobaan ke-	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Banyak restart	Banyak iterasi per restart
---------------	------------	-------------	-------------------------------	------------------------------	----------------	----------------------------

1	[[8, 125, 80, 17, 38, 111, 65, 11, 1, 7, 74, 12, 33, 25, 45, 68, 103, 50, 14, 113, 51, 36, 44, 3, 13, 115, 41, 62, 54, 40, 29, 71, 93, 49, 107, 90, 4, 23, 89, 118, 10, 28, 43, 70, 9, 92, 55, 99, 95, 60, 84, 69, 77, 114, 39, 97, 21, 119, 112, 108, 122, 86, 98, 67, 35, 57, 87, 27, 121, 63, 64, 94, 120, 5, 106, 88, 47, 56, 48, 22, 37, 79, 59, 96, 18, 78, 100, 117, 123, 24, 85, 30, 26, 104, 109, 105, 102, 81, 75, 58, 46, 82, 19, 66, 42, 61, 32, 72, 91, 73, 6, 116, 16, 31, 2, 52, 15, 101, 76, 110, 34, 83, 124, 53, 20], [119, 2, 78, 57, 96, 43, 120, 5, 50, 14, 12, 102, 45, 77, 62, 85, 65, 103, 3, 21, 37, 1, 95, 51, 28, 6, 98, 55, 60, 19, 94, 83, 49, 39, 91, 11, 10, 70, 9, 52, 71, 106, 20, 86, 23, 38, 101, 32, 13, 111, 25, 26, 42, 93, 22, 125, 116, 56, 80, 54, 81, 76, 75, 31, 48, 68, 84, 118, 115, 72, 99, 53, 89, 113, 30, 17, 63, 107, 4, 34,	[47, 76, 17, 104, 111, 79, 96, 27, 109, 4, 10, 120, 86, 16, 83, 82, 119, 43, 20, 51, 94, 36, 73, 46, 66, 61, 90, 23, 48, 15, 115, 75, 9, 108, 8, 12, 100, 116, 69, 18, 45, 103, 65, 30, 72, 24, 92, 89, 59, 106, 97, 29, 39, 28, 2, 19, 55, 114, 113, 35, 42, 22, 93, 102, 57, 101, 49, 123, 5, 37, 53, 118, 25, 67, 52, 99, 64, 112, 14, 62, 122, 1, 60, 41, 91, 58, 63, 21, 107, 54, 13, 110, 38, 34, 68, 81, 77, 84, 33, 40, 11, 56, 124, 121, 125, 31, 88, 105, 85, 6, 7, 117, 50, 80, 26, 74, 3, 78, 95, 87, 98, 32, 44, 70, 71]	39		5	[28695, 30406, 28583, 29777, 30834, 31471]
---	---	---	----	--	---	---



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

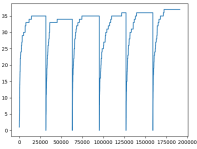
66, 53, 38, 111, 24, 63, 101, 25, 94, 17, 90, 43, 73, 83, 79, 77, 124, 117, 12, 86, 16, 102, 36, 42, 99, 104, 54, 78, 121, 2, 45, 107, 4, 56, 89, 71, 55, 92, 116, 39, 118, 75, 112, 100, 91, 97, 106, 84, 64, 13, 114, 1, 120, 15, 110, 122, 31, 51, 8, 60, 103, 37, 29, 61, 80, 68, 57, 93, 47, 40, 72, 20,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

	37, 51, 38, 65, 78, 48, 1, 89, 2, 44, 92, 45, 115, 16, 26, 36, 66, 99, 88, 14, 113, 60, 124], [47, 36, 115, 104, 111, 79, 10, 27, 96, 76, 109, 120, 86, 16, 83, 82, 60, 43, 72, 51, 81, 66, 91, 46, 40, 61, 90, 25, 4, 15, 17, 75, 54, 19, 8, 12, 100, 116, 69, 23, 71, 103, 65, 30, 20, 24, 70, 89, 59, 106, 29, 97, 39, 28, 2, 123, 55, 114, 50, 21, 26, 22, 93, 9, 108, 101, 49, 57, 5, 37, 53, 118, 3, 67, 52, 73, 122, 112, 14, 44, 64, 1, 119, 124, 80, 6, 117, 35, 107, 102, 13, 110, 58, 34, 68, 94, 77, 84, 33, 11, 48, 56, 41, 121, 88, 31, 125, 105, 85, 38, 7, 63, 113, 95, 42, 18, 74, 78, 99, 87, 98, 32, 62, 92, 45]]					
2	[[14, 58, 92, 111, 41, 16, 10, 100, 120, 22, 76, 43, 117, 63, 94, 39, 2, 28, 56, 23, 45, 107, 60, 70, 83, 37, 11, 65, 59, 30, 34, 57, 67, 52, 102, 88, 91, 108, 32, 51, 54, 21, 95,	[23, 67, 30, 105, 98, 114, 27, 31, 32, 96, 25, 92, 46, 56, 39, 21, 84, 11, 81, 118, 49, 45, 37, 90, 94, 87, 122, 59, 66, 125,	34		5	[28532, 26547, 31112, 30702, 29193, 31691]

