

# LAPORAN TUGAS BESAR

## IF3170 INTELIGENSI BUATAN

Pencarian Solusi *Magic Cube* Dengan Algoritma *Local Search*



Disusun oleh:

Yusuf Ardian Sandi	13522015
Ibrahim Ihsan Rasyid	13522018
Zaki Yudhistira Candra	13522031
Tazkia Nizami	13522032

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2024**

# **DAFTAR ISI**

<b>DAFTAR ISI</b>	<b>2</b>
<b>BAB I</b>	
<b>DESKRIPSI PERSOALAN</b>	<b>3</b>
<b>BAB II</b>	
<b>PEMBAHASAN</b>	<b>6</b>
1. Pemilihan Objective Function	6
2. Penjelasan Implementasi Algoritma Local Search	6
3. Hasil Eksperimen dan Analisis	6
<b>BAB III</b>	
<b>KESIMPULAN DAN SARAN</b>	<b>8</b>
<b>BAB IV</b>	
<b>PEMBAGIAN TUGAS</b>	<b>9</b>
1. Kesimpulan	9
2. Saran	9
<b>BAB V</b>	
<b>REFERENSI</b>	<b>10</b>

# BAB I

## DESKRIPSI PERSOALAN

Diagonal magic cube merupakan kubus yang tersusun dari angka 1 hingga  $n^3$  tanpa pengulangan dengan  $n$  adalah panjang sisi pada kubus tersebut. Angka-angka pada tersusun sedemikian rupa sehingga properti-properti berikut terpenuhi:

- Terdapat satu angka yang merupakan magic number dari kubus tersebut (Magic number tidak harus termasuk dalam rentang 1 hingga  $n^3$ , magic number juga bukan termasuk ke dalam angka yang harus dimasukkan ke dalam kubus)
- Jumlah angka-angka untuk setiap baris sama dengan magic number
- Jumlah angka-angka untuk setiap kolom sama dengan magic number
- Jumlah angka-angka untuk setiap tiang sama dengan magic number
- Jumlah angka-angka untuk seluruh diagonal ruang pada kubus sama dengan magic number
- Jumlah angka-angka untuk seluruh diagonal pada suatu potongan bidang dari kubus sama dengan magic number

Pada tugas ini, peserta kuliah akan menyelesaikan permasalahan Diagonal Magic Cube berukuran 5x5x5. Initial state dari suatu kubus adalah susunan angka 1 hingga  $5^3$  secara acak. Kemudian, tiap iterasi pada algoritma local search, langkah yang boleh dilakukan adalah menukar posisi dari 2 angka pada kubus tersebut (2 angka yang ditukar tidak harus bersebelahan). Khusus untuk genetic algorithm, boleh dilakukan penukaran posisi lebih dari 2 angka sekaligus dalam satu iterasi (tetapi hanya menukar posisi 2 angka saja juga diperbolehkan).

Peserta kuliah akan diminta untuk mengimplementasikan rencana yang telah peserta kuliah buat pada Tugas Kecil 1. Berikut merupakan hal-hal yang perlu dilakukan oleh setiap kelompok:

- Implementasikan 3 algoritma local search dengan rincian sebagai berikut:
  - Salah satu algoritma hill-climbing
  - Simulated Annealing
  - Genetic Algorithm
- Lakukan eksperimen dengan skema sebagai berikut:
  - Jalankan setiap algoritma sebanyak 3 kali, kemudian catat beberapa hal berikut:
    - Berlaku untuk semua algoritma
    - 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
- Berlaku hanya untuk Steepest Ascent Hill-Climbing dan Stochastic Hill-Climbing
  - Banyak iterasi hingga proses pencarian berhenti
- Berlaku hanya untuk Hill-Climbing with Sideways Move
  - Banyak iterasi hingga proses pencarian berhenti
  - Note: Tambahkan parameter maximum sideways move, dimana ketika banyak sideways move yang dilakukan sudah mencapai maksimum, pencarian dihentikan
- Berlaku hanya untuk Random Restart Hill-Climbing
  - Banyak restart
  - Banyak iterasi per restart
  - Note: Tambahkan parameter maximum restart, dimana ketika banyak restart sudah mencapai maksimum, pencarian dihentikan.
- Berlaku hanya untuk Simulated Annealing
  - Plot  $e^{\frac{\Delta E}{T}}$  terhadap banyak iterasi yang telah dilewati
  - Frekuensi ‘stuck’ di local optima
- Khusus untuk Genetic Algorithm, lakukan beberapa hal berikut:
  - Terdapat 2 parameter yang dapat diubah, yaitu jumlah populasi dan banyak iterasi.
  - Jadikan jumlah populasi sebagai kontrol, kemudian pilih 3 variasi banyak iterasi yang berbeda. Jalankan program sebanyak masing-masing 3 kali untuk setiap konfigurasi parameter.
  - Jadikan banyak iterasi sebagai kontrol, kemudian pilih 3 variasi jumlah populasi yang berbeda. Jalankan program sebanyak masing-masing 3 kali untuk setiap konfigurasi parameter.
  - Untuk setiap eksperimen, catat beberapa hal berikut:
    - State awal dan akhir dari kubus

- Nilai objective function akhir yang dicapai
  - Plot nilai objective function terhadap banyak iterasi yang telah dilewati (Cukup plot nilai objective function maksimum dan rata-rata dari populasi terhadap banyak iterasi yang telah dilewati. Jika sudah terlanjur membuat plot untuk tiap individu pada populasi tidak menjadi masalah, silahkan diberikan keterangan saja maksud plotnya apa)
  - Jumlah populasi
  - Banyak iterasi
  - Durasi proses pencarian
- Lakukan analisis terhadap hasil eksperimen, berikut merupakan beberapa pertanyaan yang dapat menjadi acuan untuk analisis yang peserta kuliah lakukan (peserta kuliah boleh menambahkan beberapa pertanyaan tambahan jika dirasa perlu untuk dijelaskan):
  - Seberapa dekat tiap-tiap algoritma bisa mendekati global optima dan mengapa hasilnya demikian?
  - Bagaimana perbandingan hasil pencarian tiap-tiap algoritma dengan algoritma local search yang lain?
  - Bagaimana perbandingan durasi proses pencarian tiap algoritma relatif terhadap algoritma lainnya?
  - Seberapa konsisten hasil akhir yang didapatkan dari tiap-tiap eksperimen yang dilakukan?
  - Bagaimana pengaruh banyak iterasi dan jumlah populasi terhadap hasil akhir pencarian pada Genetic Algorithm?
  - dst...
- Program yang dibuat harus bisa memvisualisasikan state awal kubus, state akhir kubus, dan juga hasil eksperimentnya (sesuaikan informasi hasil eksperimen yang ditampilkan dengan ketentuan yang telah dijelaskan di poin sebelum ini).
- Cara visualisasi dibebaskan kepada kelompok masing-masing selama seluruh angka pada kubus dan juga hasil eksperimen terlihat dengan jelas.
- Dibebaskan menggunakan bahasa pemrograman apapun.
- Diperbolehkan untuk menggunakan heuristik yang peserta kuliah buat sendiri atau dari referensi lain untuk optimasi pencarian solusi, asalkan masih dalam lingkup local search. Jangan lupa jelaskan heuristik yang peserta kuliah pakai di laporan.

## BAB II

# PEMBAHASAN

### 1. Pemilihan Objective Function

Fungsi objektif adalah komponen penting dalam optimisasi dan algoritma pencarian solusi yang bertujuan untuk mengevaluasi seberapa baik suatu solusi memenuhi tujuan yang diinginkan. Dalam konteks algoritma pencarian atau optimisasi, fungsi objektif memberikan nilai numerik pada setiap solusi potensial, yang kemudian digunakan untuk membandingkan solusi tersebut satu sama lain. Tujuannya adalah untuk memaksimumkan atau meminimumkan nilai fungsi objektif, tergantung pada jenis masalah yang sedang diselesaikan.

Dalam masalah Magic Cube 5x5x5, tujuannya adalah membuat kubus di mana setiap baris, kolom, tiang (vertikal), diagonal bidang (diagonal pada tiap sisi 2D), dan diagonal ruang (diagonal yang menghubungkan sudut-sudut kubus 3D) memiliki jumlah angka yang sama, yaitu magic number. Magic number ini biasanya dihitung sebagai:

$$\text{magic number} = n(n^3 + 1) \div 2$$

n adalah panjang sisi kubus, dalam hal ini 5. Sehingga diperoleh magic number = 315.

Dalam program penyelesaian permasalahan Magic Cube, fungsi objektif digunakan untuk memaksimalkan total banyak segmen kubus yang jumlahnya sama dengan magic number. Pendefinisian fungsi objektif Anda adalah:

$$\text{Fungsi Objektif} = \text{Total banyak Segmen yang berjumlah sama dengan magic number}$$

Banyak segmen yang dibentuk oleh unsur baris, kolom, pilar, diagonal, dan/atau lain-lain ialah sejumlah 109. Oleh karena itu dapat diambil suatu kesimpulan bahwa 0 merupakan state value terburuk dan 109 merupakan state value terbaik (global optima).

## 2. Penjelasan Implementasi Algoritma Local Search

Dalam implementasi algoritma local search, kami menggunakan bahasa Java dengan mengaplikasikan polymorphism. Berikut kelas/interface yang kami gunakan:

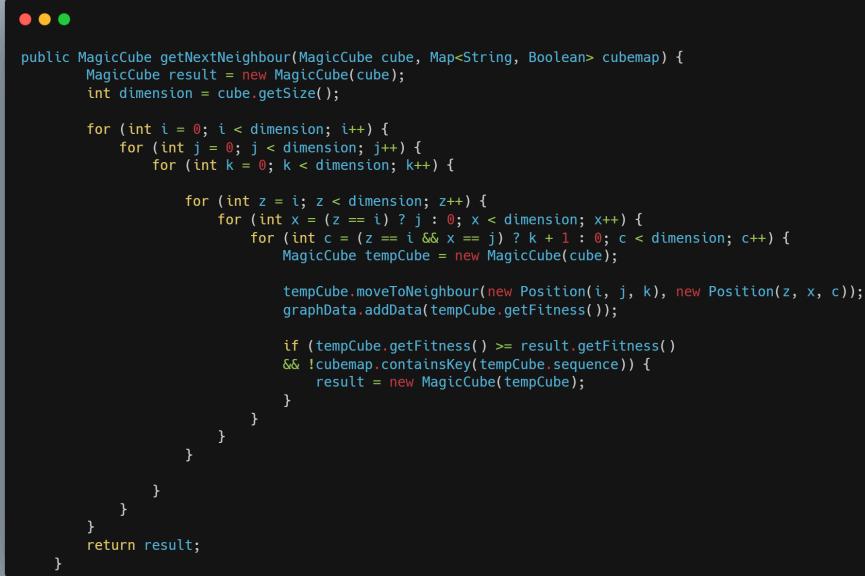
### a. Interface IAlgoritma

Java Interface untuk merepresentasikan algoritma yang akan menyelesaikan magic cube.

Deskripsi Fungsi	Source Code
Fungsi untuk menyelesaikan magic cube dan mengembalikan objek cube yang telah diselesaikan.	<pre>public interface IAlgorithm {     MagicCube getSolvedCube(MagicCube cube);      GraphData getGraphData(); }</pre>
Mendapatkan graf data yang berhubungan dengan algoritma tertentu.	

### b. Kelas HillClimbingSideMove

HillClimbingSideMove merupakan implementasi interfaceAlgorithm yang menggunakan algoritma Steepest Ascent dengan gerakan samping (sideways move) untuk memecahkan MagicCube. Ia berupaya menemukan solusi dengan berpindah secara berulang ke tetangga dengan state value yang sama atau lebih baik, yang memungkinkan sejumlah gerakan samping terbatas (bergerak ke state dengan state value yang sama) sebelum berakhir.

