

Tugas Besar 1 IF3170 Inteligensi Artifisial

Pencarian Solusi Diagonal Magic Cube dengan Local Search



Disusun Oleh:

- Adril Putra Merin (13522068)
- Matthew Vladimir Hutabarat (13522093)
- Suthasoma Mahardhika Munthe (13522098)
- Berto Ricardo Togatorop (13522118)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

DAFTAR ISI

DAFTAR ISI.....	2
BAB I	
DESKRIPSI persoalan.....	3
BAB II	
FUNGSI objektif.....	4
A. Number of Slices.....	4
B. Sum of Absolute Differences.....	5
C. Sum of Squared Differences.....	5
BAB III	
ALGORITMA pencarian lokal.....	7
A. Steepest Ascent Hill-Climbing.....	14
B. Hill-climbing with Sideways Move.....	15
C. Hill-climbing with Random Restart.....	16
D. Stochastic Hill-Climbing.....	18
E. Simulated Annealing.....	19
F. Genetic Algorithm.....	21
BAB IV	
HASIL eksperimen.....	26
A. Steepest Ascent Hill-Climbing.....	26
B. Hill-Climbing with Sideways Move.....	29
C. Hill-Climbing with Random Restart.....	32
D. Stochastic Hill-Climbing.....	41
E. Simulated Annealing.....	45
F. Genetic Algorithm.....	49
BAB V	
ANALISIS DAN PEMBAHASAN.....	66
KESIMPULAN.....	68
REFERENSI.....	69
PEMBAGIAN TUGAS.....	70

BAB I

DESKRIPSI PERSOALAN

Diagonal magic cube adalah sebuah kubus dengan ukuran $n \times n \times n$ yang terdiri atas angka-angka dari 1 sampai n^3 tanpa pengulangan, dengan n adalah panjang sisi pada kubus tersebut. *Magic cube* tersebut memiliki n^2 buah baris, n^2 buah kolom, n^2 buah pilar, $6n$ buah diagonal bidang, dan 4 buah diagonal ruang yang memiliki jumlah elemen sama dengan *magic number*.

Jika sebuah kubus memiliki *magic number*, angka tersebut akan bersifat konstan yang diberikan oleh persamaan berikut:

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

(Eq. 1)

Bukti dari persamaan (1) adalah sebagai berikut. Misalkan sebuah kubus memiliki *magic number* dan sudah tersusun sedemikian rupa sehingga memenuhi syarat untuk disebut *magic cube*. Jumlah setiap elemen dari setiap elemen pada kubus tersebut adalah jumlah angka dari 1 sampai n^3 dimana elemen-elemen tersebut dapat dibagi ke dalam n^2 baris yang memiliki jumlah elemen sama dengan *magic number*. Dengan kata lain,

$$\begin{aligned} n^2 \times M(n) &= \frac{1}{2}n^3(n^3 + 1) \\ M(n) &= \frac{1}{2}n(n^3 + 1) \end{aligned}$$

Pada tugas ini kita akan mencoba menyelesaikan *diagonal magic cube* dengan ukuran $5 \times 5 \times 5$. 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).

Pada tahun 2003, C. Boyer and W. Trump menemukan *magic cube* dengan ukuran $5 \times 5 \times 5$ sehingga dijamin terdapat solusi yang dapat ditemukan oleh pencarian lokal. Berdasarkan persamaan (1), *magic number* dari kubus dengan ukuran sisi 5 adalah **315**.

Magic cube dengan ukuran sisi 5 memiliki 25 buah baris, 25 buah kolom, 25 buah pilar, 30 buah diagonal bidang, dan 4 buah diagonal ruang. Jadi, secara total, terdapat $25 + 25 + 25 + 30 + 4 = 109$ buah segmen yang harus memiliki jumlah elemen sama dengan *magic cube*.

Pada persoalan ini, setiap state diwakili oleh suatu larik yang berisi permutasi angka 1 hingga 125 yang ditempatkan ke dalam suatu kubus yang berukuran $5 \times 5 \times 5$. Jadi, objektif dari persoalan ini adalah mencari permutasi yang memenuhi setiap properti dari *diagonal magic cube*. Neighbor untuk setiap state didapatkan dengan menukar dua buah elemen pada larik state saat ini. Dengan demikian, setiap state memiliki $125C2 = \frac{1}{2}(125 \times 124) = 7750$ buah *neighbors*.

BAB II

FUNGSI OBJEKTIF

Fungsi objektif adalah fungsi yang ingin meminimumkan atau memaksimumkan nilainya berdasarkan kendala yang ada. Pada konteks pencarian lokal, fungsi objektif akan mengembalikan nilai dari suatu *state* berdasarkan batasan yang berlaku. Tujuan dari pencarian lokal sendiri adalah untuk memaksimumkan atau meminimumkan hasil fungsi objektif ini.

Pada persoalan ini, batasan yang didefinisikan adalah semua 109 potongan, meliputi baris, kolom, diagonal baris, dan diagonal ruang, harus memiliki jumlah elemen yang sama dengan 315. Kami mengajukan beberapa kandidat fungsi objektif dan akan memilih fungsi objektif terbaik di akhir bab ini.

A. Number of Slices

Kandidat fungsi objektif yang pertama adalah banyak potongan (dari 109 potongan) yang memenuhi batasan, yaitu memiliki jumlah elemen yang sama dengan 315. Pada fungsi objektif ini, tujuan kita adalah memaksimalkan banyak jumlah potongan yang memenuhi batasan tersebut.

Definisikan $h(s)$ sebagai fungsi objektif yang mengembalikan banyak potongan yang memiliki jumlah elemen yang sama dengan 315 pada suatu state. Secara matematis, fungsi objektif ini dapat didefinisikan sebagai berikut:

$$h(s) = \sum_{i=1}^{109} slice(i)$$

dimana $slice(i)$ akan bernilai 1 jika jumlah elemen pada potongan ke- i sama dengan 315, dan bernilai 0 jika sebaliknya. Dengan demikian nilai maksimum dari h adalah $h(s) = 109$. Adapun pseudocode dari fungsi objektif ini adalah sebagai berikut.

```
function OBJECTIVE(state) → integer
    cnt ← 0
    for all slice in state.CUBE:
        if sum_element(slice) = 315 then
            cnt ← cnt + 1
    return cnt
```

Perhatikan bahwa terdapat 125! (lebih dari 10^{209}) konfigurasi state yang mungkin, sedangkan banyak kemungkinan nilai h hanyalah 109 buah sehingga suatu state sangat mungkin memiliki nilai h yang sama dengan banyak state lainnya. Akibatnya, pencarian lokal akan memiliki kemungkinan yang sangat besar untuk terjebak pada *local maximum* atau *flat maximum local (plateau)*. Disamping itu, untuk beberapa algoritma pencarian, suatu *state* akan kesulitan untuk memutuskan *neighbor* yang terbaik untuk dipilih karena banyak *neighbor* dengan nilai h yang sama berjumlah sangat banyak. Hal ini tentu sangat buruk untuk proses pencarian lokal sehingga kami memutuskan tidak menggunakan fungsi objektif ini.