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

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

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

	122, 59, 66, 78, 47, 100, 110, 97, 95, 57, 73, 16, 19, 33, 114, 91, 86, 116, 76, 108, 53, 93, 104, 62, 54, 79, 71, 18, 10, 112, 20, 6, 40, 34, 125, 21, 23, 15, 102, 113, 111, 44, 5, 118, 26, 74, 2, 119, 9, 43, 115, 89, 107, 55, 77, 63, 65, 60, 51, 38, 22, 12, 49, 83, 64, 82, 52, 109, 85, 39, 14, 88, 92, 42, 58, 80, 8, 124, 121, 106, 75, 69, 4, 1, 117, 103, 41, 123, 46, 7, 72, 70, 35, 96, 101, 17, 11, 29, 13]]					
3	[[56, 26, 35, 49, 110, 36, 28, 91, 65, 102, 51, 87, 54, 23, 16, 99, 68, 119, 20, 75, 103, 37, 58, 32, 27, 78, 12, 77, 107, 111, 123, 52, 114, 31, 1, 63, 10, 34, 19, 61, 17, 83, 6, 46, 117, 94, 5, 84, 25, 40, 104, 85, 66, 82, 2, 3, 115, 86, 109, 90, 53, 121, 122, 41, 70, 81, 60, 59, 4, 73, 47, 62, 18, 22, 124, 74, 15, 57, 44, 11, 21, 105, 101, 106, 79, 69, 39, 108, 97, 98, 125, 38,	[74, 101, 67, 60, 13, 124, 51, 48, 37, 55, 47, 38, 72, 40, 103, 50, 123, 58, 79, 95, 70, 73, 39, 87, 49, 56, 27, 10, 78, 89, 125, 110, 16, 66, 111, 36, 17, 98, 102, 62, 108, 61, 120, 25, 1, 9, 2, 71, 44, 52, 92, 113, 22, 6, 86, 105, 63, 26, 12, 109, 5, 115, 75, 45, 21,	37		5	[31589, 31512, 32040, 31824, 31949, 32142]

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



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


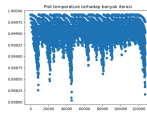
<p> 94, 22, 99, 121,  38, 78, 11, 112,  17], [8, 99, 85,  105, 14, 28, 122,  29, 117, 60, 75,  100, 103, 35, 115,  27, 61, 109, 24,  10, 97, 107, 57,  49, 92, 71, 112,  31, 16, 80, 15, 78,  110, 69, 66, 33,  76, 38, 22, 106,  84, 41, 87, 74, 82,  1, 3, 45, 9, 89, 53,  12, 50, 25, 47, 90,  93, 64, 70,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

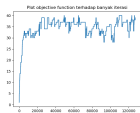
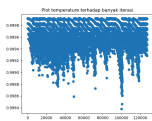
	7, 82, 29, 93, 85, 4, 32, 61, 31, 96, 76, 112, 121, 2, 117, 90, 114, 65, 97, 113, 77, 30, 53, 84, 69, 14, 120, 110, 47, 49, 116, 11, 8, 43, 73, 81, 59, 64, 106, 35, 89, 118, 88, 119, 57, 122, 99, 27, 80, 24, 91, 83, 104, 41, 100, 68, 42]]					
--	---	--	--	--	--	--


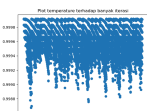
Dapat disimpulkan dari ketiga eksperimen bahwa Random Restart Hill-Climbing (dengan Stochastic Hill-Climbing) cukup random sehingga nilai objektif function bisa meningkat bisa menurun tetapi karena dilakukan berkali-kali juga berkemungkinan besar menemukan solusi yang lebih baik daripada dengan dilakukan sekali. Kelemahannya adalah waktu yang dibutuhkan yang cukup lama.

e. Simulated Annealing

Pada bagian ini ditunjukkan hasil dari percobaan eksperimen Simulated Annealing yang telah dilakukan.