Deskripsi Fungsi	Source Code
Inisiasi objek hill-climbing	 <pre data-bbox="584 756 1323 889"> public HillClimbingSideMove(int max_side_moves) {     this.max_side_moves = max_side_moves;     this.graphData = new GraphData(false); } </pre>
Fungsi untuk memperoleh tetangga selanjutnya	 <pre data-bbox="523 1136 1356 1649"> public MagicCube getNextNeighbour(MagicCube cube, Map&lt;String, Boolean&gt; cubemap) {     MagicCube result = new MagicCube(cube);     int dimension = cube.getSize();      for (int i = 0; i &lt; dimension; i++) {         for (int j = 0; j &lt; dimension; j++) {             for (int k = 0; k &lt; dimension; k++) {                  for (int z = i; z &lt; dimension; z++) {                     for (int x = (z == i) ? 0 : x &lt; dimension; x++) {                         for (int c = (z == i &amp;&amp; x == j) ? k + 1 : 0; c &lt; dimension; c++) {                             MagicCube tempCube = new MagicCube(cube);                              tempCube.moveToNeighbour(new Position(i, j, k), new Position(z, x, c));                             graphData.addData(tempCube.getFitness());                              if (tempCube.getFitness() &gt;= result.getFitness()                                 &amp;&amp; !cubemap.containsKey(tempCube.sequence)) {                                 result = new MagicCube(tempCube);                             }                         }                     }                 }             }         }     }     return result; } </pre>

Fungsi utama dari hill climbing algorithm with sideway moves. Fungsi akan menerima kubus dan melakukan while loop hingga diperoleh next neighbour berupa dirinya sendiri atau ketika jumlah iterasi sideway moves yang dilakukan sudah melebihi batas maksimal.

```
● ● ●

public MagicCube getSolvedCube(MagicCube cube) {
    Map<String, Boolean> cubemap = new HashMap<>();
    graphData.addData(cube.getFitness());

    MagicCube prev = new MagicCube(cube);
    cubemap.put(cube.sequence, true);

    int side_moves = 0; // Counter untuk menghitung sideways moves

    while (true) {
        graphData.finishIteration();
        MagicCube next = new MagicCube(getNextNeighbour(prev, cubemap));
        graphData.addData(next.getFitness());

        if (next.getFitness() > prev.getFitness()) {
            side_moves = 0; // Reset jika ada peningkatan fitness
        } else if (next.getFitness() == prev.getFitness()) {
            side_moves++; // Increment jika fitness sama
            if (side_moves >= max_side_moves) {
                return prev; // Hentikan pencarian
            }
        }

        if (next.sequence.equals(prev.sequence)) {
            return next;
        }

        cubemap.put(next.sequence, true);
        prev = new MagicCube(next);
    }
}
```

### c. Kelas GeneticAlgorithm

Kelas GeneticAlgorithm mengimplementasikan interface IAlgorithm dengan algoritma genetik untuk memecahkan masalah MagicCube. Kelas ini mengelola populasi MagicCube dan mengembangkannya dari generasi ke generasi untuk menemukan solusi terbaik. Kelas ini mencakup metode untuk menghasilkan populasi awal, melakukan operasi persilangan dan mutasi, serta memilih solusi terbaik.

Deskripsi Fungsi	Source Code
------------------	-------------

Inisiasi objek  
GeneticAlgorithm

```
● ● ●

public GeneticAlgorithm(int population_size, int max_generations, double mutation_rate) {
    this.population_size = population_size;
    this.max_generations = max_generations;
    this.mutation_rate = mutation_rate;
    this.bestCube = new MagicCube(5);
    this.populations = new ArrayList<>();
    this.generateInitialPopulation();
    this.graphData = new GraphData(false);
}
```

Fungsi utama untuk  
mencari solusi  
menggunakan genetic  
algorithm

```
● ● ●

public MagicCube getSolvedCube(MagicCube cube) {
    // Find best solution from population
    this.populations.set(0, cube);
    int i = 0;
    int best_eval;
    do {
        List<MagicCube> new_mcs = new ArrayList<>();
        for (int j = 0; j < this.population_size / 2; j++) {
            MagicCube cube1 = this.randomMagicCube();
            MagicCube cube2 = this.randomMagicCube();
            List<MagicCube> children = this.crossover(cube1, cube2);
            for (MagicCube child : children) {
                MagicCube mutated_child = this.mutate(child);
                graphData.addData(mutated_child.getFitness());
                new_mcs.add(mutated_child);
            }
        }
        this.populations = new_mcs;
        i++;
        best_eval = this.getBestFitness();
        graphData.finishIteration();
        this.mutation_rate -= 0.005;
    } while (i < this.max_generations && best_eval < 90);
    return this.bestCube;
}
```

Fungsi untuk menghasilkan populasi awal

```
private void generateInitialPopulation() {  
    // Generate random initial population of MagicCubes  
    for (int i = 0; i < this.population_size; i++) {  
        MagicCube mc = new MagicCube(5);  
        this.populations.add(mc);  
    }  
}
```

Fungsi untuk melakukan random selection dan mengembalikan MagicCube yang terpilih

```
private MagicCube randomMagicCube() {  
    Random rand = new Random();  
    int sum = 0;  
    for (MagicCube mc : this.populations) {  
        sum += mc.getFitness()+1;  
    }  
    double select = rand.nextDouble();  
    select *= sum;  
    int temp = 0;  
    for (MagicCube mc : this.populations) {  
        temp += mc.getFitness()+1;  
        if (select <= temp) {  
            return mc;  
        }  
    }  
    return null;  
}
```

Fungsi untuk  
mentranslasikan  
MagicCube ke array

```
private int[] translateCubetoArray(MagicCube mc) {  
    int size = mc.getSize();  
    int[] ret = new int[125];  
    for (int i = 0; i < size; i++) {  
        for (int j = 0; j < size; j++) {  
            for (int k = 0; k < size; k++) {  
                Position pos = new Position(i, j, k);  
                int index = i * 25 + j * 5 + k;  
                ret[index] = mc.getCubeElement(pos);  
            }  
        }  
    }  
    return ret;  
}
```

Fungsi untuk  
mentranslasikan array  
ke MagicCube

```
private MagicCube translateArraytoCube(int[] arr) {  
    MagicCube ret = new MagicCube(5);  
    for (int i = 0; i < 5; i++) {  
        for (int j = 0; j < 5; j++) {  
            for (int k = 0; k < 5; k++) {  
                Position pos = new Position(i, j, k);  
                int index = i * 25 + j * 5 + k;  
                ret.setCubeElement(pos, arr[index]);  
            }  
        }  
    }  
    return ret;  
}
```

Fungsi untuk melakukan crossover

```
private List<MagicCube> crossover(MagicCube parent1, MagicCube parent2) {
    // Combine two MagicCubes to create a new MagicCube
    Random rand = new Random();
    int[] arr_parent1 = this.translateCubetoArray(parent1);
    int[] arr_parent2 = this.translateCubetoArray(parent2);

    int[] child1 = new int[125];
    int[] child2 = new int[125];
    Arrays.fill(child1, -1);
    Arrays.fill(child2, -1);

    int crossover_point1 = rand.nextInt(0, 124);
    int crossover_point2 = rand.nextInt(0, 124);
    if (crossover_point1 > crossover_point2) {
        int temp = crossover_point1;
        crossover_point1 = crossover_point2;
        crossover_point2 = temp;
    }

    for (int i = crossover_point1; i <= crossover_point2; i++) {
        child1[i] = arr_parent1[i];
        child2[i] = arr_parent2[i];
    }

    Map<Integer, Integer> mapping1 = new HashMap<>();
    Map<Integer, Integer> mapping2 = new HashMap<>();
    for (int i = crossover_point1; i <= crossover_point2; i++) {
        mapping1.put(arr_parent2[i], arr_parent1[i]);
        mapping2.put(arr_parent1[i], arr_parent2[i]);
    }

    for (int i = 0; i < 125; i++) {
        if (child1[i] == -1) {
            int gene = arr_parent2[i];
            while (mapping1.containsKey(gene)) {
                gene = mapping1.get(gene);
            }
            child1[i] = gene;
        }
    }

    for (int i = 0; i < 125; i++) {
        if (child2[i] == -1) {
            int gene = arr_parent1[i];
            while (mapping2.containsKey(gene)) {
                gene = mapping2.get(gene);
            }
            child2[i] = gene;
        }
    }

    MagicCube mc_child1 = this.translateArraytoCube(arr_parent1);
    MagicCube mc_child2 = this.translateArraytoCube(arr_parent2);

    List<MagicCube> ret = new ArrayList<>();
    if (mc_child1.getFitness() > parent1.getFitness()) {
        ret.add(mc_child1);
    } else {
        ret.add(parent1);
    }
    if (mc_child2.getFitness() > parent2.getFitness()) {
        ret.add(mc_child2);
    } else {
        ret.add(parent2);
    }

    return ret;
}
```

Fungsi untuk melakukan mutate

```
private MagicCube mutate(MagicCube cube) {
    // Mutate the cube based on mutation rate
    Random rand = new Random();

    int[] arr_chromosome = this.translateCubetoArray(cube);

    if (rand.nextDouble() < mutation_rate) {
        int pos1 = rand.nextInt(arr_chromosome.length);
        int pos2 = rand.nextInt(arr_chromosome.length);
        while (pos1 == pos2) {
            pos2 = rand.nextInt(arr_chromosome.length);
        }
        int temp = arr_chromosome[pos1];
        arr_chromosome[pos1] = arr_chromosome[pos2];
        arr_chromosome[pos2] = temp;
    }
    MagicCube mutated = this.translateArraytoCube(arr_chromosome);
    return mutated;
}
```

Fungsi untuk menentukan nilai fitness function terbaik dari populasi saat itu

```
private int getBestFitness() {
    int ret = 0;
    for (MagicCube mc : this.populations) {
        if (mc.getFitness() > ret) {
            ret = mc.getFitness();
            this.bestCube.copyFrom(mc);
        }
    }
    return ret;
}
```

Fungsi-fungsi getter dan setter pada kelas

```
public int getPopulationSize() {
    return population_size;
}

public void setPopulationSize(int population_size)
{   this.population_size = population_size;
}

public int getMaxGenerations() {
    return max_generations;
}

public void setMaxGenerations(int max_generations)
{   this.max_generations = max_generations;
}
```

#### d. Kelas SimulatedAnnealing

Kelas untuk implementasi algoritma simulated annealing untuk menyelesaikan magic cube. Menerima masukkan initial temperatur yang akan diturunkan secara perlahan sekaligus menjelajahi tetangga magic cube secara acak. Kelas ini juga menyimpan riwayat probabilitas yang dihasilkan untuk ditampilkan ke grafik. Selain itu, algoritma ini juga terdapat sebuah modifikasi (heuristic) dimana daripada mengembalikan solusi berupa current state, algoritma ini akan mengembalikan solusi terbaik yang pernah dihasilkan saja. Threshold peluang yang digunakan pada algoritma ini adalah 0,95 dengan suhu awal dan *cooling rate* secara berturut-turut adalah 10 dan 0,000001.

Deskripsi Fungsi	Source Code
------------------	-------------

Menghasilkan tetangga acak dari Magic Cube yang diberikan dengan menukar dua posisi acak.

```
public MagicCube getRandomNeighbour(MagicCube cube) {  
    MagicCube neighbor = new MagicCube(cube);  
  
    // Get random two Positions  
    Random rand = new Random();  
    int x1 = rand.nextInt(cube.getSize());  
    int y1 = rand.nextInt(cube.getSize());  
    int z1 = rand.nextInt(cube.getSize());  
    Position el1 = new Position(x1, y1, z1);  
  
    int x2 = rand.nextInt(cube.getSize());  
    int y2 = rand.nextInt(cube.getSize());  
    int z2 = rand.nextInt(cube.getSize());  
    Position el2 = new Position(x2, y2, z2);  
  
    // Swap the two elements  
    neighbor.moveToNeighbour(el1, el2);  
    return neighbor;  
}
```

Menyelesaikan Magic Cube yang diberikan. Menghapus riwayat probabilitas sebelumnya dan menambahkannya dengan yang baru dari algoritma yang telah dijalankan.

```
● ● ●

@Override
public MagicCube getSolvedCube(MagicCube cube) {
    double temperature = initial_temperature;
    MagicCube currentCube = new MagicCube(cube);
    MagicCube bestCube = new MagicCube(cube);
    int iteration = 0;

    probabilityHistory.clear();

    int localOptimumFrequency = 0;

    while (temperature > 1) {
        iteration++;

        MagicCube neighbour = getRandomNeighbour(currentCube);
        int currentFitness = currentCube.getFitness();
        int neighbourFitness = neighbour.getFitness();

        double accProbability = acceptanceProbability(currentFitness, neighbourFitness, temperature);

        graphData.addData(neighbourFitness, temperature);
        // probabilityHistory.add(accProbability);

        // Hitung frekuensi stuck local optimum
        if ((neighbourFitness > currentFitness)) {
            localOptimumFrequency++;
        }

        if (accProbability > 0.95) {
            graphData.finishIteration();
            currentCube = new MagicCube(neighbour);
        }

        if (currentFitness > bestCube.getFitness()) {
            bestCube = new MagicCube(currentCube);
            // System.out.println("Best Fitness: " + bestCube.getFitness());
            // System.out.println("Temperature: " + temperature);
        }
        temperature *= 1 - cooling_rate;
    }

    System.out.println("Number of iterations: " + iteration);
    System.out.println("Local Optimum Stuck Frequency: " + localOptimumFrequency);
    return bestCube;
}
```