B. Sum of Absolute Differences

Pada kandidat ini, fungsi $g(s)$ didefinisikan sebagai fungsi objektif yang mengembalikan jumlah dari *absolute difference* antara jumlah elemen pada suatu potongan dengan 315 untuk setiap 109 potongan. Tujuan dari pencarian lokal adalah meminimumkan nilai dari fungsi objektif $g(s)$. Secara matematis, fungsi objektif ini dapat didefinisikan sebagai berikut:

$$g(s) = \sum_{i=1}^{109} |315 - \text{sum_element}(i)|$$

dimana $\text{sum_element}(i)$ adalah jumlah semua elemen pada potongan ke- i . Jadi, nilai minimum dari $g(s)$ adalah 0 yang merupakan tujuan utama dari pencarian yang akan dilakukan. Dengan menggunakan fungsi objektif g , kita mencoba mencari susunan kubus yang memiliki *sum of absolute differences* paling kecil. Secara kasar, berikut adalah implementasi fungsi objektif ini dalam pseudocode.

```
function OBJECTIVE(state) → integer
    sum ← 0
    for all slice in state.CUBE:
        sum ← sum + abs(sum_element(i) - 315)
    return sum
```

Perhatikan bahwa, nilai yang dihasilkan oleh fungsi objektif g lebih bervariasi dibandingkan dengan nilai yang dihasilkan oleh fungsi objektif h pada bagian sebelumnya karena kita tidak hanya menghitung banyak potongan kubus yang sudah memenuhi syarat, tetapi juga menghitung jumlah dari perbedaan antara tiap jumlah elemen potongan dengan *magic number*, yaitu 315. Dua buah state yang memiliki $h(s)$ yang sama mungkin saja memiliki nilai $g(s)$ yang berbeda karena alasan diatas. Dengan demikian, kita dapat mengurangi kemungkinan terjebak di *maximum local* dan *flat maximum local* yang sebelumnya dihasilkan oleh $h(s)$.

C. Sum of Squared Differences

Walaupun fungsi objektif $g(s)$ lebih baik daripada $h(s)$, kita masih memiliki kemungkinan yang cukup besar untuk terjebak di *maximum local* dan *plateau*. Untuk nilai *sum of absolute of differences* yang sama, kita ingin agar distribusi *absolute of differences* yang lebih merata, memiliki nilai yang lebih rendah. Namun, perbedaan ini tidak ada pada fungsi objektif $g(s)$. Untuk itu, fungsi objektif yang menjadi alternatif paling baik adalah *sum of squared differences*.

Didefinisikan fungsi $f(s)$ sebagai fungsi objektif yang mengembalikan jumlah dari squared differences antara jumlah elemen dengan 315 untuk setiap potongan. Tujuan dari pencarian lokal adalah meminimumkan nilai dari fungsi objektif $g(s)$. Secara matematis, fungsi objektif ini dapat dituliskan sebagai berikut:

$$f(s) = \sum_{i=1}^{109} (315 - \text{sum_element}(i))^2$$

Adapun implementasi fungsi objektif ini dalam bentuk pseudocode adalah sebagai berikut:

```
function OBJECTIVE(state) → integer
    sum ← 0
    for all slice in state.CUBE:
        sum ← sum + (sum_element(i) - 315) * (sum_element(i) - 315)
    return sum
```

Perbedaan utama antara fungsi objektif ini dengan fungsi objektif sebelumnya (*sum of absolute differences*) terdapat pada kuadrat antara perbedaan jumlah elemen pada potongan dengan 315. Hal ini dilakukan agar perbedaan antara jumlah elemen potongan dengan 315, untuk setiap potongan, yang memiliki distribusi lebih merata, memiliki nilai yang lebih rendah karena tujuan kita adalah minimisasi. Disamping itu, grafik fungsi $f(s)$ bersifat lebih “kontinu” dibandingkan $g(s)$ dan $h(s)$ yang mana lebih baik untuk melakukan pencarian lokal.

Karena alasan yang sudah dijelaskan sebelumnya, kami memutuskan untuk menggunakan *sum of squared differences* sebagai fungsi objektif yang digunakan pada tugas ini.

BAB III

ALGORITMA PENCARIAN LOKAL

Algoritma pencarian lokal adalah algoritma pencarian yang bekerja dengan mencari dari sebuah *start state* ke *neighbouring states* tanpa menyimpan himpunan dari setiap state yang sudah dikunjungi. Hal ini berarti, algoritma ini tidak sistematis karena mungkin saja pencarian tidak akan pernah mengunjungi bagian dari ruang pencarian dimana solusi utama berada. Namun, algoritma ini memiliki dua keuntungan utama dibandingkan *classical search*. Pertama, algoritma ini menggunakan sangat sedikit memori. Kedua, algoritma ini mampu mencari solusi pada ruang pencarian yang besar atau tak hingga dimana pencarian algoritma sistematis (*classical search*) tidak dapat.

Algoritma pencarian lokal juga dapat menyelesaikan persoalan optimasi yang mana tujuannya adalah mencari *state* terbaik berdasarkan suatu fungsi objektif. Karena itu, kita akan menyelesaikan persoalan *diagonal magic cube* dengan menggunakan pencarian lokal. Pertimbangan lain adalah algoritma *classical search* hampir tidak mungkin menemukan solusi yang mungkin karena terdapat $125!$ states yang perlu dikunjungi.

Selanjutnya, kita akan merancang kelas-kelas dan fungsi-fungsi yang akan digunakan untuk menyelesaikan persoalan ini dalam bahasa Go. Implementasi kelas sebenarnya tidak ada dalam bahasa Go, tetapi untuk mempermudah, kita tetap akan menggunakan konsep kelas untuk menggabungkan fungsi-fungsi yang berhubungan pada suatu entitas. Berikut adalah berbagai kelas dan fungsi yang digunakan dalam implementasi algoritma pencarian lokal.

1. Kelas State

Kelas state akan menyimpan larik sepanjang 125 yang merepresentasikan state dari sebuah kubus. Kelas ini juga akan menyimpan *objective value* saat ini dan jumlah dari setiap elemen dari 109 *slices* yang ada untuk mempercepat proses kalkulasi. Selain itu, kelas state juga akan menyimpan beberapa fungsi utilitas yang akan digunakan untuk keperluan pencarian lokal nantinya. Kelas state ini akan digunakan untuk algoritma pencarian lokal seperti *steepest ascent hill climbing*, *hill climbing with sideways move*, *hill-climbing with random restart*, *stochastic hill climbing*, *simulated annealing*, dan *genetic algorithm*.

A. Definisi Tipe

Seperti yang sudah dibahas sebelumnya, sebuah **State** terdiri atas *array of integer* sepanjang 125 yang menyimpan susunan angka pada kubus, jumlah dari setiap elemen dari 109 *slices*, dan *objective value* saat ini.

```
type State struct {
    Cubes    []int
    CurrSum []int
    Value    int
}
```