Percobaan ke-	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Durasi Proses Pencarian	Plot $e^{\frac{\Delta E}{T}}$	Frekuensi 'stuck'
1	[109, 87, 11, 25, 12, 100, 49, 98, 8, 33, 9, 46, 32, 13, 26, 78, 108, 118, 14, 52, 107, 91, 89, 80, 42, 72, 40, 94, 27, 113, 36, 58, 103, 99, 31, 39, 6, 29, 111, 84, 76, 59, 96, 10, 112, 19, 115, 7, 75, 37, 51, 114, 44, 73, 3, 18, 20, 79, 120, 15, 77,	[109, 114, 11, 60, 101, 100, 20, 28, 82, 108, 46, 66, 32, 33, 12, 97, 24, 107, 112, 52, 37, 91, 56, 89, 42, 75, 40, 94, 27, 31, 76, 73, 103, 4, 59, 39, 62, 65, 111, 84, 122,	40		0:02:00.003889		36

	41, 90, 123, 45, 16, 57, 61, 48, 55, 62, 38, 93, 68, 119, 110, 56, 69, 104, 35, 74, 102, 5, 60, 28, 85, 116, 24, 92, 2, 86, 105, 65, 82, 17, 117, 54, 95, 64, 71, 63, 23, 50, 101, 88, 22, 125, 67, 21, 47, 124, 66, 30, 97, 83, 34, 106, 43, 4, 53, 81, 70, 1, 121, 122]	1, 18, 53, 121, 3, 80, 35, 120, 22, 51, 23, 44, 99, 47, 8, 49, 79, 77, 102, 55, 41, 81, 72, 45, 26, 43, 96, 38, 2, 93, 6, 68, 29, 119, 17, 118, 98, 34, 48, 74, 67, 95, 54, 25, 85, 30, 58, 16, 64, 61, 105, 115, 87, 36, 78, 71, 86, 123, 125, 63, 10, 50, 104, 88, 57, 106, 117, 21, 69, 90, 116, 124, 83, 110, 9, 13, 19, 15, 113, 14, 70, 5, 92, 7]					
2	[43, 55, 81, 95, 101, 60, 96, 56, 7, 50, 89, 88, 6, 104, 97, 70, 27, 112, 12, 105, 76, 99, 78, 39, 117, 40, 94, 84, 16, 71, 92, 25, 4, 10, 85, 9, 75, 83, 47, 69, 98, 110, 30, 33, 53, 111, 2, 37, 38, 91, 44, 62, 123, 108, 57, 42, 72, 20, 3, 32, 86, 115, 46, 82, 113, 19, 77, 119, 52, 49, 64, 59, 18, 125, 11, 1,	[66, 75, 15, 95, 24, 77, 68, 36, 99, 35, 20, 96, 59, 104, 90, 88, 69, 45, 8, 105, 64, 7, 121, 9, 114, 118, 94, 22, 16, 101, 26, 25, 103, 43, 122, 79, 10, 47, 17, 84, 42, 119, 106, 33, 57, 5, 67, 37, 83, 92, 102, 34, 100,	39		0:02:00. 004967		47

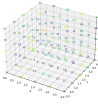
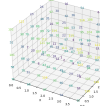
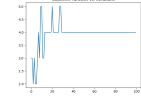
	118, 87, 29, 58, 31, 109, 23, 45, 90, 17, 121, 74, 122, 61, 65, 51, 8, 124, 63, 120, 114, 116, 26, 48, 102, 13, 34, 14, 93, 24, 5, 41, 66, 73, 79, 67, 15, 80, 22, 106, 35, 28, 54, 36, 107, 68, 103, 21, 100]	108, 124, 12, 72, 58, 3, 32, 56, 115, 38, 82, 6, 111, 14, 40, 52, 98, 19, 80, 91, 70, 55, 23, 93, 87, 29, 85, 41, 61, 1, 63, 53, 71, 44, 74, 110, 46, 60, 51, 30, 109, 65, 120, 27, 116, 4, 48, 112, 13, 49, 123, 18, 125, 76, 117, 31, 73, 11, 50, 97, 2, 89, 86, 62, 39, 113, 54, 107, 81, 21, 78, 28]					
3	[64, 35, 13, 88, 46, 109, 100, 115, 71, 124, 60, 85, 62, 1, 125, 37, 108, 41, 82, 9, 78, 94, 84, 70, 92, 44, 107, 7, 105, 29, 119, 54, 3, 23, 56, 97, 113, 31, 8, 121, 69, 24, 61, 59, 89, 50, 104, 19, 72, 91, 32, 86, 75, 26, 63, 80, 76, 83, 77, 5, 66, 103, 57, 95, 58, 20, 68, 114, 34, 14, 45, 49, 47, 16, 30, 2, 25, 40, 73, 36, 4, 65, 116, 67, 11, 51, 18, 12, 87, 112, 99, 48, 118, 52, 15, 42, 22, 96,	[36, 72, 79, 88, 40, 5, 75, 115, 35, 10, 14, 125, 62, 1, 113, 37, 108, 41, 122, 60, 78, 20, 18, 91, 92, 65, 30, 86, 105, 29, 109, 54, 8, 76, 66, 9, 33, 94, 58, 121, 82, 15, 124, 59, 13, 50, 89, 110, 120, 12, 32, 19, 97, 16, 118, 80, 46, 83, 64, 42, 73, 70, 74, 95, 3, 85, 61, 34, 77, 100,	34		0:02:00. 040828		44