Menghitung probabilitas perpindahan ke solusi tetangga dalam algoritma simulated annealing.

```
● ● ●

private double acceptanceProbability(int currentFitness, int neighbourFitness, double temperature) {
    if (neighbourFitness > currentFitness) {
        // counter++;
        return 1.0;
    }
    return Math.exp((neighbourFitness - currentFitness) / temperature);
}
```

Mengambil riwayat probabilitas yang terekam selama proses algoritma.

```
● ● ●  
public ArrayList<Double> getProbabilityHistory() {  
    return probabilityHistory;  
}
```

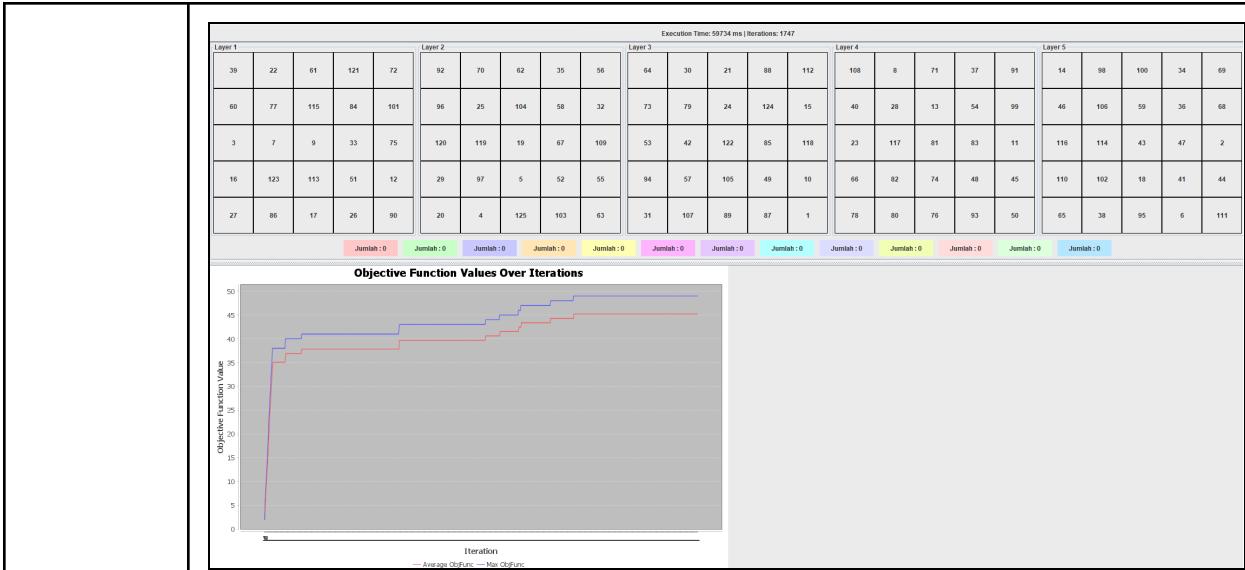
Mendapatkan data graf

```
● ● ●  
@Override  
public GraphData getGraphData() {  
    return graphData;  
}
```

### 3. Hasil Eksperimen dan Analisis

#### a. Algoritma Hill Climbing With Sideways Move

Percobaan	Hasil																																																																																
1	<p>State sebelum solving:</p> <p>Execution Time: 52024 ms   Iterations: 1450</p> <table border="1"> <thead> <tr> <th>Layer 1</th><th>Layer 2</th><th>Layer 3</th><th>Layer 4</th><th>Layer 5</th></tr> </thead> <tbody> <tr><td>96 41 88 38 6</td><td>2 115 17 53 59</td><td>117 69 125 26 119</td><td>113 100 116 20 106</td><td>121 79 107 43 54</td></tr> <tr><td>77 63 82 16 71</td><td>89 12 28 111 31</td><td>39 49 95 58 23</td><td>5 1 105 56 19</td><td>78 10 13 103 83</td></tr> <tr><td>70 87 18 37 46</td><td>7 36 4 97 98</td><td>45 32 123 92 62</td><td>75 33 44 60 86</td><td>3 15 34 51 25</td></tr> <tr><td>8 76 91 67 29</td><td>110 64 35 40 112</td><td>93 50 14 11 9</td><td>81 72 120 42 22</td><td>68 21 102 104 47</td></tr> <tr><td>94 57 66 48 85</td><td>101 124 122 99 27</td><td>74 108 84 80 118</td><td>73 61 30 114 55</td><td>109 65 90 24 52</td></tr> <tr><td colspan="10">Jumlah : 0 Jumlah : 0</td></tr> </tbody> </table> <p>State setelah solving:</p> <p>Execution Time: 52024 ms   Iterations: 1450</p> <table border="1"> <thead> <tr> <th>Layer 1</th><th>Layer 2</th><th>Layer 3</th><th>Layer 4</th><th>Layer 5</th></tr> </thead> <tbody> <tr><td>96 32 97 38 52</td><td>64 100 23 69 59</td><td>17 48 53 117 80</td><td>7 20 105 111 72</td><td>54 115 37 44 65</td></tr> <tr><td>77 63 123 91 71</td><td>14 2 34 26 31</td><td>89 49 29 56 92</td><td>79 88 116 39 22</td><td>12 19 62 103 99</td></tr> <tr><td>70 87 18 94 46</td><td>45 41 101 16 112</td><td>27 36 107 95 59</td><td>75 33 61 86 60</td><td>98 118 28 24 47</td></tr> <tr><td>8 76 62 67 58</td><td>110 125 35 40 5</td><td>74 83 113 11 78</td><td>81 21 3 104 106</td><td>42 10 102 93 68</td></tr> <tr><td>9 57 13 25 85</td><td>4 124 122 119 108</td><td>120 90 84 6 15</td><td>73 43 30 114 55</td><td>109 1 66 51 121</td></tr> <tr><td colspan="10">Jumlah : 0 Jumlah : 0</td></tr> </tbody> </table> <p>Objective Function Values Over Iterations</p> <p>Keterangan:</p> <pre>Fitness before solving: 0 Fitness after solving: 52 iterasi: 1450 Waktu eksekusi: 52024 ms</pre>	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	96 41 88 38 6	2 115 17 53 59	117 69 125 26 119	113 100 116 20 106	121 79 107 43 54	77 63 82 16 71	89 12 28 111 31	39 49 95 58 23	5 1 105 56 19	78 10 13 103 83	70 87 18 37 46	7 36 4 97 98	45 32 123 92 62	75 33 44 60 86	3 15 34 51 25	8 76 91 67 29	110 64 35 40 112	93 50 14 11 9	81 72 120 42 22	68 21 102 104 47	94 57 66 48 85	101 124 122 99 27	74 108 84 80 118	73 61 30 114 55	109 65 90 24 52	Jumlah : 0										Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	96 32 97 38 52	64 100 23 69 59	17 48 53 117 80	7 20 105 111 72	54 115 37 44 65	77 63 123 91 71	14 2 34 26 31	89 49 29 56 92	79 88 116 39 22	12 19 62 103 99	70 87 18 94 46	45 41 101 16 112	27 36 107 95 59	75 33 61 86 60	98 118 28 24 47	8 76 62 67 58	110 125 35 40 5	74 83 113 11 78	81 21 3 104 106	42 10 102 93 68	9 57 13 25 85	4 124 122 119 108	120 90 84 6 15	73 43 30 114 55	109 1 66 51 121	Jumlah : 0									
Layer 1	Layer 2	Layer 3	Layer 4	Layer 5																																																																													
96 41 88 38 6	2 115 17 53 59	117 69 125 26 119	113 100 116 20 106	121 79 107 43 54																																																																													
77 63 82 16 71	89 12 28 111 31	39 49 95 58 23	5 1 105 56 19	78 10 13 103 83																																																																													
70 87 18 37 46	7 36 4 97 98	45 32 123 92 62	75 33 44 60 86	3 15 34 51 25																																																																													
8 76 91 67 29	110 64 35 40 112	93 50 14 11 9	81 72 120 42 22	68 21 102 104 47																																																																													
94 57 66 48 85	101 124 122 99 27	74 108 84 80 118	73 61 30 114 55	109 65 90 24 52																																																																													
Jumlah : 0																																																																																	
Layer 1	Layer 2	Layer 3	Layer 4	Layer 5																																																																													
96 32 97 38 52	64 100 23 69 59	17 48 53 117 80	7 20 105 111 72	54 115 37 44 65																																																																													
77 63 123 91 71	14 2 34 26 31	89 49 29 56 92	79 88 116 39 22	12 19 62 103 99																																																																													
70 87 18 94 46	45 41 101 16 112	27 36 107 95 59	75 33 61 86 60	98 118 28 24 47																																																																													
8 76 62 67 58	110 125 35 40 5	74 83 113 11 78	81 21 3 104 106	42 10 102 93 68																																																																													
9 57 13 25 85	4 124 122 119 108	120 90 84 6 15	73 43 30 114 55	109 1 66 51 121																																																																													
Jumlah : 0																																																																																	
2	<p>State sebelum solving:</p> <table border="1"> <thead> <tr> <th>Layer 1</th><th>Layer 2</th><th>Layer 3</th><th>Layer 4</th><th>Layer 5</th></tr> </thead> <tbody> <tr><td>115 22 61 121 72</td><td>13 70 62 35 56</td><td>47 57 21 88 112</td><td>108 8 71 69 11</td><td>98 95 100 34 37</td></tr> <tr><td>44 77 18 84 101</td><td>98 25 104 51 68</td><td>123 40 58 24 99</td><td>125 28 30 54 110</td><td>46 106 29 102 45</td></tr> <tr><td>3 20 105 120 75</td><td>109 119 67 41 1</td><td>53 94 122 59 91</td><td>23 83 81 6 74</td><td>85 17 43 15 52</td></tr> <tr><td>16 2 7 27 12</td><td>57 60 5 10 33</td><td>42 107 64 49 39</td><td>38 82 73 48 93</td><td>78 36 9 92 118</td></tr> <tr><td>116 63 113 26 86</td><td>114 4 55 103 76</td><td>31 87 90 89 79</td><td>50 80 19 117 124</td><td>65 111 66 32 14</td></tr> <tr><td colspan="10">Jumlah : 0 Jumlah : 0</td></tr> </tbody> </table> <p>State setelah solving:</p>	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	115 22 61 121 72	13 70 62 35 56	47 57 21 88 112	108 8 71 69 11	98 95 100 34 37	44 77 18 84 101	98 25 104 51 68	123 40 58 24 99	125 28 30 54 110	46 106 29 102 45	3 20 105 120 75	109 119 67 41 1	53 94 122 59 91	23 83 81 6 74	85 17 43 15 52	16 2 7 27 12	57 60 5 10 33	42 107 64 49 39	38 82 73 48 93	78 36 9 92 118	116 63 113 26 86	114 4 55 103 76	31 87 90 89 79	50 80 19 117 124	65 111 66 32 14	Jumlah : 0																																																	
Layer 1	Layer 2	Layer 3	Layer 4	Layer 5																																																																													
115 22 61 121 72	13 70 62 35 56	47 57 21 88 112	108 8 71 69 11	98 95 100 34 37																																																																													
44 77 18 84 101	98 25 104 51 68	123 40 58 24 99	125 28 30 54 110	46 106 29 102 45																																																																													
3 20 105 120 75	109 119 67 41 1	53 94 122 59 91	23 83 81 6 74	85 17 43 15 52																																																																													
16 2 7 27 12	57 60 5 10 33	42 107 64 49 39	38 82 73 48 93	78 36 9 92 118																																																																													
116 63 113 26 86	114 4 55 103 76	31 87 90 89 79	50 80 19 117 124	65 111 66 32 14																																																																													
Jumlah : 0																																																																																	



Keterangan:

**Fitness before solving:** 2  
**Fitness after solving:** 49  
**iterasi:** 1747  
**Waktu eksekusi:** 59734 ms

3

State sebelum solving:

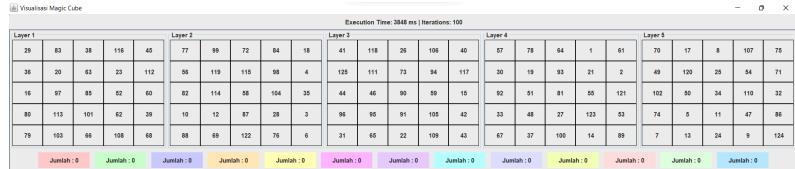
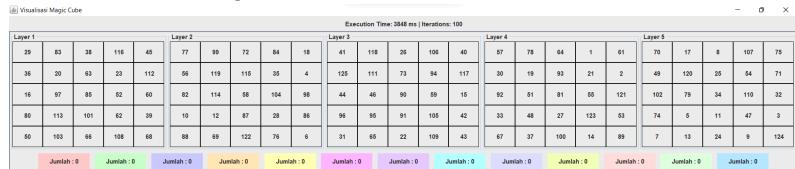
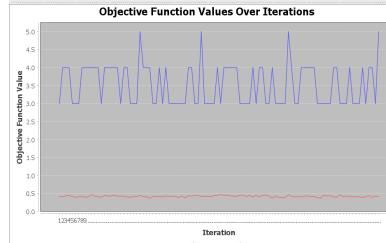
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5				
90	112	85	1	92	39	80	113	47	4	46	54	98	116	63	36	81	5	28	26	35	21	17	124	102
120	25	42	32	33	15	61	52	88	68	64	101	115	29	72	13	50	14	65	6	114	84	96	34	121
56	57	118	66	67	122	12	103	49	71	119	45	100	125	82	97	93	20	105	70	76	94	60	95	27
18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