B. Fungsi InitVal

Fungsi ini adalah fungsi yang menghitung jumlah dari setiap elemen dari 109 *slices* yang ada dan nilai dari *objective value* berdasarkan susunan angka pada kubus.

```

func (state *State) InitVal() {
    state.CurrSum = make([]int, 109)
    // row
    idx := 0
    for i := 0; i < 5; i++ {
        for j := 0; j < 5; j++ {
            sum := 0
            for k := 0; k < 5; k++ {
                absIdx := i*5*5 + j*5 + k
                sum += state.Cubes[absIdx]
            }
            state.CurrSum[idx] = sum
            idx++
        }
    }

    // column
    for i := 0; i < 5; i++ {
        for k := 0; k < 5; k++ {
            sum := 0
            for j := 0; j < 5; j++ {
                absIdx := i*5*5 + j*5 + k
                sum += state.Cubes[absIdx]
            }
            state.CurrSum[idx] = sum
            idx++
        }
    }

    // tiang
    for j := 0; j < 5; j++ {
        for k := 0; k < 5; k++ {
            sum := 0
            for i := 0; i < 5; i++ {
                absIdx := i*5*5 + j*5 + k
                sum += state.Cubes[absIdx]
            }
            state.CurrSum[idx] = sum
            idx++
        }
    }

    // calculate the first diagonal plane
    for i := 0; i < 5; i++ {
        sum1 := 0
        sum2 := 0
        for j := 0; j < 5; j++ {
            absIdx1 := i*5*5 + j*5 + j
            absIdx2 := i*5*5 + j*5 + (4 - j)
            sum1 += state.Cubes[absIdx1]
            sum2 += state.Cubes[absIdx2]
        }
        state.CurrSum[idx] = sum1
        idx++
        state.CurrSum[idx] = sum2
    }
}

```

```

        idx++
    }

    // calculate the second diagonal plane
    for j := 0; j < 5; j++ {
        sum1 := 0
        sum2 := 0
        for i := 0; i < 5; i++ {
            absIdx1 := i*5*5 + j*5 + i
            absIdx2 := i*5*5 + j*5 + (4 - i)
            sum1 += state.Cubes[absIdx1]
            sum2 += state.Cubes[absIdx2]
        }
        state.CurrSum[idx] = sum1
        idx++
        state.CurrSum[idx] = sum2
        idx++
    }

    // calculate the third diagonal plane
    for k := 0; k < 5; k++ {
        sum1 := 0
        sum2 := 0
        for j := 0; j < 5; j++ {
            absIdx1 := j*5*5 + j*5 + k
            absIdx2 := (4-j)*5*5 + j*5 + k
            sum1 += state.Cubes[absIdx1]
            sum2 += state.Cubes[absIdx2]
        }
        state.CurrSum[idx] = sum1
        idx++
        state.CurrSum[idx] = sum2
        idx++
    }

    // space diagonal
    sum1 := 0
    sum2 := 0
    sum3 := 0
    sum4 := 0
    for i := 0; i < 5; i++ {
        absIdx1 := i*5*5 + i*5 + i
        absIdx2 := i*5*5 + (4-i)*5 + i
        absIdx3 := (4-i)*5*5 + i*5 + i
        absIdx4 := (4-i)*5*5 + (4-i)*5 + i
        sum1 += state.Cubes[absIdx1]
        sum2 += state.Cubes[absIdx2]
        sum3 += state.Cubes[absIdx3]
        sum4 += state.Cubes[absIdx4]
    }
    state.CurrSum[idx] = sum1
    idx++
    state.CurrSum[idx] = sum2
    idx++
    state.CurrSum[idx] = sum3

```

```

    idx++
    state.CurrSum[idx] = sum4
    idx++

    res := 0
    for i := 0; i < 109; i++ {
        res += ((315 - state.CurrSum[i]) * (315 - state.CurrSum[i]))
    }
    state.Value = res
}

```

C. Fungsi InitEmptyCubes

Fungsi yang menginisialisasi susunan angka pada kubus menjadi kosong.

```

func (state *State) InitEmptyCubes() {
    state.Cubes = make([]int, 125)
}

```

D. Fungsi Init

Fungsi yang menginisialisasi sebuah **State** dengan susunan angka yang acak. Fungsi ini juga memanggil InitVal untuk melakukan kalkulasi terhadap Value dan CurrSum.

```

func (state *State) Init() {
    state.Cubes = []int{}
    for i := 1; i <= 125; i++ {
        state.Cubes = append(state.Cubes, i)
    }
    rand.Shuffle(len(state.Cubes), func(i, j int) { state.Cubes[i], state.Cubes[j] = state.Cubes[j], state.Cubes[i] })
    state.InitVal()
}

```

E. Fungsi getAffectedArea

Fungsi yang mengembalikan indeks dari baris, kolom, diagonal bidang, dan diagonal ruang yang terpengaruh oleh suatu elemen pada larik kubus.

```

func getAffectedArea(idx int) []int {
    res := []int{}
    i := idx / 25
    j := (idx % 25) / 5
    k := idx % 5

    // check row
    idxRow := 5*i + j
    res = append(res, idxRow)

    // check col
    idxCol := 25 + 5*i + k
    res = append(res, idxCol)

    // check tiang
}

```

```

idxTiang := 50 + 5*j + k
res = append(res, idxTiang)

// check diagonal-1
if j == k {
    res = append(res, 75+i*2)
} else {
    res = append(res, -1)
}

if k == 4-j {
    res = append(res, 75+i*2+1)
} else {
    res = append(res, -1)
}

// check diagonal-2
if i == k {
    res = append(res, 85+j*2)
} else {
    res = append(res, -1)
}

if k == 4-i {
    res = append(res, 85+j*2+1)
} else {
    res = append(res, -1)
}

// check diagonal-3
if i == j {
    res = append(res, 95+k*2)
} else {
    res = append(res, -1)
}

if i == 4-j {
    res = append(res, 95+k*2+1)
} else {
    res = append(res, -1)
}

// check space diagonal

if i == j && j == k {
    res = append(res, 105)
} else {
    res = append(res, -1)
}
if j == 4-i && i == k {
    res = append(res, 106)
} else {
    res = append(res, -1)
}
if j == k && i == 4-j {

```

```

        res = append(res, 107)
    } else {
        res = append(res, -1)
    }
    if i == j && j == 4-k {
        res = append(res, 108)
    } else {
        res = append(res, -1)
    }
    return res
}

```

F. Fungsi BestNeighbor

Fungsi yang mengembalikan *neighbor* dengan *objective value* paling tinggi. Fungsi ini bekerja dengan menukar dua buah elemen untuk setiap pasangan indeks (i, j) dengan $0 \leq i < j < 125$ dan mencari penukaran yang menghasilkan *value* paling rendah.

```

func (state *State) BestNeighbor() (*State, int, int) {
    minValue := 1000000
    first := -1
    second := -1
    arrayFirst := []int{}
    arraySecond := []int{}
    for i := 0; i < 125; i++ {
        for j := i + 1; j < 125; j++ {
            currVal := int(state.Value)
            affectedArea1 := getAffectedArea(i)
            affectedArea2 := getAffectedArea(j)

            for k := 0; k < 13; k++ {
                area1 := affectedArea1[k]
                area2 := affectedArea2[k]
                if area1 != area2 {
                    if area1 != -1 {
                        currVal -= (315 -
state.CurrSum[area1]) * (315 - state.CurrSum[area1])
                        currVal += (315 -
(state.CurrSum[area1] + state.Cubes[j] - state.Cubes[i])) * (315 -
(state.CurrSum[area1] + state.Cubes[j] - state.Cubes[i]))
                    }
                    if area2 != -1 {
                        currVal -= (315 -
state.CurrSum[area2]) * (315 - state.CurrSum[area2])
                        currVal += (315 -
(state.CurrSum[area2] + state.Cubes[i] - state.Cubes[j])) * (315 -
(state.CurrSum[area2] + state.Cubes[i] - state.Cubes[j]))
                    }
                }
            }

            if currVal < minValue {
                minValue = currVal
                arrayFirst = []int{}