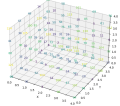
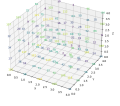
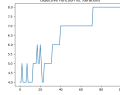
	39, 101, 21, 123, 111, 98, 81, 28, 53, 38, 120, 106, 79, 43, 90, 74, 33, 10, 93, 122, 17, 27, 102, 6, 110, 55, 117]	45, 119, 27, 63, 52, 116, 7, 2, 99, 47, 102, 87, 48, 67, 11, 24, 28, 21, 104, 112, 31, 71, 43, 93, 44, 39, 69, 49, 4, 101, 22, 123, 51, 38, 81, 55, 53, 25, 114, 68, 56, 6, 106, 57, 90, 96, 107, 23, 17, 98, 103, 26, 111, 84, 117]					
--	---	---	--	--	--	--	--

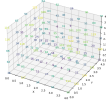
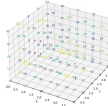
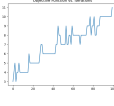
Dapat disimpulkan dari ketiga eksperimen bahwa Simulated Annealing meningkat secara cepat pada awalnya yang apabila “stuck” dapat menurun kemudian naik lagi tetapi berdasarkan keberuntungan annealing algoritma dapat menurunkan objective function dan tidak menaikannya lagi sehingga nilai akhirnya mungkin lebih buruk.

f. Genetic Algorithm

Pada bagian ini ditunjukkan hasil dari percobaan eksperimen Genetic Algorithm yang telah dilakukan.

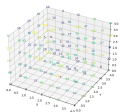
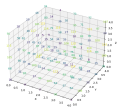
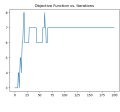
Jumlah populasi	Banyak iterasi	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Durasi
20	100	 [57, 97, 125, 59, 61, 116, 105, 30, 28, 96, 79, 27, 7, 107, 55, 104, 72, 74, 36, 53, 5, 52, 85, 99, 1, 45, 25, 67, 51, 40, 109, 113, 121, 112, 81, 87, 34, 12,	 [71, 58, 101, 51, 102, 65, 31, 37, 7, 79, 87, 124, 83, 81, 112, 34, 46, 49, 1, 56, 77, 10, 19, 63, 125, 75, 68, 38, 90, 122, 92, 8, 107, 98,	5		0:00:02.72072 1

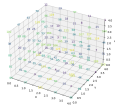
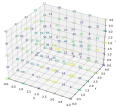
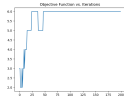
		77, 122, 22, 6, 31, 20, 15, 47, 32, 21, 101, 78, 19, 48, 49, 68, 94, 2, 4, 29, 46, 24, 10, 63, 35, 71, 44, 3, 89, 41, 13, 124, 119, 111, 115, 39, 58, 103, 114, 100, 76, 110, 82, 83, 33, 43, 8, 86, 84, 75, 73, 91, 16, 65, 56, 11, 102, 88, 18, 98, 38, 92, 80, 50, 37, 118, 42, 108, 120, 9, 95, 14, 64, 106, 17, 60, 69, 66, 90, 54, 26, 123, 62, 117, 70, 23, 93]	88, 93, 76, 18, 11, 114, 66, 24, 78, 29, 115, 120, 103, 40, 5, 69, 70, 20, 85, 59, 23, 82, 99, 94, 72, 35, 61, 2, 113, 22, 25, 3, 123, 109, 73, 21, 119, 60, 118, 121, 86, 41, 111, 89, 48, 26, 12, 33, 91, 96, 30, 9, 67, 4, 53, 36, 84, 105, 17, 104, 108, 28, 110, 74, 80, 15, 45, 50, 14, 97, 100, 64, 57, 106, 42, 16, 32, 39, 95, 116, 54, 62, 13, 44, 43, 6, 117, 52, 47, 27, 55]			
40	100	 [68, 121, 80, 41, 124, 30, 33, 23, 16, 125, 3, 123, 42, 4, 25, 99, 108, 28, 36, 2, 84, 107, 57, 78, 111, 40, 35, 90, 34, 43, 52, 1, 89, 39, 93, 12, 105, 51, 73, 20, 67, 24, 31, 60, 26, 81, 29, 117, 37, 119, 97, 82, 83, 100, 49, 9, 48, 17, 72, 45, 94, 46, 109, 32, 56,	 [115, 27, 34, 32, 107, 58, 18, 117, 41, 33, 124, 71, 40, 114, 101, 106, 92, 65, 78, 39, 43, 118, 59, 98, 120, 38, 47, 91, 20, 104, 60, 122, 87, 51, 99, 77, 119, 13, 29, 21, 108, 42, 2, 94, 67, 89, 31, 45, 11, 73, 37, 1, 53, 9, 72, 49, 36, 100, 79, 70, 76, 75, 110, 48,	8		0:00:05.71606 8