State setelah solving:



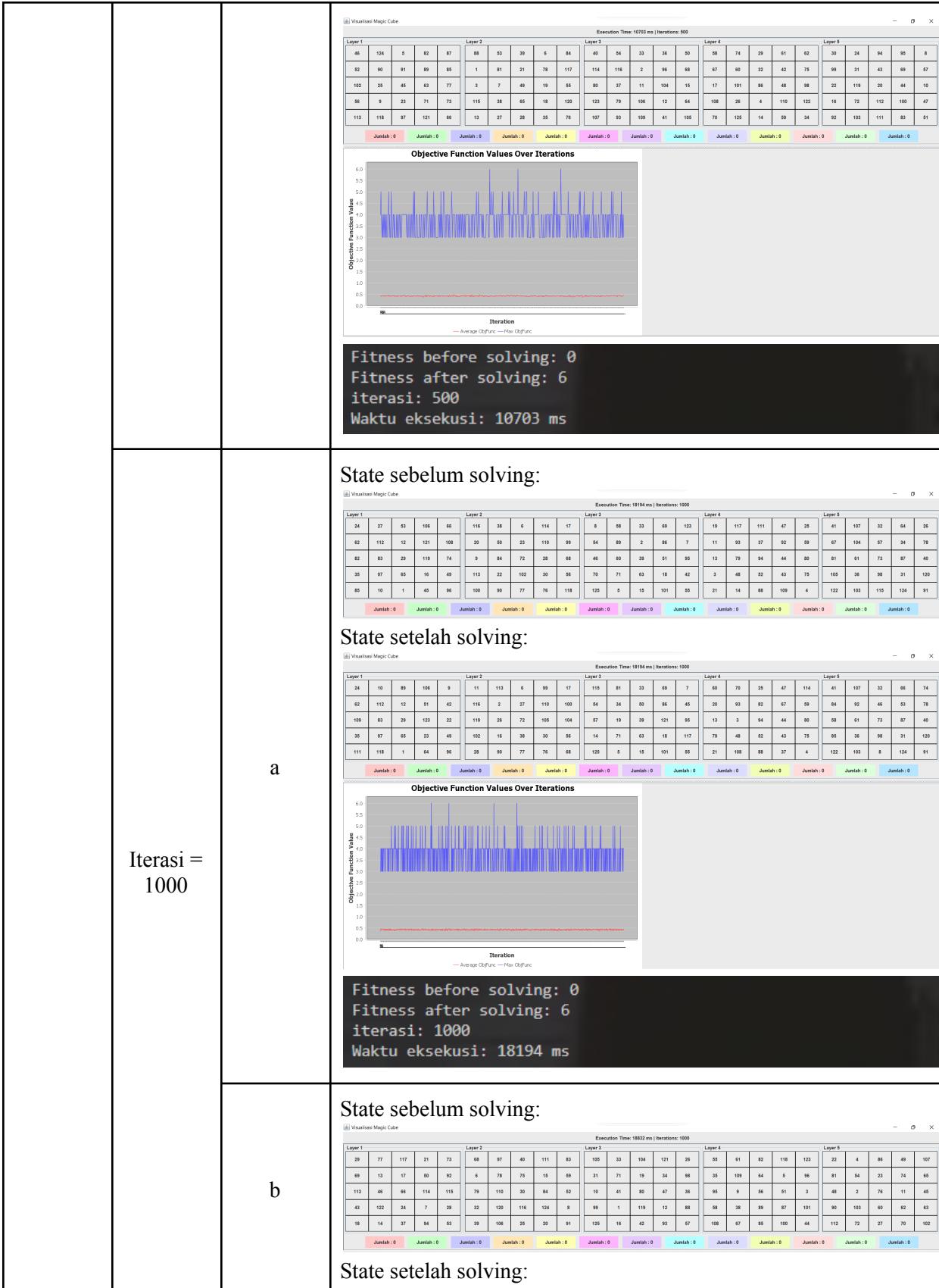
	<p>Keterangan:</p> <pre> Fitness before solving: 0 Fitness after solving: 40 iterasi: 711 Waktu eksekusi: 23825 ms </pre>
--	---

## b. Algoritma Genetic

Variabel Kontrol	Keterangan	Percobaan	Hasil
Populasi = 1000	Iterasi = 100	a	<p>State sebelum solving:</p>  <p>State setelah solving:</p>  <p>Objective Function Values Over Iterations</p>  <pre> Fitness before solving: 0 Fitness after solving: 5 iterasi: 100 Waktu eksekusi: 3848 ms </pre>
		b	<p>State sebelum solving:</p>  <p>State setelah solving:</p> 