```

```

        arraySecond = []int{}
        arrayFirst = append(arrayFirst, i)
        arraySecond = append(arraySecond, j)
    } else if currVal == minVal {
        arrayFirst = append(arrayFirst, i)
        arraySecond = append(arraySecond, j)
    }
}

rndFirst := rand.Intn(len(arrayFirst))
first = arrayFirst[rndFirst]
second = arraySecond[rndFirst]

newState := State{Value: minValue}
newState.Cubes = make([]int, len(state.Cubes))
copy(newState.Cubes, state.Cubes)
newState.CurrSum = make([]int, len(state.CurrSum))
copy(newState.CurrSum, state.CurrSum)

affectedArea1 := getAffectedArea(first)
affectedArea2 := getAffectedArea(second)

for k := 0; k < 13; k++ {
    area1 := affectedArea1[k]
    area2 := affectedArea2[k]
    if area1 != area2 {
        if area1 != -1 {
            newState.CurrSum[area1] = newState.CurrSum[area1]
+ newState.Cubes[second] - newState.Cubes[first]
        }
        if area2 != -1 {
            newState.CurrSum[area2] = newState.CurrSum[area2]
+ newState.Cubes[first] - newState.Cubes[second]
        }
    }
}

temp := newState.Cubes[first]
newState.Cubes[first] = newState.Cubes[second]
newState.Cubes[second] = temp
return &newState, first, second
}

```

G. Fungs RandomNeighbor

Fungsi yang mengembalikan *neighbor* acak dari *state* saat ini. Algoritma ini bekerja dengan cara memilih pasangan (i, j) secara acak dan menukar elemen pada pasangan indeks tersebut.

```

func (state *State) RandomNeighbor() (*State, int, int) {
    first := rand.Intn(125)
    second := rand.Intn(125)

    for first == second {
        second = rand.Intn(125)
    }
}

```

```

    }

    newState := State{}
    newState.Cubes = make([]int, len(state.Cubes))
    copy(newState.Cubes, state.Cubes)
    newState.CurrSum = make([]int, len(state.CurrSum))
    copy(newState.CurrSum, state.CurrSum)

    currVal := state.Value
    affectedArea1 := getAffectedArea(first)
    affectedArea2 := getAffectedArea(second)

    for k := 0; k < 13; k++ {
        area1 := affectedArea1[k]
        area2 := affectedArea2[k]
        if area1 != area2 {
            if area1 != -1 {
                currVal -= (315 - state.CurrSum[area1]) * (315 -
state.CurrSum[area1])
                currVal += (315 - (state.CurrSum[area1] +
state.Cubes[second] - state.Cubes[first])) * (315 - (state.CurrSum[area1] +
state.Cubes[second] - state.Cubes[first]))
                newState.CurrSum[area1] = newState.CurrSum[area1] +
newState.Cubes[second] - newState.Cubes[first]
            }
            if area2 != -1 {
                currVal -= (315 - state.CurrSum[area2]) * (315 -
state.CurrSum[area2])
                currVal += (315 - (state.CurrSum[area2] +
state.Cubes[first] - state.Cubes[second])) * (315 - (state.CurrSum[area2] +
state.Cubes[first] - state.Cubes[second]))
                newState.CurrSum[area2] = newState.CurrSum[area2] +
newState.Cubes[first] - newState.Cubes[second]
            }
        }
    }

    temp := newState.Cubes[first]
    newState.Cubes[first] = newState.Cubes[second]
    newState.Cubes[second] = temp

    newState.Value = currVal
    return &newState, first, second
}

```

A. Steepest Ascent Hill-Climbing

Algoritma Steepest Ascent Hill-Climbing melakukan iterasi terus menerus mencari *neighbor* dengan fungsi objektif dengan nilai yang paling rendah, jika hasil fungsi objektif dari state *neighbor* sama atau lebih dari fungsi objektif state *current* maka akan mengembalikan state *current*. Jika nilainya kurang dari fungsi objektif state *current* maka perbarui variabel *current* dengan state *neighbor*.

```

func HillClimbing() (*models.State, *models.State, []models.Iteration, int) {
    initial := &models.State{}
    initial.Init()

    current := initial
    iterations := []models.Iteration{}
    iter := 0

    for {
        iter++
        neighbor, first, second := current.BestNeighbor()

        if neighbor.Value >= current.Value {
            return initial, current, iterations, iter
        }

        current = neighbor
        iterations = append(iterations, models.Iteration{
            First: first,
            Second: second,
            Value: current.Value,
            Exp: 1,
        })
    }
}

```

Kelebihan	Kekurangan
Karena selalu memilih neighbor dengan nilai objektif yang lebih baik maka akan menyelesaikan problem lebih cepat (walaupun tidak global optimum)	Sering sekali terjebak di local optima karena tidak punya alternatif lain dalam memilih neighbor selain memilih nilai objektif yang lebih tinggi

B. Hill-climbing with Sideways Move

Algoritma Hill-Climbing with Sideways Move melakukan iterasi terus menerus mencari *neighbor* dengan fungsi objektif dengan nilai yang paling rendah, jika hasil fungsi objektif dari state *neighbor* lebih dari fungsi objektif state *current* maka akan mengembalikan state *current*. Jika nilainya kurang dari atau sama dengan fungsi objektif state *current* maka perbarui variabel *current* dengan state *neighbor*. Algoritma ini memiliki kemungkinan untuk terus menerus sideway tanpa batas sehingga membuat program terjebak dalam *infinity loop*. Untuk menghindari hal ini, diperlukan pembatasan seberapa banyak langkah sideway yang terjadi.

```

func HillClimbingSideways(maxMove int) (*models.State, *models.State, []models.Iteration, int) {
    initial := &models.State{}
    initial.Init()

    current := initial
    iterations := []models.Iteration{}

```

```

iter := 0
counterMove := 0

for {
    iter++
    neighbor, first, second := current.BestNeighbor()

    if neighbor.Value > current.Value {
        return initial, current, iterations, iter
    } else if neighbor.Value == current.Value {
        if counterMove < maxMove {
            counterMove++
            current = neighbor
            iterations = append(iterations, models.Iteration{
                First: first,
                Second: second,
                Value: current.Value,
                Exp: 1,
            })
        } else {
            return initial, current, iterations, iter
        }
    } else {
        counterMove = 0
        current = neighbor
        iterations = append(iterations, models.Iteration{
            First: first,
            Second: second,
            Value: current.Value,
            Exp: 1,
        })
    }
}
}

```

Kelebihan	Kekurangan
Opsi memilih neighbor dengan nilai objektif yang sama membuat kemungkinan mencapai global optimum lebih tinggi daripada steepest ascent	Tanpa mekanisme untuk membatasi langkah sideways, program akan terus menerus bergerak di local optima tanpa adanya kemajuan
	Masih terdapat kemungkinan untuk stuck di local optima karena tidak ada opsi lain untuk memilih <i>neighbor</i> selain nilai objektif lebih tinggi atau sama dari <i>current</i>

C. Hill-climbing with Random Restart

Algoritma hill-climbing with random restart bekerja dengan melakukan serangkaian percobaan pencarian hill-climbing dengan initial state yang random hingga solusi tujuan ditemukan. Untuk setiap

hill climbing, cara kerjanya sama dengan *steepest ascent hill climbing*, yaitu untuk setiap state akan dibangkitkan tetangga yang terbaik dan berpindah ke tetangga tersebut jika memiliki nilai yang lebih optimal dibandingkan nilai state saat ini. Berikut adalah implementasi dari algoritma ini.

```
func RandomRestartHillClimbing(maxRestart int) (*models.State, *models.State, []models.RestartIteration, int) {
    restartIteration := []models.RestartIteration{}
    counter := 0

    best := &models.State{}
    best.Init()
    best.Value = 10000000

    for counter < maxRestart {
        counter++
        init, final, iteration, iter := HillClimbing()
        restartIteration = append(restartIteration, models.RestartIteration{
            Initial: *init,
            Final:   *final,
            Iter:     iteration,
            NumIter: iter,
        })
        if final.Value < best.Value {
            best = final
        }
        if best.Value == 0 {
            break
        }
    }

    return &restartIteration[0].Initial, best, restartIteration, counter
}
```