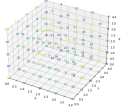
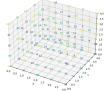
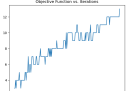
		22, 44, 115, 116, 59, 7, 102, 71, 95, 87, 21, 88, 47, 120, 14, 54, 91, 19, 50, 10, 38, 114, 66, 65, 113, 70, 103, 79, 110, 85, 106, 118, 104, 11, 61, 8, 122, 13, 63, 86, 64, 77, 75, 55, 58, 53, 74, 98, 18, 101, 62, 96, 15, 5, 69, 27, 92, 6, 112, 76]	62, 97, 93, 5, 69, 30, 56, 15, 54, 102, 111, 74, 26, 23, 121, 81, 64, 57, 80, 125, 103, 8, 66, 113, 17, 112, 25, 61, 19, 28, 50, 83, 24, 12, 86, 35, 90, 85, 123, 4, 55, 68, 88, 16, 14, 10, 95, 22, 7, 63, 3, 116, 82, 52, 46, 96, 44, 109, 6, 105, 84]			
100	100	 [36, 62, 66, 116, 113, 117, 83, 53, 115, 10, 54, 11, 29, 75, 24, 61, 76, 4, 18, 49, 93, 114, 79, 98, 108, 70, 23, 34, 39, 65, 86, 16, 71, 94, 109, 27, 7, 84, 15, 69, 92, 8, 40, 103, 48, 121, 21, 41, 5, 42, 57, 45, 91, 97, 50, 12, 3, 28, 59, 118, 58, 107, 13, 95, 73, 2, 19, 110, 123, 9, 124, 35, 25, 104, 81, 105, 30, 22, 32, 125, 44, 33, 1, 52, 63, 14, 72, 47, 74, 46, 55, 112, 38, 100, 56, 111, 68, 20, 119, 80, 67, 60, 89,	 [99, 101, 89, 78, 45, 107, 39, 115, 81, 74, 18, 32, 24, 5, 6, 93, 21, 57, 62, 82, 37, 96, 53, 77, 108, 117, 23, 124, 85, 102, 65, 88, 8, 4, 50, 86, 13, 25, 17, 66, 29, 125, 95, 98, 26, 22, 52, 40, 111, 71, 105, 36, 3, 97, 9, 12, 120, 28, 59, 114, 60, 15, 123, 44, 73, 2, 90, 91, 7, 72, 34, 43, 51, 33, 47, 80, 56, 83, 112, 75, 76, 41, 121, 16, 30, 14, 116, 113, 58, 104, 122, 94, 68, 70,	11		0:00:12.75811 5



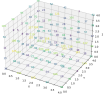
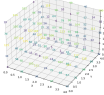
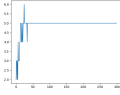
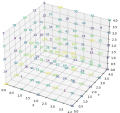
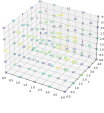
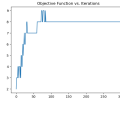
		106, 64, 85, 87, 6, 43, 82, 99, 122, 101, 31, 26, 88, 90, 102, 51, 96, 17, 120, 37, 78, 77]	64, 1, 84, 100, 19, 110, 63, 38, 35, 54, 61, 118, 69, 46, 31, 20, 79, 109, 87, 48, 92, 42, 55, 10, 67, 11, 49, 106, 27, 119, 103]			
--	--	--	--	--	--	--