			<p>Visualisation Magic Cube</p> <p>Execution Time: 3747 ms   Iterations: 100</p> <table border="1"> <thead> <tr><th colspan="5">Layer 1</th><th colspan="5">Layer 2</th><th colspan="5">Layer 3</th><th colspan="5">Layer 4</th><th colspan="5">Layer 5</th></tr> </thead> <tbody> <tr><td>37</td><td>110</td><td>43</td><td>62</td><td>86</td><td>82</td><td>40</td><td>61</td><td>93</td><td>59</td><td>104</td><td>10</td><td>106</td><td>112</td><td>69</td><td>91</td><td>21</td><td>15</td><td>50</td><td>57</td><td>20</td><td>11</td><td>9</td><td>117</td><td>77</td></tr> <tr><td>17</td><td>8</td><td>76</td><td>118</td><td>80</td><td>73</td><td>25</td><td>28</td><td>27</td><td>16</td><td>33</td><td>24</td><td>69</td><td>46</td><td>123</td><td>38</td><td>53</td><td>98</td><td>41</td><td>47</td><td>92</td><td>46</td><td>50</td><td>65</td><td>92</td></tr> <tr><td>19</td><td>4</td><td>83</td><td>30</td><td>32</td><td>101</td><td>121</td><td>94</td><td>102</td><td>96</td><td>12</td><td>48</td><td>26</td><td>119</td><td>26</td><td>113</td><td>111</td><td>6</td><td>23</td><td>67</td><td>80</td><td>79</td><td>66</td><td>14</td><td>125</td></tr> <tr><td>56</td><td>42</td><td>56</td><td>79</td><td>5</td><td>2</td><td>44</td><td>109</td><td>39</td><td>1</td><td>95</td><td>79</td><td>74</td><td>85</td><td>107</td><td>61</td><td>64</td><td>7</td><td>34</td><td>122</td><td>134</td><td>22</td><td>50</td><td>88</td><td>3</td></tr> <tr><td>100</td><td>54</td><td>35</td><td>51</td><td>55</td><td>68</td><td>31</td><td>114</td><td>84</td><td>70</td><td>115</td><td>108</td><td>49</td><td>103</td><td>87</td><td>116</td><td>73</td><td>63</td><td>29</td><td>18</td><td>97</td><td>13</td><td>120</td><td>105</td><td>77</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p>Objective Function Values Over Iterations</p> <p>Iteration</p> <p>Average ObjFunc — Max ObjFunc</p> <p><b>Fitness before solving:</b> 0  <b>Fitness after solving:</b> 6  <b>iterasi:</b> 100  <b>Waktu eksekusi:</b> 3747 ms</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					37	110	43	62	86	82	40	61	93	59	104	10	106	112	69	91	21	15	50	57	20	11	9	117	77	17	8	76	118	80	73	25	28	27	16	33	24	69	46	123	38	53	98	41	47	92	46	50	65	92	19	4	83	30	32	101	121	94	102	96	12	48	26	119	26	113	111	6	23	67	80	79	66	14	125	56	42	56	79	5	2	44	109	39	1	95	79	74	85	107	61	64	7	34	122	134	22	50	88	3	100	54	35	51	55	68	31	114	84	70	115	108	49	103	87	116	73	63	29	18	97	13	120	105	77																																																																																																																																																						
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
37	110	43	62	86	82	40	61	93	59	104	10	106	112	69	91	21	15	50	57	20	11	9	117	77																																																																																																																																																																																																																																																																																							
17	8	76	118	80	73	25	28	27	16	33	24	69	46	123	38	53	98	41	47	92	46	50	65	92																																																																																																																																																																																																																																																																																							
19	4	83	30	32	101	121	94	102	96	12	48	26	119	26	113	111	6	23	67	80	79	66	14	125																																																																																																																																																																																																																																																																																							
56	42	56	79	5	2	44	109	39	1	95	79	74	85	107	61	64	7	34	122	134	22	50	88	3																																																																																																																																																																																																																																																																																							
100	54	35	51	55	68	31	114	84	70	115	108	49	103	87	116	73	63	29	18	97	13	120	105	77																																																																																																																																																																																																																																																																																							
			<p>State sebelum solving:</p> <p>Visualisation Magic Cube</p> <p>Execution Time: 3648 ms   Iterations: 100</p> <table border="1"> <thead> <tr><th colspan="5">Layer 1</th><th colspan="5">Layer 2</th><th colspan="5">Layer 3</th><th colspan="5">Layer 4</th><th colspan="5">Layer 5</th></tr> </thead> <tbody> <tr><td>45</td><td>97</td><td>55</td><td>105</td><td>115</td><td>106</td><td>38</td><td>84</td><td>78</td><td>62</td><td>90</td><td>31</td><td>10</td><td>91</td><td>79</td><td>124</td><td>94</td><td>122</td><td>121</td><td>48</td><td>70</td><td>87</td><td>25</td><td>34</td><td>51</td></tr> <tr><td>7</td><td>75</td><td>114</td><td>18</td><td>34</td><td>95</td><td>29</td><td>1</td><td>57</td><td>110</td><td>86</td><td>53</td><td>61</td><td>88</td><td>43</td><td>80</td><td>28</td><td>65</td><td>33</td><td>120</td><td>47</td><td>16</td><td>23</td><td>116</td><td>68</td></tr> <tr><td>65</td><td>22</td><td>59</td><td>13</td><td>50</td><td>44</td><td>21</td><td>46</td><td>12</td><td>42</td><td>118</td><td>100</td><td>95</td><td>104</td><td>8</td><td>111</td><td>67</td><td>38</td><td>41</td><td>85</td><td>117</td><td>11</td><td>75</td><td>119</td><td>54</td></tr> <tr><td>19</td><td>92</td><td>4</td><td>109</td><td>71</td><td>14</td><td>77</td><td>30</td><td>103</td><td>5</td><td>102</td><td>37</td><td>17</td><td>40</td><td>27</td><td>58</td><td>63</td><td>2</td><td>52</td><td>26</td><td>82</td><td>48</td><td>88</td><td>123</td><td>113</td></tr> <tr><td>73</td><td>98</td><td>99</td><td>39</td><td>6</td><td>3</td><td>93</td><td>66</td><td>20</td><td>101</td><td>91</td><td>74</td><td>9</td><td>83</td><td>69</td><td>64</td><td>107</td><td>56</td><td>112</td><td>72</td><td>108</td><td>15</td><td>125</td><td>32</td><td>25</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p>State setelah solving:</p> <p>Visualisation Magic Cube</p> <p>Execution Time: 3648 ms   Iterations: 100</p> <table border="1"> <thead> <tr><th colspan="5">Layer 1</th><th colspan="5">Layer 2</th><th colspan="5">Layer 3</th><th colspan="5">Layer 4</th><th colspan="5">Layer 5</th></tr> </thead> <tbody> <tr><td>45</td><td>97</td><td>55</td><td>105</td><td>115</td><td>106</td><td>38</td><td>84</td><td>78</td><td>62</td><td>90</td><td>31</td><td>10</td><td>91</td><td>79</td><td>124</td><td>94</td><td>122</td><td>121</td><td>48</td><td>70</td><td>87</td><td>25</td><td>34</td><td>51</td></tr> <tr><td>7</td><td>75</td><td>114</td><td>18</td><td>34</td><td>95</td><td>29</td><td>1</td><td>57</td><td>110</td><td>86</td><td>53</td><td>61</td><td>88</td><td>43</td><td>80</td><td>28</td><td>65</td><td>33</td><td>120</td><td>47</td><td>16</td><td>23</td><td>116</td><td>68</td></tr> <tr><td>65</td><td>22</td><td>59</td><td>13</td><td>50</td><td>44</td><td>21</td><td>46</td><td>12</td><td>42</td><td>118</td><td>100</td><td>95</td><td>104</td><td>8</td><td>111</td><td>67</td><td>38</td><td>41</td><td>85</td><td>117</td><td>11</td><td>75</td><td>119</td><td>54</td></tr> <tr><td>19</td><td>92</td><td>4</td><td>109</td><td>71</td><td>14</td><td>77</td><td>30</td><td>103</td><td>5</td><td>102</td><td>37</td><td>17</td><td>40</td><td>27</td><td>58</td><td>63</td><td>2</td><td>52</td><td>26</td><td>82</td><td>48</td><td>88</td><td>123</td><td>113</td></tr> <tr><td>73</td><td>98</td><td>99</td><td>39</td><td>6</td><td>3</td><td>93</td><td>66</td><td>20</td><td>101</td><td>91</td><td>74</td><td>9</td><td>83</td><td>69</td><td>64</td><td>107</td><td>56</td><td>112</td><td>72</td><td>108</td><td>15</td><td>125</td><td>32</td><td>25</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p>Objective Function Values Over Iterations</p> <p>Iteration</p> <p>Average ObjFunc — Max ObjFunc</p> <p><b>Fitness before solving:</b> 1  <b>Fitness after solving:</b> 5  <b>iterasi:</b> 100  <b>Waktu eksekusi:</b> 3648 ms</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					45	97	55	105	115	106	38	84	78	62	90	31	10	91	79	124	94	122	121	48	70	87	25	34	51	7	75	114	18	34	95	29	1	57	110	86	53	61	88	43	80	28	65	33	120	47	16	23	116	68	65	22	59	13	50	44	21	46	12	42	118	100	95	104	8	111	67	38	41	85	117	11	75	119	54	19	92	4	109	71	14	77	30	103	5	102	37	17	40	27	58	63	2	52	26	82	48	88	123	113	73	98	99	39	6	3	93	66	20	101	91	74	9	83	69	64	107	56	112	72	108	15	125	32	25	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					45	97	55	105	115	106	38	84	78	62	90	31	10	91	79	124	94	122	121	48	70	87	25	34	51	7	75	114	18	34	95	29	1	57	110	86	53	61	88	43	80	28	65	33	120	47	16	23	116	68	65	22	59	13	50	44	21	46	12	42	118	100	95	104	8	111	67	38	41	85	117	11	75	119	54	19	92	4	109	71	14	77	30	103	5	102	37	17	40	27	58	63	2	52	26	82	48	88	123	113	73	98	99	39	6	3	93	66	20	101	91	74	9	83	69	64	107	56	112	72	108	15	125	32	25
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
45	97	55	105	115	106	38	84	78	62	90	31	10	91	79	124	94	122	121	48	70	87	25	34	51																																																																																																																																																																																																																																																																																							
7	75	114	18	34	95	29	1	57	110	86	53	61	88	43	80	28	65	33	120	47	16	23	116	68																																																																																																																																																																																																																																																																																							
65	22	59	13	50	44	21	46	12	42	118	100	95	104	8	111	67	38	41	85	117	11	75	119	54																																																																																																																																																																																																																																																																																							
19	92	4	109	71	14	77	30	103	5	102	37	17	40	27	58	63	2	52	26	82	48	88	123	113																																																																																																																																																																																																																																																																																							
73	98	99	39	6	3	93	66	20	101	91	74	9	83	69	64	107	56	112	72	108	15	125	32	25																																																																																																																																																																																																																																																																																							
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
45	97	55	105	115	106	38	84	78	62	90	31	10	91	79	124	94	122	121	48	70	87	25	34	51																																																																																																																																																																																																																																																																																							
7	75	114	18	34	95	29	1	57	110	86	53	61	88	43	80	28	65	33	120	47	16	23	116	68																																																																																																																																																																																																																																																																																							
65	22	59	13	50	44	21	46	12	42	118	100	95	104	8	111	67	38	41	85	117	11	75	119	54																																																																																																																																																																																																																																																																																							
19	92	4	109	71	14	77	30	103	5	102	37	17	40	27	58	63	2	52	26	82	48	88	123	113																																																																																																																																																																																																																																																																																							
73	98	99	39	6	3	93	66	20	101	91	74	9	83	69	64	107	56	112	72	108	15	125	32	25																																																																																																																																																																																																																																																																																							
Iterasi = 500	a	<p>State sebelum solving:</p> <p>Visualisation Magic Cube</p> <p>Execution Time: 1137 ms   Iterations: 500</p> <table border="1"> <thead> <tr><th colspan="5">Layer 1</th><th colspan="5">Layer 2</th><th colspan="5">Layer 3</th><th colspan="5">Layer 4</th><th colspan="5">Layer 5</th></tr> </thead> <tbody> <tr><td>76</td><td>119</td><td>89</td><td>99</td><td>43</td><td>35</td><td>42</td><td>122</td><td>59</td><td>41</td><td>90</td><td>31</td><td>82</td><td>6</td><td>100</td><td>38</td><td>25</td><td>63</td><td>108</td><td>61</td><td>4</td><td>10</td><td>85</td><td>73</td><td>117</td><td>90</td></tr> <tr><td>112</td><td>54</td><td>21</td><td>32</td><td>95</td><td>62</td><td>103</td><td>98</td><td>44</td><td>60</td><td>92</td><td>106</td><td>104</td><td>81</td><td>97</td><td>24</td><td>50</td><td>33</td><td>74</td><td>105</td><td>55</td><td>64</td><td>115</td><td>49</td><td>7</td></tr> <tr><td>82</td><td>23</td><td>14</td><td>91</td><td>36</td><td>57</td><td>11</td><td>121</td><td>79</td><td>1</td><td>109</td><td>83</td><td>34</td><td>48</td><td>85</td><td>51</td><td>96</td><td>124</td><td>67</td><td>29</td><td>84</td><td>5</td><td>22</td><td>18</td><td>72</td></tr> <tr><td>15</td><td>47</td><td>26</td><td>53</td><td>79</td><td>55</td><td>27</td><td>30</td><td>37</td><td>118</td><td>113</td><td>125</td><td>94</td><td>107</td><td>20</td><td>53</td><td>17</td><td>102</td><td>8</td><td>19</td><td>5</td><td>87</td><td>123</td><td>29</td><td>45</td></tr> <tr><td>101</td><td>12</td><td>3</td><td>120</td><td>66</td><td>16</td><td>77</td><td>79</td><td>29</td><td>68</td><td>114</td><td>2</td><td>13</td><td>40</td><td>71</td><td>89</td><td>58</td><td>116</td><td>66</td><td>78</td><td>111</td><td>65</td><td>69</td><td>116</td><td>46</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p>State setelah solving:</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					76	119	89	99	43	35	42	122	59	41	90	31	82	6	100	38	25	63	108	61	4	10	85	73	117	90	112	54	21	32	95	62	103	98	44	60	92	106	104	81	97	24	50	33	74	105	55	64	115	49	7	82	23	14	91	36	57	11	121	79	1	109	83	34	48	85	51	96	124	67	29	84	5	22	18	72	15	47	26	53	79	55	27	30	37	118	113	125	94	107	20	53	17	102	8	19	5	87	123	29	45	101	12	3	120	66	16	77	79	29	68	114	2	13	40	71	89	58	116	66	78	111	65	69	116	46																																																																																																																																																						
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
76	119	89	99	43	35	42	122	59	41	90	31	82	6	100	38	25	63	108	61	4	10	85	73	117	90																																																																																																																																																																																																																																																																																						
112	54	21	32	95	62	103	98	44	60	92	106	104	81	97	24	50	33	74	105	55	64	115	49	7																																																																																																																																																																																																																																																																																							
82	23	14	91	36	57	11	121	79	1	109	83	34	48	85	51	96	124	67	29	84	5	22	18	72																																																																																																																																																																																																																																																																																							
15	47	26	53	79	55	27	30	37	118	113	125	94	107	20	53	17	102	8	19	5	87	123	29	45																																																																																																																																																																																																																																																																																							
101	12	3	120	66	16	77	79	29	68	114	2	13	40	71	89	58	116	66	78	111	65	69	116	46																																																																																																																																																																																																																																																																																							