Kelebihan	Kekurangan
Dengan berulang kali melakukan pencarian, pada akhirnya kita akan menemukan solusi <i>global optima</i> jika <i>initial state</i> memiliki nilai yang baik.	Tidak efisien untuk permasalahan yang <i>simple</i> karena <i>random restart</i> mungkin akan tidak diperlukan dan tidak efisien.
Karena berulang kali melakukan pencarian hill climbing, algoritma ini terhindar dari lokal maksimum karena dapat mengeksplor area lain melalui restart yang dilakukan pada initial state yang acak.	Setiap <i>restart</i> akan memulai pencarian yang baru dan mungkin saja nilai objektifnya menjadi sangat jauh dari nilai optimal sehingga akan memerlukan waktu yang lama untuk menyelesaikan persoalan ketimbang sideway dan <i>steepest ascent</i>

D. Stochastic Hill-Climbing

Algoritma selanjutnya yang akan dibahas adalah *stochastic hill-climbing*. Algoritma ini bekerja dengan memilih tetangga acak dari *state* saat ini. Jika nilai fungsi objektif dari tetangga tersebut lebih kecil dibandingkan nilai dari *state* saat ini, kita akan berpindah ke state tersebut. Jika tidak, kita akan melanjutkan iterasi. Iterasi akan diterminasi jika sudah mencapai NMAX kali. Algoritma ini akan *converge* lebih lambat dibandingkan algoritma *steepest ascent*, tetapi pada beberapa kasus, algoritma ini dapat menemukan solusi yang baik karena tidak akan terjebak dalam *local optima*. Berikut adalah implementasi dari algoritma ini.

```
func StochasticHillClimbing() (*models.State, *models.State, []models.Iteration, int) {
    initial := &models.State{}
    initial.Init()

    current := initial
    iterations := []models.Iteration{}
    counter := 0

    for counter < 800000 {
        counter++
        neighbor, first, second := current.RandomNeighbor()

        if neighbor.Value < current.Value {
            current = neighbor
            iterations = append(iterations, models.Iteration{
                First: first,
                Second: second,
                Value: current.Value,
                Exp: 1,
            })
        }
        if current.Value == 0 {
            return initial, current, iterations, counter
        }
    }
    return initial, current, iterations, counter
}
```

Kelebihan	Kekurangan
Tidak seperti <i>hill-climbing</i> lainnya yang deterministik, <i>stochastic hill climbing</i> mengambil tetangga secara acak. Hal ini membantu menghindari jebakan <i>local optima</i> dengan memberi kesempatan untuk mengeksplorasi solusi	Meskipun <i>stochastic hill climbing</i> dapat menghindari <i>local optima</i> dalam beberapa kasus, tidak ada jaminan bahwa algoritma ini akan mencapai solusi optimal secara global.
Karena adanya elemen acak, algoritma ini dapat menjelajahi lebih banyak solusi yang berpotensi	Karena ada elemen acak, algoritma ini mungkin saja mengeksplorasi terlalu banyak solusi yang

lebih baik daripada algoritma *hill climbing* lain yang deterministik.

kurang relevan. Akibatnya, algoritma ini mengalami *converge* yang lebih lambat dibandingkan algoritma *steepest ascent* dan meningkatkan inefisiensi.

E. Simulated Annealing

Algoritma *hill-climbing* yang selalu berpindah ke *state* dengan nilai fungsi objektif lebih tinggi sangat rawan untuk terjebak pada *local optima*. Di sisi lain, algoritma yang bergerak hanya berdasarkan tetangga yang dibangkitkan secara acak tanpa memperhatikan nilai fungsi objektif pada akhirnya akan mendapatkan nilai *global optima*, tetapi akan sangat tidak efisien. Salah satu cara untuk menggabungkan *hill climbing* dengan gerakan acak untuk menghasilkan solusi yang efisien dan *complete* adalah dengan *simulated annealing*.

Simulated annealing terinspirasi dari proses *annealing* di bidang metalurgi, yaitu proses yang digunakan untuk mengeraskan logam dan kaca dengan cara memanaskannya dengan temperatur tinggi dan kemudian mendinginkannya secara perlahan. Ide utama dari *simulated annealing* adalah membiarkan beberapa gerakan yang buruk, tetapi secara perlahan mengurangi frekuensinya. Secara umum, algoritma *simulated annealing* bekerja sebagai berikut untuk persoalan minimasi.

1. Pada awalnya, kita diberikan suatu *state* acak.
2. Pada setiap iterasi, tetangga dari *state* saat ini akan dibangkitkan secara acak.
3. Jika nilai fungsi objektif dari tetangga lebih kecil dari *state* saat ini, *state* akan berpindah ke tetangga.
4. Jika nilai fungsi objektif dari tetangga tidak lebih kecil dari *state* saat ini, algoritma ini memungkinkan *state* untuk berpindah ke tetangga dengan probabilitas tertentu. Probabilitas ini dihitung menggunakan fungsi eksponensial yang bergantung pada perbedaan nilai objektif antara tetangga dan *state*, dan suhu saat ini:

$$P = \exp\left(\frac{\Delta E}{T}\right)$$

dimana:

- $\Delta E = \text{OBJECTIVE}(\text{CURR}) - \text{OBJECTIVE}(\text{NEIGHBOR})$
 - T adalah suhu pada iterasi tersebut yang akan berangsurg-angsurg berkurang
5. Suhu diturunkan secara bertahap menurut suatu skema pendinginan sehingga awalnya banyak *state* “buruk” yang diterima, tetapi seiring waktu frekuensinya berkurang. Adapun penjadwalan suhu yang digunakan adalah penjadwalan eksponensial.

$$T = T_0 \times \alpha^t$$

dengan $T_0 = 9000000$ dan $\alpha = 0.9995$.