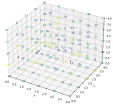
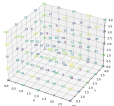
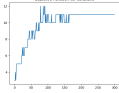
Jumlah populasi	Banyak iterasi	State awal	State akhir	Nilai akhir objektif function	Plot nilai objektif function	Durasi
20	200	 [16, 68, 105, 115, 91, 6, 85, 39, 93, 61, 2, 94, 51, 112, 43, 92, 64, 122, 44, 25, 1, 62, 81, 11, 75, 32, 60, 37, 90, 18, 17, 100, 116, 45, 117, 103, 59, 83, 70, 9, 57, 89, 40, 96, 33, 104, 55, 69, 63, 52, 31, 78, 21, 46, 29, 24, 41, 26, 5, 125, 7, 106, 23, 3, 71, 58, 74, 34, 73, 79, 95, 102, 109, 56, 13, 123, 87, 108, 53, 48, 65, 19, 54, 121, 14, 20, 82, 15, 27, 50, 38, 101, 124, 88, 47, 42,	 [16, 68, 105, 115, 91, 6, 85, 39, 93, 61, 2, 94, 51, 112, 43, 92, 64, 122, 44, 25, 1, 62, 81, 11, 75, 32, 60, 37, 59, 18, 17, 100, 49, 45, 117, 103, 90, 67, 70, 9, 57, 89, 40, 96, 33, 104, 55, 69, 63, 52, 31, 78, 21, 46, 29, 24, 41, 26, 5, 125, 7, 102, 53, 34, 119, 66, 124, 86, 120, 113, 22, 28, 8, 110, 76, 118, 87, 13, 101, 48, 65, 19, 54, 72, 14, 20, 82, 106, 27, 50, 38, 47, 3, 88, 71, 42, 35, 83, 108, 97, 111,	8		0:00:04.80755 0

		8, 118, 67, 97, 113, 110, 114, 111, 28, 76, 12, 66, 36, 4, 10, 99, 49, 120, 35, 80, 72, 86, 107, 98, 77, 22, 30, 119, 84]	15, 114, 80, 74, 56, 12, 58, 36, 4, 99, 98, 10, 95, 73, 84, 116, 30, 121, 77, 79, 109, 123, 23, 107]			
40	200	 <p>[84, 81, 80, 38, 102, 77, 76, 98, 32, 88, 121, 60, 45, 15, 52, 71, 94, 4, 42, 106, 73, 2, 122, 112, 123, 41, 62, 20, 89, 47, 9, 55, 92, 57, 26, 116, 75, 85, 101, 103, 28, 14, 70, 120, 83, 66, 46, 91, 30, 72, 37, 93, 33, 23, 43, 114, 69, 104, 53, 86, 27, 65, 125, 3, 10, 113, 36, 63, 13, 68, 74, 61, 54, 48, 117, 115, 87, 90, 118, 50, 51, 35, 24, 17, 119, 19, 34, 39, 111, 18, 82, 64, 59, 107, 8, 5, 109, 7, 124, 25, 16, 22, 67, 96, 12, 6, 105, 40, 58, 29, 99, 78, 56, 21, 97, 79, 49, 108, 95, 11, 110, 31, 44, 100, 1]</p>	 <p>[103, 34, 102, 45, 42, 80, 69, 63, 101, 10, 36, 87, 71, 17, 85, 98, 121, 115, 56, 106, 73, 2, 122, 112, 123, 41, 62, 20, 89, 47, 9, 55, 92, 57, 26, 116, 38, 15, 39, 64, 28, 14, 70, 120, 104, 66, 46, 91, 30, 72, 37, 93, 33, 109, 43, 114, 50, 108, 53, 86, 27, 65, 125, 3, 95, 113, 119, 6, 96, 24, 51, 105, 61, 88, 4, 31, 90, 78, 118, 75, 21, 16, 54, 1, 11, 107, 22, 44, 23, 13, 74, 7, 49, 99, 29, 110, 59, 124, 67, 32, 82, 58, 19, 79, 77, 83, 35, 81, 94, 18, 8, 76, 48, 97, 68, 100, 84, 25, 40, 52, 12, 5, 111, 60, 117]</p>	6		0:00:09.58328 5

100	200	 [115, 18, 11, 49, 106, 86, 93, 113, 96, 58, 20, 13, 101, 92, 50, 61, 84, 27, 16, 85, 102, 46, 80, 15, 41, 40, 35, 89, 37, 110, 94, 72, 22, 70, 112, 83, 66, 88, 14, 109, 74, 3, 71, 23, 51, 76, 105, 121, 42, 43, 53, 38, 62, 59, 87, 73, 118, 78, 108, 104, 77, 28, 63, 26, 45, 6, 47, 91, 67, 81, 10, 54, 25, 34, 117, 120, 31, 52, 9, 103, 12, 111, 21, 116, 7, 75, 69, 123, 32, 39, 95, 8, 55, 48, 17, 97, 122, 125, 44, 57, 79, 30, 64, 24, 29, 19, 2, 99, 33, 4, 82, 114, 107, 119, 65, 90, 98, 60, 124, 68, 56, 36, 100, 5, 1]	 [100, 80, 30, 46, 104, 37, 114, 71, 59, 34, 78, 69, 4, 43, 121, 67, 53, 66, 97, 57, 23, 39, 36, 111, 106, 38, 70, 109, 49, 90, 13, 51, 33, 18, 105, 42, 123, 29, 47, 72, 75, 77, 82, 10, 24, 54, 52, 3, 45, 27, 94, 93, 14, 11, 102, 12, 107, 32, 64, 85, 124, 55, 84, 115, 96, 22, 2, 79, 35, 31, 63, 58, 74, 65, 1, 41, 60, 68, 118, 92, 44, 40, 6, 98, 110, 25, 89, 99, 62, 17, 103, 81, 8, 15, 108, 87, 125, 9, 61, 120, 88, 95, 16, 83, 5, 56, 21, 91, 76, 19, 101, 20, 86, 48, 112, 50, 73, 119, 117, 7, 28, 26, 116, 122, 113]	13		0:00:23.89023 7
-----	-----	--	--	----	---	--------------------