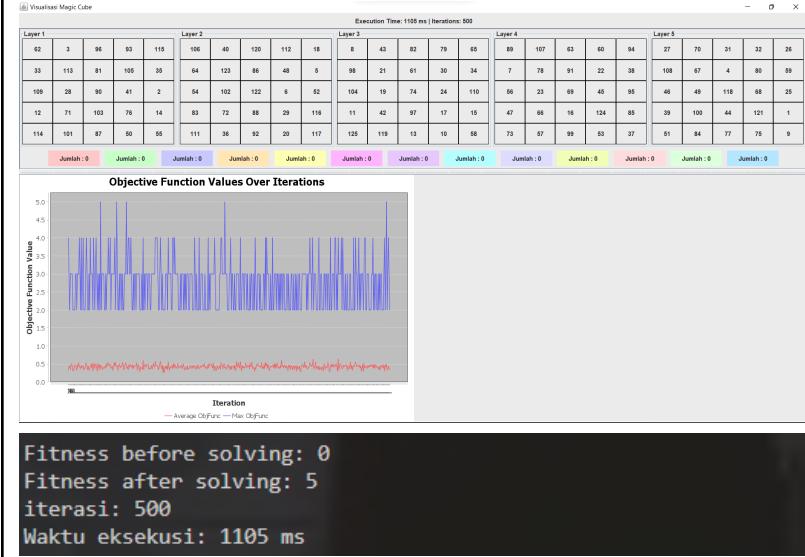


			<p><b>Execution Time: 18832 ms   Iterations: 1000</b></p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>59</td><td>16</td><td>101</td><td>50</td><td>57</td><td>102</td><td>97</td><td>18</td><td>2</td><td>48</td><td>47</td><td>103</td><td>90</td><td>95</td><td>26</td><td>23</td><td>62</td><td>32</td><td>75</td><td>108</td><td>6</td><td>78</td><td>106</td><td>98</td><td>82</td></tr> <tr><td>111</td><td>39</td><td>14</td><td>94</td><td>110</td><td>17</td><td>118</td><td>78</td><td>21</td><td>30</td><td>48</td><td>130</td><td>69</td><td>42</td><td>27</td><td>60</td><td>104</td><td>69</td><td>54</td><td>46</td><td>123</td><td>122</td><td>88</td><td>38</td><td>15</td></tr> <tr><td>113</td><td>100</td><td>19</td><td>92</td><td>68</td><td>23</td><td>105</td><td>40</td><td>3</td><td>73</td><td>67</td><td>55</td><td>116</td><td>56</td><td>28</td><td>84</td><td>121</td><td>38</td><td>98</td><td>22</td><td>72</td><td>33</td><td>51</td><td>91</td><td>45</td></tr> <tr><td>13</td><td>37</td><td>78</td><td>129</td><td>10</td><td>108</td><td>34</td><td>119</td><td>89</td><td>8</td><td>1</td><td>74</td><td>41</td><td>9</td><td>61</td><td>107</td><td>112</td><td>43</td><td>87</td><td>91</td><td>52</td><td>63</td><td>77</td><td>85</td><td>70</td></tr> <tr><td>71</td><td>89</td><td>4</td><td>66</td><td>12</td><td>7</td><td>64</td><td>83</td><td>20</td><td>24</td><td>95</td><td>117</td><td>44</td><td>124</td><td>58</td><td>29</td><td>31</td><td>11</td><td>56</td><td>114</td><td>115</td><td>93</td><td>35</td><td>53</td><td>5</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p><b>Objective Function Values Over Iterations</b></p> <p>Average ObjFunc — Max ObjFunc</p> <p><b>Fitness before solving:</b> 0  <b>Fitness after solving:</b> 6  <b>iterasi:</b> 1000  <b>Waktu eksekusi:</b> 18832 ms</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					59	16	101	50	57	102	97	18	2	48	47	103	90	95	26	23	62	32	75	108	6	78	106	98	82	111	39	14	94	110	17	118	78	21	30	48	130	69	42	27	60	104	69	54	46	123	122	88	38	15	113	100	19	92	68	23	105	40	3	73	67	55	116	56	28	84	121	38	98	22	72	33	51	91	45	13	37	78	129	10	108	34	119	89	8	1	74	41	9	61	107	112	43	87	91	52	63	77	85	70	71	89	4	66	12	7	64	83	20	24	95	117	44	124	58	29	31	11	56	114	115	93	35	53	5																																																																																																																																																						
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
59	16	101	50	57	102	97	18	2	48	47	103	90	95	26	23	62	32	75	108	6	78	106	98	82																																																																																																																																																																																																																																																																																							
111	39	14	94	110	17	118	78	21	30	48	130	69	42	27	60	104	69	54	46	123	122	88	38	15																																																																																																																																																																																																																																																																																							
113	100	19	92	68	23	105	40	3	73	67	55	116	56	28	84	121	38	98	22	72	33	51	91	45																																																																																																																																																																																																																																																																																							
13	37	78	129	10	108	34	119	89	8	1	74	41	9	61	107	112	43	87	91	52	63	77	85	70																																																																																																																																																																																																																																																																																							
71	89	4	66	12	7	64	83	20	24	95	117	44	124	58	29	31	11	56	114	115	93	35	53	5																																																																																																																																																																																																																																																																																							
			<p><b>State sebelum solving:</b></p> <p><b>Execution Time: 18864 ms   Iterations: 1000</b></p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>5</td><td>118</td><td>54</td><td>125</td><td>17</td><td>79</td><td>1</td><td>73</td><td>83</td><td>2</td><td>109</td><td>111</td><td>39</td><td>103</td><td>108</td><td>123</td><td>78</td><td>101</td><td>32</td><td>77</td><td>41</td><td>90</td><td>12</td><td>99</td><td>7</td></tr> <tr><td>16</td><td>8</td><td>19</td><td>78</td><td>33</td><td>13</td><td>104</td><td>31</td><td>57</td><td>94</td><td>22</td><td>20</td><td>74</td><td>43</td><td>37</td><td>60</td><td>70</td><td>72</td><td>47</td><td>89</td><td>97</td><td>112</td><td>67</td><td>88</td><td>116</td></tr> <tr><td>15</td><td>51</td><td>23</td><td>42</td><td>95</td><td>98</td><td>93</td><td>58</td><td>30</td><td>96</td><td>82</td><td>53</td><td>27</td><td>6</td><td>87</td><td>124</td><td>34</td><td>109</td><td>49</td><td>25</td><td>100</td><td>48</td><td>86</td><td>61</td><td>120</td></tr> <tr><td>110</td><td>63</td><td>11</td><td>80</td><td>3</td><td>62</td><td>5</td><td>16</td><td>50</td><td>36</td><td>115</td><td>65</td><td>28</td><td>122</td><td>29</td><td>121</td><td>119</td><td>24</td><td>10</td><td>103</td><td>89</td><td>59</td><td>84</td><td>64</td><td>38</td></tr> <tr><td>56</td><td>68</td><td>75</td><td>69</td><td>4</td><td>81</td><td>92</td><td>44</td><td>14</td><td>26</td><td>117</td><td>114</td><td>45</td><td>107</td><td>35</td><td>91</td><td>105</td><td>55</td><td>46</td><td>40</td><td>113</td><td>12</td><td>21</td><td>71</td><td>66</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p><b>State setelah solving:</b></p> <p><b>Execution Time: 18864 ms   Iterations: 1000</b></p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>5</td><td>118</td><td>54</td><td>125</td><td>17</td><td>79</td><td>26</td><td>73</td><td>83</td><td>112</td><td>109</td><td>111</td><td>39</td><td>102</td><td>108</td><td>105</td><td>57</td><td>101</td><td>32</td><td>77</td><td>41</td><td>90</td><td>12</td><td>106</td><td>85</td></tr> <tr><td>16</td><td>8</td><td>19</td><td>78</td><td>33</td><td>13</td><td>104</td><td>31</td><td>119</td><td>94</td><td>22</td><td>20</td><td>74</td><td>43</td><td>37</td><td>60</td><td>70</td><td>73</td><td>47</td><td>89</td><td>97</td><td>2</td><td>67</td><td>88</td><td>30</td></tr> <tr><td>15</td><td>51</td><td>23</td><td>82</td><td>95</td><td>98</td><td>93</td><td>58</td><td>116</td><td>48</td><td>82</td><td>53</td><td>27</td><td>6</td><td>87</td><td>124</td><td>10</td><td>99</td><td>49</td><td>25</td><td>4</td><td>48</td><td>96</td><td>61</td><td>120</td></tr> <tr><td>110</td><td>63</td><td>7</td><td>80</td><td>3</td><td>8</td><td>5</td><td>18</td><td>90</td><td>36</td><td>115</td><td>65</td><td>28</td><td>122</td><td>29</td><td>121</td><td>78</td><td>24</td><td>34</td><td>103</td><td>11</td><td>59</td><td>84</td><td>64</td><td>38</td></tr> <tr><td>56</td><td>68</td><td>75</td><td>69</td><td>100</td><td>81</td><td>92</td><td>44</td><td>14</td><td>1</td><td>117</td><td>114</td><td>86</td><td>107</td><td>35</td><td>91</td><td>42</td><td>55</td><td>46</td><td>66</td><td>113</td><td>123</td><td>21</td><td>71</td><td>40</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p><b>Objective Function Values Over Iterations</b></p> <p>Average ObjFunc — Max ObjFunc</p> <p><b>Fitness before solving:</b> 0  <b>Fitness after solving:</b> 6  <b>iterasi:</b> 1000  <b>Waktu eksekusi:</b> 18864 ms</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					5	118	54	125	17	79	1	73	83	2	109	111	39	103	108	123	78	101	32	77	41	90	12	99	7	16	8	19	78	33	13	104	31	57	94	22	20	74	43	37	60	70	72	47	89	97	112	67	88	116	15	51	23	42	95	98	93	58	30	96	82	53	27	6	87	124	34	109	49	25	100	48	86	61	120	110	63	11	80	3	62	5	16	50	36	115	65	28	122	29	121	119	24	10	103	89	59	84	64	38	56	68	75	69	4	81	92	44	14	26	117	114	45	107	35	91	105	55	46	40	113	12	21	71	66	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					5	118	54	125	17	79	26	73	83	112	109	111	39	102	108	105	57	101	32	77	41	90	12	106	85	16	8	19	78	33	13	104	31	119	94	22	20	74	43	37	60	70	73	47	89	97	2	67	88	30	15	51	23	82	95	98	93	58	116	48	82	53	27	6	87	124	10	99	49	25	4	48	96	61	120	110	63	7	80	3	8	5	18	90	36	115	65	28	122	29	121	78	24	34	103	11	59	84	64	38	56	68	75	69	100	81	92	44	14	1	117	114	86	107	35	91	42	55	46	66	113	123	21	71	40
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
5	118	54	125	17	79	1	73	83	2	109	111	39	103	108	123	78	101	32	77	41	90	12	99	7																																																																																																																																																																																																																																																																																							
16	8	19	78	33	13	104	31	57	94	22	20	74	43	37	60	70	72	47	89	97	112	67	88	116																																																																																																																																																																																																																																																																																							
15	51	23	42	95	98	93	58	30	96	82	53	27	6	87	124	34	109	49	25	100	48	86	61	120																																																																																																																																																																																																																																																																																							
110	63	11	80	3	62	5	16	50	36	115	65	28	122	29	121	119	24	10	103	89	59	84	64	38																																																																																																																																																																																																																																																																																							
56	68	75	69	4	81	92	44	14	26	117	114	45	107	35	91	105	55	46	40	113	12	21	71	66																																																																																																																																																																																																																																																																																							
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
5	118	54	125	17	79	26	73	83	112	109	111	39	102	108	105	57	101	32	77	41	90	12	106	85																																																																																																																																																																																																																																																																																							
16	8	19	78	33	13	104	31	119	94	22	20	74	43	37	60	70	73	47	89	97	2	67	88	30																																																																																																																																																																																																																																																																																							
15	51	23	82	95	98	93	58	116	48	82	53	27	6	87	124	10	99	49	25	4	48	96	61	120																																																																																																																																																																																																																																																																																							
110	63	7	80	3	8	5	18	90	36	115	65	28	122	29	121	78	24	34	103	11	59	84	64	38																																																																																																																																																																																																																																																																																							
56	68	75	69	100	81	92	44	14	1	117	114	86	107	35	91	42	55	46	66	113	123	21	71	40																																																																																																																																																																																																																																																																																							
Iterasi = 500	Populasi = 100	a	<p><b>State sebelum solving:</b></p> <p><b>Execution Time: 1138 ms   Iterations: 600</b></p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>1</td><td>116</td><td>77</td><td>102</td><td>118</td><td>2</td><td>57</td><td>78</td><td>78</td><td>104</td><td>96</td><td>38</td><td>124</td><td>75</td><td>120</td><td>82</td><td>6</td><td>107</td><td>9</td><td>66</td><td>43</td><td>15</td><td>39</td><td>47</td><td>41</td></tr> <tr><td>60</td><td>46</td><td>19</td><td>34</td><td>22</td><td>92</td><td>99</td><td>86</td><td>16</td><td>106</td><td>89</td><td>83</td><td>7</td><td>6</td><td>18</td><td>65</td><td>00</td><td>80</td><td>29</td><td>12</td><td>11</td><td>47</td><td>84</td><td>69</td><td>68</td></tr> <tr><td>122</td><td>54</td><td>115</td><td>111</td><td>119</td><td>90</td><td>10</td><td>21</td><td>27</td><td>30</td><td>103</td><td>40</td><td>31</td><td>101</td><td>22</td><td>25</td><td>8</td><td>109</td><td>32</td><td>14</td><td>71</td><td>84</td><td>53</td><td>51</td><td>4</td></tr> <tr><td>125</td><td>63</td><td>17</td><td>13</td><td>79</td><td>84</td><td>108</td><td>55</td><td>20</td><td>88</td><td>105</td><td>121</td><td>87</td><td>64</td><td>44</td><td>3</td><td>82</td><td>42</td><td>81</td><td>100</td><td>110</td><td>35</td><td>26</td><td>112</td><td>28</td></tr> <tr><td>21</td><td>125</td><td>26</td><td>90</td><td>112</td><td>93</td><td>14</td><td>24</td><td>61</td><td>98</td><td>20</td><td>58</td><td>117</td><td>27</td><td>49</td><td>72</td><td>45</td><td>23</td><td>52</td><td>73</td><td>70</td><td>62</td><td>114</td><td>97</td><td>48</td></tr> </tbody> </table> <p>Jumlah : 0   Jumlah : 0</p> <p><b>State setelah solving:</b></p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					1	116	77	102	118	2	57	78	78	104	96	38	124	75	120	82	6	107	9	66	43	15	39	47	41	60	46	19	34	22	92	99	86	16	106	89	83	7	6	18	65	00	80	29	12	11	47	84	69	68	122	54	115	111	119	90	10	21	27	30	103	40	31	101	22	25	8	109	32	14	71	84	53	51	4	125	63	17	13	79	84	108	55	20	88	105	121	87	64	44	3	82	42	81	100	110	35	26	112	28	21	125	26	90	112	93	14	24	61	98	20	58	117	27	49	72	45	23	52	73	70	62	114	97	48																																																																																																																																																						
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																											
1	116	77	102	118	2	57	78	78	104	96	38	124	75	120	82	6	107	9	66	43	15	39	47	41																																																																																																																																																																																																																																																																																							
60	46	19	34	22	92	99	86	16	106	89	83	7	6	18	65	00	80	29	12	11	47	84	69	68																																																																																																																																																																																																																																																																																							
122	54	115	111	119	90	10	21	27	30	103	40	31	101	22	25	8	109	32	14	71	84	53	51	4																																																																																																																																																																																																																																																																																							
125	63	17	13	79	84	108	55	20	88	105	121	87	64	44	3	82	42	81	100	110	35	26	112	28																																																																																																																																																																																																																																																																																							
21	125	26	90	112	93	14	24	61	98	20	58	117	27	49	72	45	23	52	73	70	62	114	97	48																																																																																																																																																																																																																																																																																							

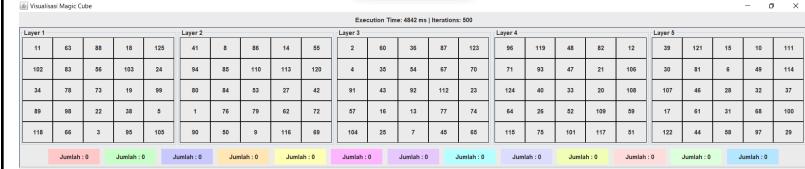


Populasi  
= 500

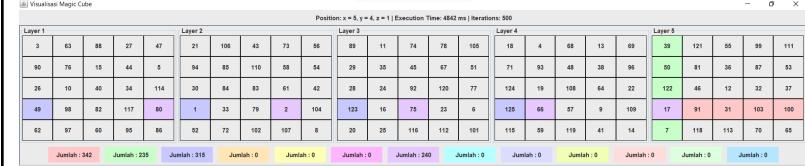
a



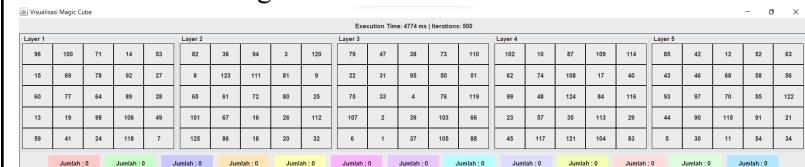
State sebelum solving:



State setelah solving:



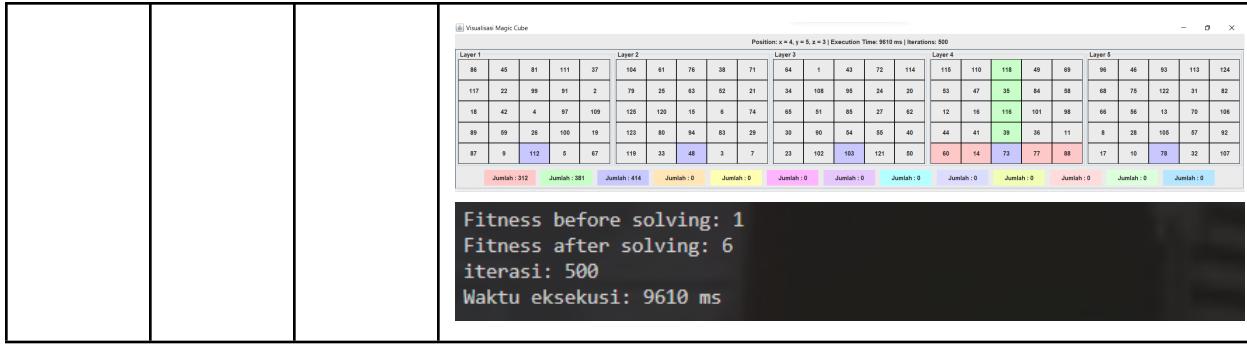
State sebelum solving:



State setelah solving:

		c	<p>State sebelum solving:</p> <p>State setelah solving:</p>
Populasi = 1000	a		<p>State sebelum solving:</p> <p>State setelah solving:</p>

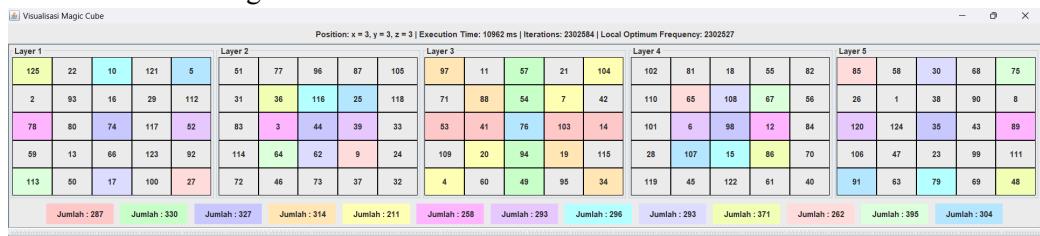


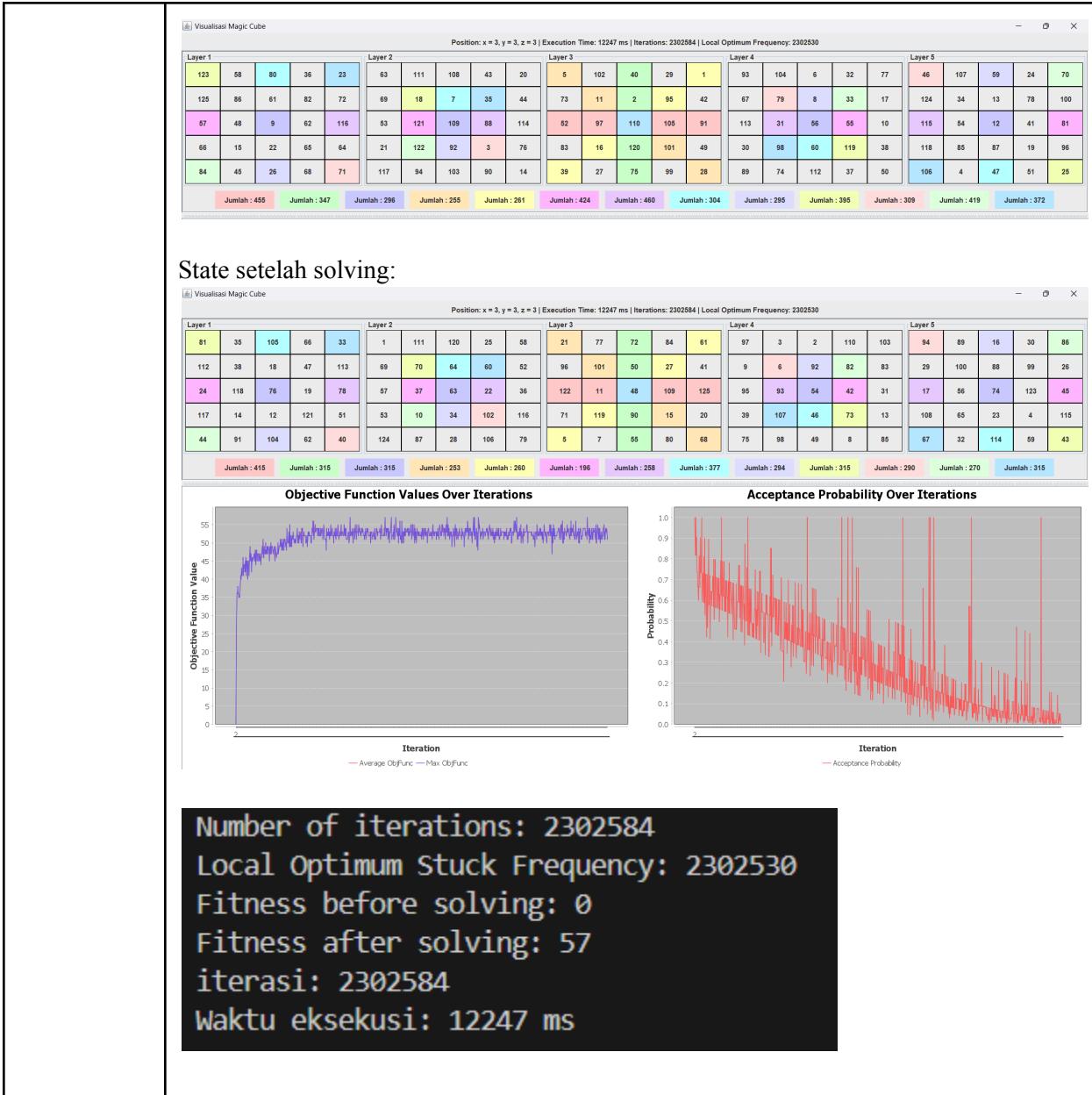


### c. Algoritma Simulated Annealing

Percobaan	Hasil																																																																																																																																																																																																																																																																																																												
1	<p>State sebelum solving:</p> <p>Visualisasi Magic Cube</p> <p>Position: x = 3, y = 3, z = 3   Execution Time: 10778 ms   Iterations: 2302584   Local Optimum Frequency: 2302529</p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>39</td><td>54</td><td>125</td><td>119</td><td>74</td><td>105</td><td>32</td><td>49</td><td>17</td><td>62</td><td>33</td><td>117</td><td>3</td><td>94</td><td>12</td><td>25</td><td>36</td><td>53</td><td>78</td><td>120</td><td>90</td><td>104</td><td>27</td><td>114</td><td>23</td></tr> <tr><td>30</td><td>72</td><td>115</td><td>4</td><td>71</td><td>86</td><td>16</td><td>112</td><td>61</td><td>111</td><td>21</td><td>113</td><td>26</td><td>6</td><td>42</td><td>48</td><td>10</td><td>34</td><td>43</td><td>37</td><td>80</td><td>107</td><td>56</td><td>75</td><td>69</td></tr> <tr><td>59</td><td>31</td><td>60</td><td>83</td><td>20</td><td>81</td><td>89</td><td>7</td><td>57</td><td>18</td><td>67</td><td>9</td><td>51</td><td>35</td><td>41</td><td>65</td><td>82</td><td>38</td><td>84</td><td>1</td><td>77</td><td>88</td><td>91</td><td>102</td><td>110</td></tr> <tr><td>11</td><td>73</td><td>123</td><td>92</td><td>108</td><td>45</td><td>100</td><td>52</td><td>8</td><td>13</td><td>103</td><td>88</td><td>24</td><td>55</td><td>15</td><td>14</td><td>66</td><td>70</td><td>106</td><td>98</td><td>50</td><td>46</td><td>40</td><td>76</td><td>93</td></tr> <tr><td>116</td><td>19</td><td>95</td><td>44</td><td>97</td><td>58</td><td>29</td><td>85</td><td>109</td><td>122</td><td>63</td><td>47</td><td>96</td><td>64</td><td>2</td><td>5</td><td>101</td><td>121</td><td>22</td><td>87</td><td>79</td><td>28</td><td>118</td><td>124</td><td>99</td></tr> </tbody> </table> <p>Jumlah : 203   Jumlah : 200   Jumlah : 247   Jumlah : 254   Jumlah : 220   Jumlah : 393   Jumlah : 287   Jumlah : 476   Jumlah : 259   Jumlah : 311   Jumlah : 256   Jumlah : 333   Jumlah : 331</p> <p>State setelah solving:</p> <p>Visualisasi Magic Cube</p> <p>Position: x = 3, y = 3, z = 3   Execution Time: 10778 ms   Iterations: 2302584   Local Optimum Frequency: 2302529</p> <table border="1"> <thead> <tr> <th colspan="5">Layer 1</th> <th colspan="5">Layer 2</th> <th colspan="5">Layer 3</th> <th colspan="5">Layer 4</th> <th colspan="5">Layer 5</th> </tr> </thead> <tbody> <tr><td>5</td><td>117</td><td>110</td><td>92</td><td>30</td><td>125</td><td>27</td><td>6</td><td>41</td><td>116</td><td>48</td><td>31</td><td>56</td><td>115</td><td>65</td><td>93</td><td>106</td><td>98</td><td>46</td><td>7</td><td>44</td><td>108</td><td>45</td><td>21</td><td>97</td></tr> <tr><td>124</td><td>24</td><td>4</td><td>64</td><td>58</td><td>96</td><td>47</td><td>105</td><td>17</td><td>50</td><td>9</td><td>82</td><td>83</td><td>80</td><td>61</td><td>57</td><td>53</td><td>28</td><td>94</td><td>113</td><td>29</td><td>109</td><td>95</td><td>49</td><td>33</td></tr> <tr><td>78</td><td>39</td><td>52</td><td>25</td><td>120</td><td>14</td><td>102</td><td>51</td><td>62</td><td>59</td><td>112</td><td>23</td><td>11</td><td>63</td><td>118</td><td>78</td><td>101</td><td>86</td><td>42</td><td>8</td><td>32</td><td>3</td><td>36</td><td>123</td><td>10</td></tr> <tr><td>91</td><td>60</td><td>114</td><td>38</td><td>12</td><td>19</td><td>73</td><td>77</td><td>74</td><td>72</td><td>100</td><td>89</td><td>84</td><td>37</td><td>13</td><td>2</td><td>40</td><td>43</td><td>111</td><td>119</td><td>103</td><td>26</td><td>87</td><td>55</td><td>99</td></tr> <tr><td>16</td><td>75</td><td>35</td><td>85</td><td>104</td><td>34</td><td>66</td><td>78</td><td>121</td><td>18</td><td>70</td><td>90</td><td>81</td><td>20</td><td>54</td><td>88</td><td>15</td><td>122</td><td>22</td><td>68</td><td>107</td><td>69</td><td>1</td><td>67</td><td>71</td></tr> </tbody> </table> <p>Jumlah : 327   Jumlah : 315   Jumlah : 236   Jumlah : 232   Jumlah : 315   Jumlah : 244   Jumlah : 326   Jumlah : 270   Jumlah : 196   Jumlah : 245   Jumlah : 286   Jumlah : 291   Jumlah : 205</p> <p>Objective Function Values Over Iterations</p> <p>Average ObjFunc   Max ObjFunc</p> <p>Acceptance Probability Over Iterations</p> <p>Acceptance Probability</p>	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					39	54	125	119	74	105	32	49	17	62	33	117	3	94	12	25	36	53	78	120	90	104	27	114	23	30	72	115	4	71	86	16	112	61	111	21	113	26	6	42	48	10	34	43	37	80	107	56	75	69	59	31	60	83	20	81	89	7	57	18	67	9	51	35	41	65	82	38	84	1	77	88	91	102	110	11	73	123	92	108	45	100	52	8	13	103	88	24	55	15	14	66	70	106	98	50	46	40	76	93	116	19	95	44	97	58	29	85	109	122	63	47	96	64	2	5	101	121	22	87	79	28	118	124	99	Layer 1					Layer 2					Layer 3					Layer 4					Layer 5					5	117	110	92	30	125	27	6	41	116	48	31	56	115	65	93	106	98	46	7	44	108	45	21	97	124	24	4	64	58	96	47	105	17	50	9	82	83	80	61	57	53	28	94	113	29	109	95	49	33	78	39	52	25	120	14	102	51	62	59	112	23	11	63	118	78	101	86	42	8	32	3	36	123	10	91	60	114	38	12	19	73	77	74	72	100	89	84	37	13	2	40	43	111	119	103	26	87	55	99	16	75	35	85	104	34	66	78	121	18	70	90	81	20	54	88	15	122	22	68	107	69	1	67	71
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																									
39	54	125	119	74	105	32	49	17	62	33	117	3	94	12	25	36	53	78	120	90	104	27	114	23																																																																																																																																																																																																																																																																																					
30	72	115	4	71	86	16	112	61	111	21	113	26	6	42	48	10	34	43	37	80	107	56	75	69																																																																																																																																																																																																																																																																																					
59	31	60	83	20	81	89	7	57	18	67	9	51	35	41	65	82	38	84	1	77	88	91	102	110																																																																																																																																																																																																																																																																																					
11	73	123	92	108	45	100	52	8	13	103	88	24	55	15	14	66	70	106	98	50	46	40	76	93																																																																																																																																																																																																																																																																																					
116	19	95	44	97	58	29	85	109	122	63	47	96	64	2	5	101	121	22	87	79	28	118	124	99																																																																																																																																																																																																																																																																																					
Layer 1					Layer 2					Layer 3					Layer 4					Layer 5																																																																																																																																																																																																																																																																																									
5	117	110	92	30	125	27	6	41	116	48	31	56	115	65	93	106	98	46	7	44	108	45	21	97																																																																																																																																																																																																																																																																																					
124	24	4	64	58	96	47	105	17	50	9	82	83	80	61	57	53	28	94	113	29	109	95	49	33																																																																																																																																																																																																																																																																																					
78	39	52	25	120	14	102	51	62	59	112	23	11	63	118	78	101	86	42	8	32	3	36	123	10																																																																																																																																																																																																																																																																																					
91	60	114	38	12	19	73	77	74	72	100	89	84	37	13	2	40	43	111	119	103	26	87	55	99																																																																																																																																																																																																																																																																																					
16	75	35	85	104	34	66	78	121	18	70	90	81	20	54	88	15	122	22	68	107	69	1	67	71																																																																																																																																																																																																																																																																																					