6. Algoritma ini berhenti ketika suhu sudah mencapai titik 0 ($T = 0$) atau solusi global sudah ditemukan.

Berdasarkan deskripsi tersebut, kita dapat merancang solusi *simulated annealing* untuk menemukan solusi *diagonal magic cube*. Berikut adalah implementasi *simulated annealing* untuk menyelesaikan *diagonal magic cube*.

```
func SimulatedAnnealing() (*models.State, *models.State, []models.Iteration, int, int) {
    initial := &models.State{}
    initial.Init()

    current := initial
    iterations := []models.Iteration{}
    iter := 0
    stuck := 0

    for t := 1; ; t++ {
        iter++
        T := schedule(t)

        if T <= 0.01 || current.Value == 0 {
            return initial, current, iterations, iter, stuck
        }

        neighbor, first, second := current.RandomNeighbor()

        deltaE := current.Value - neighbor.Value

        if deltaE > 0 {
            current = neighbor
            iterations = append(iterations, models.Iteration{
                First: first,
                Second: second,
                Value: current.Value,
                Exp: 1,
            })
        } else {
            probability := math.Exp(float64(deltaE) / T)
            if rand.Float64() <= probability {
                current = neighbor
                stuck++
                iterations = append(iterations, models.Iteration{
                    First: first,
                    Second: second,
                    Value: current.Value,
                    Exp: probability,
                })
            }
        }
    }

    func schedule(t int) float64 {
        initialTemperature := 9000000.0
        coolingRate := 0.99995
    }
}
```

```

T := initialTemperature * math.Pow(coolingRate, float64(t))

return T
}

```

F. Genetic Algorithm

Genetic Algorithm (GA) adalah metode untuk memecahkan masalah optimasi, baik *constrained* maupun *unconstrained* berdasarkan proses seleksi dalam dalam biologi evolusi. Secara umum, algoritma ini memodifikasi populasi dari *state* berulang kali. Pada setiap langkah, *genetic algorithm* memilih individu-individu dari populasi saat ini dan menggunakan sebagai induk untuk memproduksi anak-anak untuk generasi berikutnya. Dari generasi ke generasi, populasi “berevolusi” untuk menemukan solusi yang optimal. Untuk lebih mengerti tentang algoritma ini, berikut adalah beberapa terminologi penting dalam *genetic algorithm*.

1. **Populasi:** GA bekerja dengan sekelompok solusi potensial yang disebut populasi. Setiap populasi disebut individu (*state*). Pada awalnya, populasi ini diinisialisasi secara acak.
2. **Fitness Function:** Setiap individu (*state*), dievaluasi menggunakan fungsi yang disebut fitness function, yang mengukur seberapa baik individu tersebut dalam menyelesaikan masalah atau mendekati solusi optimal.
3. **Selection:** Individu yang memiliki nilai fitness yang lebih tinggi memiliki peluang lebih besar untuk dipilih untuk menghasilkan keturunan (*offspring*). Teknik seleksi yang umum dilakukan antara lain: *roulette wheel selection*, *tournament rank*, dan *rank-based selection*.
4. **Crossover:** Dua individu dipilih dari populasi dan digabungkan untuk menghasilkan keturunan baru. Crossover adalah proses yang mirip dengan reproduksi seksual dalam biologi, di mana sebagian dari materi genetik orang tua dipertukarkan.
5. **Mutation:** Setelah crossover, beberapa individu mengalami mutasi, yaitu perubahan acak pada kromosom. Mutasi membuat keragaman dalam populasi dan mencegah GA terjebak dalam solusi lokal.

Pada persoalan *diagonal magic cube*, kita menggunakan fungsi objektif yang ingin diminimasi, sedangkan *fitness function* dalam konteks GA dirancang untuk memaksimalkan nilainya. Untuk itu, kita perlu merancang *fitness function* untuk mengkonversi persoalan minimasi menjadi persoalan maksimasi. Pendekatan yang dipilih adalah *inverse objective function* sehingga *fitness function* didefinisikan sebagai berikut.

$$fitness(state) = \frac{1}{1+OBJECTIVE(state)}$$

Fungsi tersebut memastikan bahwa semakin kecil nilai *OBJECTIVE(state)*, semakin besar pula nilai *fitness*-nya.

Untuk algoritma *selection*, kita akan memilih *roulette wheel selection* dimana probabilitas tiap individu untuk dipilih didefinisikan sebagai nilai *fitness* suatu individu dibagi dengan nilai *fitness* total untuk setiap individu dalam populasi. Persentase *roulette wheel* akan disusun dimulai dari individu dengan persentase paling tinggi. Dengan demikian, individu dengan nilai *fitness* lebih tinggi memiliki probabilitas yang lebih tinggi pula untuk dipilih.

Proses *crossover* membutuhkan sebuah algoritma yang dapat merekombinasi dua buah individu menjadi individu baru yang tidak kehilangan sifat permutasinya (memiliki angka unik dari 1 - 125). Untuk itu, kami memutuskan untuk menggunakan algoritma *order crossover*. Berikut adalah cara kerja algoritma ini secara singkat.

1. Misalkan terdapat dua buah individu dalam populasi yang ingin direkombinasi, yaitu P_0 dan P_1 .

$$P_0 = (A, B, C, D, E, F, G, H, I, J)$$

$$P_1 = (B, D, A, H, J, C, E, G, F, I)$$

2. Pilih beberapa segment acak pada P_0 .

$$P_0 = (\underline{A}, \underline{B}, C, D, E, \underline{F}, \underline{G}, \underline{H}, I, J)$$

3. Copy segment yang terpilih dari P_0 ke individu anak (hasil).

$$P_C = (A, B, ?, ?, ?, F, G, H, ?, ?)$$

4. Sisa elemen yang belum ada pada anak, ditransfer dari P_1 sesuai dengan urutan kemunculannya pada individu P_1 .

$$P_{\text{missing}} = \{C, D, E, I, J\}$$

$$P_{\text{in order from } P_1} = (D, J, C, E, I)$$

5. Hasil dari rekombinasi tersebut akan menghasilkan individu baru.

$$P_C = (A, B, \underline{D}, \underline{J}, C, F, G, H, \underline{E}, I)$$

Tahap terakhir yang perlu diperhatikan adalah mutasi. Sama halnya seperti *crossover*, mutasi juga memerlukan sebuah algoritma yang dapat memastikan bahwa individu yang dimutasi tetap mempertahankan sifat permutasi yang dimilikinya. Algoritma yang kita pilih adalah *rotation to the right*. Berikut adalah cara kerja algoritma ini secara singkat.

1. Misalkan kita memiliki sebuah individu yang berada dalam bentuk permutasi, misalkan P_0 .
2. Pilih sebuah *list* parsial dengan menentukan sebuah index awal i dan indeks akhir j pada P_0 , yang memenuhi $0 < i, j < |P_0| - 1$. Perhatikan bahwa, i dan j dapat bersifat sirkular.
3. Tentukan bilangan k yang menyatakan banyak rotasi dilakukan pada *list* parsial tersebut.
4. Lakukan rotasi pada *sublist* tersebut sebanyak k kali ke kanan.

Selanjutnya, kita akan mendesain implementasi dari GA untuk menyelesaikan persoalan *diagonal magic cube* berdasarkan algoritma-algoritma yang sudah dibahas sebelumnya. Berikut adalah rancangan kelas dan implementasi dari algoritma GA dalam pseudo code

```
func GeneticAlgorithm(maxIteration int, initPopulation int) ([]*models.State,
[]*models.State, []models.GeneticIteration, *models.State, int) {
    initialPops := []*models.State{}
```

```

        for i := 0; i < initPopulation; i++ {
            newState := models.State{}
            newState.Init()
            initialPops = append(initialPops, &newState)
        }

        iterations := []models.GeneticIteration{}

        valSum := 0
        minVal := 10000000
        minIdx := 0
        population := initialPops
        for iter := 1; iter <= maxIteration; iter++ {
            newPopulation := []*models.State{}
            valSum = 0
            minVal = 10000000
            minIdx = 0
            for i := 0; i < len(population); i++ {
                x := randomSelection(population)
                y := randomSelection(population)
                child := reproduce(x, y)
                if rand.Float64() < 0.1 {
                    mutation(child)
                }
                valSum += child.Value
                if minVal > child.Value {
                    minVal = child.Value
                    minIdx = i
                }
                newPopulation = append(newPopulation, child)
            }
            population = newPopulation
            avgVal := float64(valSum) / float64(len(population))
            iterations = append(iterations, models.GeneticIteration{MinValue:
minVal, AvgValue: avgVal})

            if minVal == 0 {
                return initialPops, population, iterations,
population[minIdx], iter
            }
        }
        return initialPops, population, iterations, population[minIdx],
maxIteration
    }

    func fitness(state *models.State) float64 {
        return 1.0 / (1.0 + float64(state.Value))
    }

    func randomSelection(population []*models.State) *models.State {
        prefSum := []float64{0}
        sum := 0.0
        for i := 0; i < len(population); i++ {
            fit := fitness(population[i]) * 1000
            sum += fit