Jumlah populasi	Banyak iterasi	State awal	State akhir	Nilai akhir objectif function	Plot nilai objetif function	Durasi
-----------------	----------------	------------	-------------	-------------------------------	-----------------------------	--------

20	300	 <p>[30, 35, 11, 54, 102, 111, 51, 107, 70, 5, 120, 60, 77, 49, 110, 63, 24, 73, 109, 119, 89, 20, 64, 96, 46, 68, 124, 108, 9, 98, 21, 16, 28, 14, 82, 95, 62, 104, 22, 69, 106, 40, 105, 15, 13, 92, 85, 7, 116, 6, 36, 45, 8, 2, 93, 86, 59, 56, 118, 91, 66, 74, 41, 53, 97, 58, 43, 115, 88, 26, 4, 25, 10, 67, 48, 87, 125, 117, 34, 55, 79, 113, 75, 12, 23, 3, 47, 18, 50, 121, 84, 29, 31, 94, 100, 114, 19, 112, 57, 37, 101, 65, 83, 17, 90, 32, 71, 44, 80, 52, 123, 39, 72, 33, 42, 122, 103, 38, 76, 1, 81, 99, 27, 78, 61]</p>	 <p>[9, 34, 123, 70, 19, 76, 92, 84, 124, 59, 36, 111, 11, 55, 116, 87, 75, 17, 5, 106, 43, 67, 66, 61, 81, 58, 48, 107, 100, 114, 105, 15, 60, 122, 62, 29, 110, 80, 90, 26, 68, 101, 2, 82, 95, 71, 45, 117, 103, 88, 46, 125, 18, 69, 102, 8, 51, 120, 108, 28, 50, 54, 3, 89, 64, 115, 63, 24, 93, 94, 86, 118, 21, 42, 52, 23, 44, 73, 96, 4, 27, 16, 31, 12, 119, 20, 85, 113, 79, 10, 72, 37, 49, 104, 97, 65, 78, 22, 112, 14, 121, 77, 47, 41, 91, 35, 74, 6, 98, 56, 33, 53, 30, 1, 99, 83, 25, 39, 57, 40, 32, 7, 38, 13, 109]</p>	6		0:00:07.2535 70
40	300	 <p>[124, 24, 38,</p>	 <p>[35, 18, 125, 46, 10, 60, 32,</p>	9		0:00:14.3900 30

		77, 5, 71, 7, 115, 87, 8, 32, 68, 73, 26, 75, 83, 118, 104, 59, 47, 36, 56, 53, 95, 122, 114, 50, 15, 4, 58, 65, 40, 117, 105, 61, 69, 125, 14, 78, 37, 1, 28, 64, 121, 88, 39, 101, 29, 111, 35, 16, 100, 45, 79, 89, 86, 93, 97, 10, 44, 30, 67, 108, 109, 80, 31, 119, 110, 107, 63, 27, 112, 91, 3, 96, 51, 70, 22, 52, 11, 33, 99, 2, 34, 57, 60, 41, 46, 42, 18, 84, 81, 94, 19, 49, 48, 76, 103, 82, 9, 98, 116, 92, 12, 6, 113, 102, 85, 66, 120, 21, 74, 106, 54, 55, 90, 13, 123, 20, 62, 23, 17, 25, 43, 72]	42, 116, 111, 41, 72, 27, 65, 110, 79, 84, 20, 99, 47, 36, 56, 53, 95, 122, 87, 50, 68, 4, 58, 75, 40, 117, 105, 61, 69, 26, 103, 83, 11, 49, 124, 102, 104, 112, 59, 21, 115, 29, 121, 31, 28, 54, 64, 1, 51, 86, 80, 88, 7, 71, 63, 74, 90, 89, 100, 119, 16, 107, 113, 101, 24, 91, 3, 96, 78, 67, 22, 52, 39, 97, 70, 45, 98, 5, 17, 123, 114, 19, 118, 109, 85, 108, 38, 30, 14, 37, 81, 48, 15, 62, 9, 77, 82, 44, 93, 106, 25, 94, 73, 120, 55, 6, 92, 23, 43, 12, 66, 8, 13, 33, 57, 76, 34, 2]			
100	300	 [30, 122, 17, 99, 47, 81, 109, 39, 37, 23, 60, 97, 113, 14, 42, 72, 112, 78, 96, 104, 92, 69, 62,	 [18, 118, 34, 120, 100, 84, 54, 37, 19, 104, 80, 25, 94, 40, 62, 7,	12		0:00:36.1922 03