	<p><b>Number of iterations:</b> 2302584  <b>Local Optimum Stuck Frequency:</b> 2302529  <b>Fitness before solving:</b> 0  <b>Fitness after solving:</b> 56  <b>iterasi:</b> 2302584  <b>Waktu eksekusi:</b> 10778 ms</p>
--	--

2	<p><b>State sebelum solving:</b></p>  <p><b>State setelah solving:</b></p>  <p><b>Number of iterations:</b> 2302584  <b>Local Optimum Stuck Frequency:</b> 2302527  <b>Fitness before solving:</b> 0  <b>Fitness after solving:</b> 58  <b>iterasi:</b> 2302584  <b>Waktu eksekusi:</b> 10962 ms</p>
3	<p><b>State sebelum solving:</b></p>



Dari hasil eksperimen, genetic algorithm memiliki hasil solusi yang sangat jauh dari global optima. Hal ini disebabkan oleh pembangkitan populasinya yang dilakukan secara acak, dan ketika dilakukan random selection hingga crossover, hasilnya tidak dijamin mendekati global optima.

Kemudian, algoritma hill climbing with sideways move menunjukkan solusi yang cenderung lebih mendekati, namun tidak sampai setengah dari state value global optima. Hal ini disebabkan oleh algoritma Hill Climbing yang cukup tergantung pada titik awal dan banyaknya iterasi yang dilakukan. Ketika algoritma Hill Climbing dimulai dengan state awal yang cukup ‘beruntung’, maka akan mempermudah langkah selanjutnya dan meningkatkan kemungkinan untuk menemukan global optimum. Selain itu, permasalahan Magic Cube juga memiliki kemungkinan yang cukup besar untuk terjebak di

dalam local optimum, sehingga keberuntungan pada State awal akan sangat membantu untuk keluar dari plateau.

Setiap kali dijalankan, algoritma Hill Climbing with Sideways Move menghasilkan Magic Cube dengan nilai objective function sekitar 40 - 45. Dari beberapa kali percobaan, hasil Magic Cube cukup konsisten dan dieksekusi dalam durasi yang mirip. Durasi dari eksekusi algoritma Hill Climbing with Sideways Move ini rata-rata menghabiskan waktu sekitar 20 detik atau lebih hingga akhirnya berhenti karena sudah mencapai batas banyak iterasi sideways moves yang diatur yaitu 500.

Perbedaan parameter jumlah sideway moves maksimal berpengaruh pada runtime dan keakuratan program. Semakin banyak sideway moves yang diizinkan, maka program akan semakin akurat dan mendekati nilai maksimal hingga mencapai local optimal tanpa tetangga yang bernilai lesih. Akan tetapi, runtime program akan menjadi lebih lambat secara signifikan. Kami berekspresimen dan menentukan 500 sebagai jumlah sideway moves maksimal. Angka 500 konsisten menghasilkan hasil yang sama dengan angka sideway moves lainnya, sembari memiliki runtime yang relatif singkat dan tetap menjaga konsistensi.

Berbeda dengan algoritma lainnya, algoritma Simulated Annealing (SA) cenderung menghasilkan solusi yang paling mendekati global optimum, dengan state value sering mencapai lebih dari setengah nilai global optimum. Pendekatan ini mengandalkan peluang berbasis suhu untuk mengizinkan pergerakan ke solusi yang lebih buruk, terutama pada tahap awal eksplorasi, sehingga mencegah algoritma terjebak di lokal optimum. Seiring waktu, suhu menurun dan peluang menerima solusi buruk berkurang, mendorong eksplorasi solusi terbaik yang ditemukan. Dengan banyak iterasi, SA semakin mendekati solusi optimal karena eksplorasi berangsur-angsur dikurangi, menjaga keseimbangan eksplorasi dan eksplotasi. Ini membuat SA efektif dan fleksibel dalam menghadapi masalah optimasi kompleks. Dari banyak percobaan yang dilakukan diluar eksperimen di atas, SA konsisten akan hasil solusi state value nya di rentang 54 hingga 63 (sekitar 57,8% mendekati global optimum). Akan tetapi, algoritma ini juga tetap stuck di lokal optimum dan belum pernah bisa mencapai state value lebih dari 70. Ada beberapa kemungkinan penyebab akan terjadinya hal ini:

- Parameter Tidak Tepat

Parameter suhu awal dan cooling rate mungkin tidak cukup untuk mengeksplorasi ruang solusi secara optimal. Suhu awal yang terlalu rendah atau cooling rate yang terlalu cepat dapat membatasi kemampuan algoritma untuk menerima solusi buruk yang dibutuhkan untuk keluar dari lokal optimum.

- Landscape Fungsi Objektif

Jika fungsi objektif memiliki banyak puncak lokal yang sangat menonjol dan tidak cukup perbedaan dengan solusi global, SA mungkin kesulitan untuk keluar dari jebakan lokal optimum tersebut.

- Distribusi Solusi Buruk

Mungkin solusi yang lebih buruk di sekitar lokal optimum tidak cukup buruk sehingga peluang untuk keluar dari lokal optimum rendah. Ini menyebabkan algoritma menjadi cepat konvergen sebelum mencapai solusi global.

- Kriteria Penerimaan

Batas probabilitas solusi mungkin perlu disesuaikan agar algoritma dapat menerima solusi buruk dalam jumlah yang memadai. Batas probabilitas yang terlalu ketat dapat menyebabkan algoritma SA sulit menjelajah area di sekitar lokal optimum.

Berdasarkan hasil eksperimen, tiap algoritma memiliki durasi yang berbeda-beda. Namun, durasi setiap program ini tidak bisa dibandingkan secara langsung karena adanya faktor atau variabel yang tidak dikontrol, seperti parameter tiap algoritma, sehingga durasi waktu (dalam hal ini sebagai variabel terikat) tidak bisa dikaitkan dengan algoritma (dalam hal ini sebagai variabel bebas) untuk dibandingkan dari satu algoritma ke algoritma lain.

Algoritma Hill Climbing dengan Sideway Moves dan Simulated Annealing menghasilkan nilai objective function yang cukup konsisten, berada dalam kisaran 40-60. Sebaliknya, Genetic Algorithm hanya mencapai nilai objective function sebesar 6, yang menunjukkan performa yang jauh lebih rendah. Hal ini menunjukkan bahwa Genetic Algorithm kurang cocok untuk skenario ini karena pola pencarian solusinya tidak dapat mengandalkan mekanisme crossover untuk menemukan solusi optimal, terutama dalam menghadapi pola khusus yang dibutuhkan dalam masalah ini.

Pada Genetic Algorithm, banyaknya iterasi maksimal dan jumlah populasi berpengaruh terhadap hasil pencarian, meskipun pada penerapannya, pengaruhnya tidak begitu signifikan. Jumlah populasi berkontribusi dalam memperkaya kemungkinan-kemungkinan dalam mencari jalan pada solution space. Banyaknya iterasi dapat memberi kesempatan lebih banyak untuk algoritma dalam melakukan eksplorasi dari gen-gen yang dihasilkan.

# BAB III

## KESIMPULAN DAN SARAN

### 1. Kesimpulan

Pada tugas besar ini, kami telah berhasil mengimplementasikan algoritma local search dalam menyelesaikan persoalan *magic cube*. Dengan pendekatan ini, kami hanya mampu mendapatkan solusi yang 57,8% mendekati global optimum yang memenuhi syarat-syarat dari suatu *magic cube*. Algoritma local search yang digunakan dapat meminimalkan jumlah langkah yang diperlukan, serta menunjukkan efisiensi dan efektivitas dalam mencapai hasil yang diinginkan.

Dari ketiga algoritma yang telah kami implementasikan, kami menemukan bahwa Algoritma Simulated Annealing merupakan algoritma yang terbaik untuk menyelesaikan permasalahan ini. Hal ini karena dari hasil pengujian yang telah dilakukan, Algoritma Simulated Annealing rata-rata berhasil menyusun Magic Cube dengan nilai objective function lebih tinggi daripada algoritma Genetic dan Hill Climbing with Side Move. Sehingga dapat disimpulkan bahwa algoritma Simulated Annealing dapat mendekati kondisi Magic Cube yang paling mendekati optimal.

### 2. Saran

Terdapat beberapa saran yang ingin kami sampaikan dalam penggerjaan tugas besar ini, yaitu:

1. Perlu dipertimbangkan kembali terkait pemilihan Objective Function dalam algoritma. Objective Function yang lebih relevan dan spesifik dengan karakteristik Magic Cube dapat meningkatkan kinerja algoritma untuk menghasilkan solusi yang lebih efisien dalam waktu dan iterasi yang lebih singkat.
2. Perlu dilakukan evaluasi menyeluruh terhadap pemilihan bahasa pemrograman Java dan struktur kode program Magic Cube Solver. Analisis mendalam terhadap kinerja, kemudahan pengembangan, dan skalabilitas bahasa lain akan membantu mengidentifikasi bahasa yang lebih sesuai untuk mencapai efisiensi optimal dalam pemecahan masalah Magic Cube. Selain itu, optimasi kode yang ada juga perlu dilakukan untuk meningkatkan keterbacaan dan mempermudah pemeliharaan program.

## **BAB IV**

### **PEMBAGIAN TUGAS**

<b>NIM</b>	<b>Nama</b>	<b>Tugas</b>
13522015	Yusuf Ardian Sandi	MagicCube Class, Algoritma Simulated Annealing, Main Class, Menu Class
13522018	Ibrahim Ihsan Rasyid	Kelas GeneticAlgorithm
13522031	Zaki Yudhistira Candra	Kelas HillClimbing
13522032	Tazkia Nizami	Visualisasi Magic Cube, Graph Data

## **BAB V**

## **REFERENSI**

- Link Github: <https://github.com/TazakiN/Tubes1-AI>
- Spesifikasi Tugas: [Spesifikasi Tugas Besar 1 IF3070 Dasar Inteligensi Artifisial & IF3170](#)  
[Inteligensi Artifisial 2024/2025](#)
- Magic Cube : [Features of the magic cube - Magisch vierkant](#)
- Salindia Kuliah Intelegensi Buatan : [Beyond Classical Search](#)