```

```

        prev := prefSum[i]
        prefSum = append(prefSum, prev+fit)
    }

    selectedVal := rand.Float64() * sum

    lo := 0
    hi := len(prefSum)
    ans := -1
    for lo <= hi {
        mid := (lo + hi) / 2
        if prefSum[mid] <= selectedVal {
            ans = mid
            lo = mid + 1
        } else {
            hi = mid - 1
        }
    }

    return population[ans]
}

func reproduce(p1 *models.State, p2 *models.State) *models.State {
    child := models.State{}
    vis := make([]bool, 125)
    child.InitEmptyCubes()
    start, end := randomSegment()
    rangeLen := (end - start + 1 + 125) % 125
    for i := 0; i < rangeLen; i++ {
        child.Cubes[(start+i)%125] = p1.Cubes[(start+i)%125]
        vis[child.Cubes[(start+i)%125]-1] = true
    }

    i := 0
    j := 0

    for i < 125 && j < 125 {
        if child.Cubes[i] != 0 {
            i++
            continue
        }
        if vis[p2.Cubes[j]-1] {
            j++
            continue
        }
        child.Cubes[i] = p2.Cubes[j]
        i++
        j++
    }
    child.InitVal()
    return &child
}

func randomSegment() (int, int) {
    start := rand.Intn(125)

```

```
    end := rand.Intn(125)
    for start == end {
        end = rand.Intn(125)
    }
    return start, end
}

func mutation(state *models.State) {
    first := rand.Intn(125)
    second := rand.Intn(125)
    segLen := (second - first + 1 + 125) % 125

    rot := 0
    if segLen != 0 {
        rot = rand.Intn(segLen)
    }

    temp := make([]int, segLen)
    for i := 0; i < segLen; i++ {
        srcIdx := (first + i) % 125
        targetIdx := (i + rot) % segLen
        temp[targetIdx] = state.Cubes[srcIdx]
    }

    for i := 0; i < segLen; i++ {
        state.Cubes[(first+i)%125] = temp[i]
    }
}
```

BAB IV

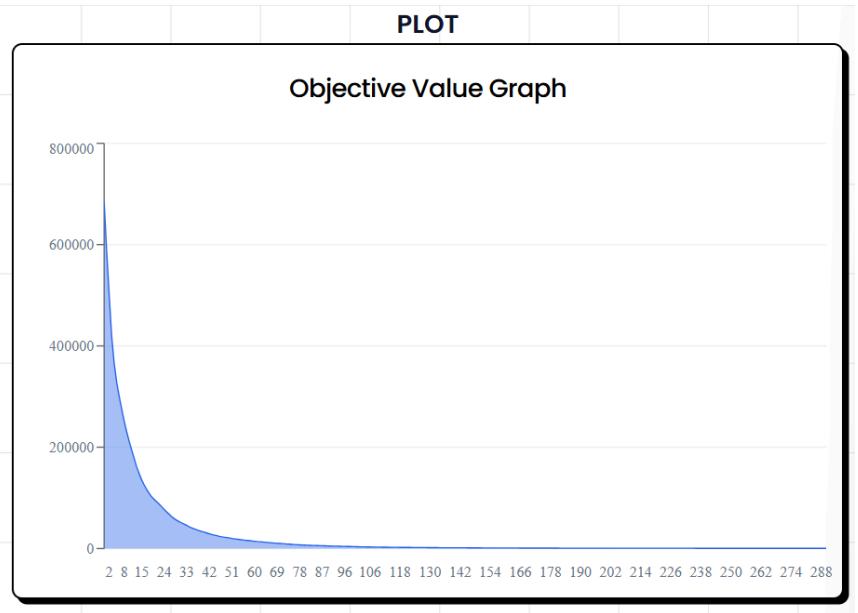
HASIL EKSPERIMENT

A. Steepest Ascent Hill-Climbing

Percobaan 1

Objective Value dan Durasi	<p style="text-align: center;">Result</p> <p>Best Objective : 261</p> <p>Total Iteration : 289</p> <p>Duration Search : 1158.205 ms</p>
State Awal	<p style="text-align: center;">INITIAL</p> <p>Objective Value: 685996</p> <p>The initial state consists of four 2x4 grids of numbers:</p> <ul style="list-style-type: none">Grid 1: 48, 51, 89, 75, 53; 125, 91, 65, 59, 80; 109, 2, 19, 35, 93; 16, 78, 55, 111, 72; 115, 22, 113, 69, 7.Grid 2: 103, 96, 106, 122, 14; 68, 12, 121, 27, 114; 61, 15, 13, 52, 29; 74, 104, 124, 42, 31; 9, 116, 56, 36, 92.Grid 3: 73, 87, 79, 21, 40; 66, 107, 99, 46, 77; 28, 119, 20, 58, 24; 41, 26, 63, 57, 110; 94, 17, 86, 102, 33.Grid 4: 101, 76, 98, 120, 84; 64, 108, 123, 50, 25; 6, 118, 45, 34, 18; 62, 1, 95, 82, 44; 5, 4, 54, 32, 30.
State Akhir	<p style="text-align: center;">FINAL</p> <p>Objective Value: 261</p> <p>The final state consists of four 2x4 grids of numbers:</p> <ul style="list-style-type: none">Grid 1: 42, 47, 110, 85, 33; 70, 99, 14, 56, 77; 125, 13, 64, 28, 82; 37, 123, 7, 68, 80; 40, 34, 120, 78, 43.Grid 2: 11, 97, 3, 119, 84; 31, 46, 107, 35, 96; 121, 9, 103, 81, 2; 100, 41, 92, 53, 29; 51, 122, 12, 24, 104.Grid 3: 71, 106, 74, 39, 25; 66, 75, 54, 114, 6; 23, 109, 62, 45, 76; 60, 20, 98, 17, 118; 94, 4, 27, 102, 90.Grid 4: 101, 1, 113, 15, 86; 61, 63, 22, 89, 79; 26, 69, 49, 67, 105; 108, 72, 5, 93, 38; 19, 112, 124, 52, 8.