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

Dapat disimpulkan dari ketiga eksperimen bahwa Genetic Algorithm semakin besar populasi awal dan banyak iterasi semakin tinggi kemungkinan untuk mendapatkan nilai objektif function yang lebih tinggi.

## BAB III

### Kesimpulan dan Saran

#### A. Kesimpulan

Jenis-jenis algoritma yang dibuat memiliki kelebihan dan kelemahan yang sesuai dengan tujuannya masing-masing. Dalam penyelesaian persoalan kubus pada bagian deskripsi persoalan untuk mencapai solusi optimal dalam penyelesaian dari kubus 5x5x5. Banyak point yang menentukan algoritma yang dipilih dalam penyelesaian persoalan ini yang dapat dilihat pada tiap tabel analisis dan eksperimen yang telah dibuat. Untuk masalah *Diagonal magic cube*, Simulated Annealing merupakan algoritma *local search* yang terbaik untuk digunakan untuk memecahkan masalah tersebut. Algoritma ini mampu menangani *local maximum* dengan baik dan memiliki kemampuan untuk menemukan solusi optimal waktu.

Dari eksperimen dan analisis dapat dilihat bahwa Simulated Annealing memiliki kelebihan untuk mendapatkan objektif function yang lebih tinggi dengan cepat serta dapat beradaptasi ketika mencapai *local maximum*. Meskipun Simulated Annealing berkemungkinan untuk menurunkan nilai akhirnya tetapi dengan probabilitas dan temperatur yang sesuai simulated annealing akan dapat mencapai akhir yang tinggi dengan konsisten.

#### B. Saran

Berdasarkan berjalannya aktivitas eksperimen dan analisis persoalan masalah terdapat beberapa hal yang mungkin dapat dikembangkan atau diubah untuk mencapai hasil yang lebih baik seperti,

1. Pemilihan bahasa pemrograman yang lebih cepat.
2. Pemaksimalan pengerjaan proyek dengan visualisasi yang lebih baik.
3. Pembuatan algoritma program objektif function yang efektif.

### C. Pembagian tugas tiap anggota kelompok

<b>Nama Mahasiswa</b>	:	M. Fadhil Atha
<b>NIM</b>	:	18221003
<b>Tugas ke-</b>	<b>Kegiatan</b>	
1	Membuat algoritma genetic	
2	Melakukan eksperimen	
3	Melakukan analisis algoritma	

<b>Nama Mahasiswa</b>	:	Nicholas Francis Aditjandra
<b>NIM</b>	:	18221005
<b>Tugas ke-</b>	<b>Kegiatan</b>	
1	Membuat algoritma steepest ascent hill climbing, hill climbing with sideway move, stochastic hill climbing, simulated annealing dan random restart hill climbing	
2	Membantu membuat pembahasan algoritma search dan analisis algoritma	
3		

<b>Nama Mahasiswa</b>	:	M. Rivaldi Mahari
<b>NIM</b>	:	18222119
<b>Tugas ke-</b>	<b>Kegiatan</b>	
1	Membuat algoritma stochastic hill climbing, simulated annealing	
2	Membantu membuat visualisasi algoritma	
3	Membuat dan merapikan dokumen	
4	Membantu membuat Readme	

<b>Nama Mahasiswa</b>	:	Regan Adiesta Mahendra
<b>NIM</b>	:	18222122
<b>Tugas ke-</b>	<b>Kegiatan</b>	
1	Membuat algoritma genetic	
2	Merapikan algoritma yang lain	
3	Membantu dan merapikan dokumen & readme	
4	Melakukan eksperimen	



# Referensi

- [Features of the magic cube - Magisch vierkant](#)
- [Perfect Magic Cubes \(trump.de\)](#)
- [Magic cube - Wikipedia](#)
- [Russell, S., & Norvig, P. \(2021\). \*Artificial Intelligence: A Modern Approach\*. Global Edition. Pearson Higher Ed.](#)