Plot Objective Value terhadap Iterasi



Percobaan 2

Objective Value dan Durasi

Result

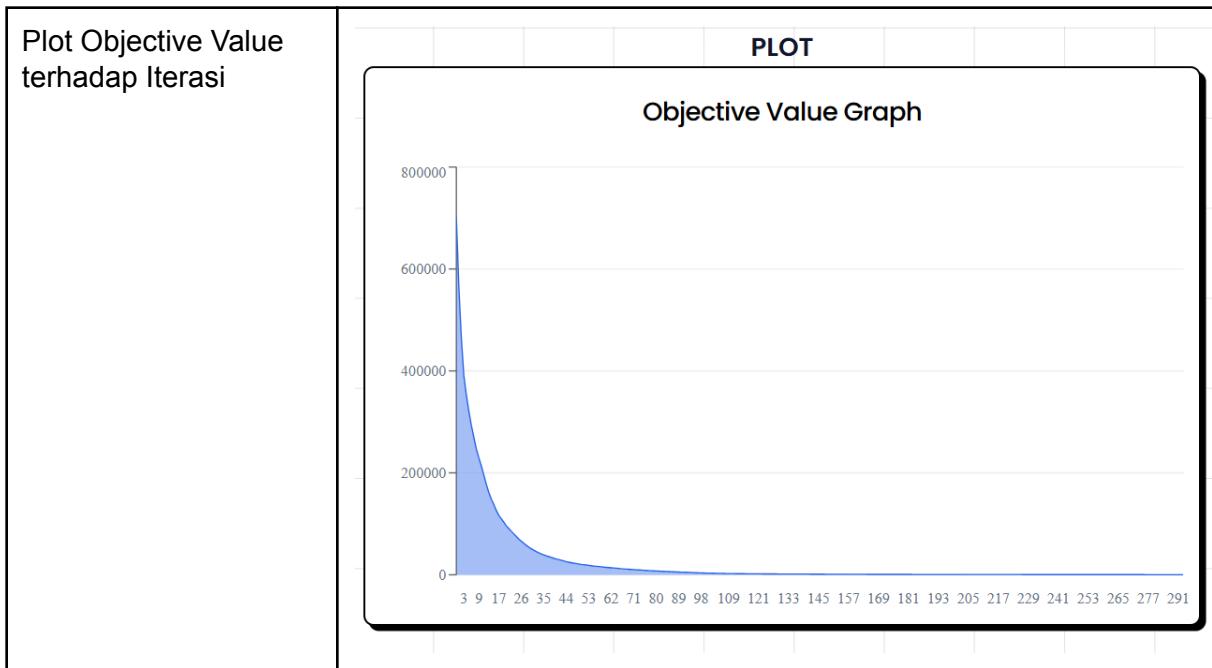
Best Objective : 188
 Total Iteration : 292
 Duration Search : 1163.411 ms

State Awal

INITIAL																																																																																																							
Objective Value: 702637																																																																																																							
<table border="1"> <tr><td>44</td><td>106</td><td>59</td><td>115</td><td>5</td></tr> <tr><td>49</td><td>67</td><td>60</td><td>63</td><td>17</td></tr> <tr><td>36</td><td>62</td><td>73</td><td>110</td><td>16</td></tr> <tr><td>114</td><td>117</td><td>101</td><td>111</td><td>82</td></tr> <tr><td>88</td><td>40</td><td>42</td><td>13</td><td>91</td></tr> </table>	44	106	59	115	5	49	67	60	63	17	36	62	73	110	16	114	117	101	111	82	88	40	42	13	91	<table border="1"> <tr><td>66</td><td>121</td><td>30</td><td>103</td><td>102</td></tr> <tr><td>7</td><td>52</td><td>93</td><td>65</td><td>105</td></tr> <tr><td>69</td><td>72</td><td>85</td><td>61</td><td>43</td></tr> <tr><td>120</td><td>90</td><td>96</td><td>26</td><td>20</td></tr> <tr><td>51</td><td>12</td><td>4</td><td>112</td><td>55</td></tr> </table>	66	121	30	103	102	7	52	93	65	105	69	72	85	61	43	120	90	96	26	20	51	12	4	112	55	<table border="1"> <tr><td>8</td><td>56</td><td>89</td><td>70</td><td>23</td></tr> <tr><td>11</td><td>99</td><td>21</td><td>54</td><td>58</td></tr> <tr><td>31</td><td>124</td><td>108</td><td>113</td><td>92</td></tr> <tr><td>123</td><td>53</td><td>84</td><td>1</td><td>37</td></tr> <tr><td>27</td><td>86</td><td>119</td><td>48</td><td>33</td></tr> </table>	8	56	89	70	23	11	99	21	54	58	31	124	108	113	92	123	53	84	1	37	27	86	119	48	33	<table border="1"> <tr><td>18</td><td>9</td><td>29</td><td>22</td><td>97</td></tr> <tr><td>83</td><td>47</td><td>118</td><td>15</td><td>95</td></tr> <tr><td>57</td><td>87</td><td>19</td><td>122</td><td>80</td></tr> <tr><td>81</td><td>41</td><td>6</td><td>35</td><td>98</td></tr> <tr><td>68</td><td>2</td><td>3</td><td>10</td><td>45</td></tr> </table>	18	9	29	22	97	83	47	118	15	95	57	87	19	122	80	81	41	6	35	98	68	2	3	10	45
44	106	59	115	5																																																																																																			
49	67	60	63	17																																																																																																			
36	62	73	110	16																																																																																																			
114	117	101	111	82																																																																																																			
88	40	42	13	91																																																																																																			
66	121	30	103	102																																																																																																			
7	52	93	65	105																																																																																																			
69	72	85	61	43																																																																																																			
120	90	96	26	20																																																																																																			
51	12	4	112	55																																																																																																			
8	56	89	70	23																																																																																																			
11	99	21	54	58																																																																																																			
31	124	108	113	92																																																																																																			
123	53	84	1	37																																																																																																			
27	86	119	48	33																																																																																																			
18	9	29	22	97																																																																																																			
83	47	118	15	95																																																																																																			
57	87	19	122	80																																																																																																			
81	41	6	35	98																																																																																																			
68	2	3	10	45																																																																																																			
<table border="1"> <tr><td>38</td><td>125</td><td>50</td><td>79</td><td>34</td></tr> <tr><td>75</td><td>14</td><td>116</td><td>94</td><td>77</td></tr> <tr><td>76</td><td>104</td><td>107</td><td>109</td><td>78</td></tr> <tr><td>39</td><td>71</td><td>24</td><td>74</td><td>46</td></tr> <tr><td>32</td><td>100</td><td>28</td><td>25</td><td>64</td></tr> </table>	38	125	50	79	34	75	14	116	94	77	76	104	107	109	78	39	71	24	74	46	32	100	28	25	64	<table border="1"> <tr><td>38</td><td>125</td><td>50</td><td>79</td><td>34</td></tr> <tr><td>75</td><td>14</td><td>116</td><td>94</td><td>77</td></tr> <tr><td>76</td><td>104</td><td>107</td><td>109</td><td>78</td></tr> <tr><td>39</td><td>71</td><td>24</td><td>74</td><td>46</td></tr> <tr><td>32</td><td>100</td><td>28</td><td>25</td><td>64</td></tr> </table>	38	125	50	79	34	75	14	116	94	77	76	104	107	109	78	39	71	24	74	46	32	100	28	25	64	<table border="1"> <tr><td>111</td><td>34</td><td>59</td><td>6</td><td>104</td></tr> <tr><td>22</td><td>89</td><td>114</td><td>53</td><td>36</td></tr> <tr><td>31</td><td>95</td><td>7</td><td>109</td><td>74</td></tr> <tr><td>90</td><td>91</td><td>11</td><td>66</td><td>56</td></tr> <tr><td>61</td><td>5</td><td>124</td><td>81</td><td>44</td></tr> </table>	111	34	59	6	104	22	89	114	53	36	31	95	7	109	74	90	91	11	66	56	61	5	124	81	44	<table border="1"> <tr><td>60</td><td>80</td><td>18</td><td>118</td><td>39</td></tr> <tr><td>67</td><td>2</td><td>115</td><td>32</td><td>99</td></tr> <tr><td>79</td><td>101</td><td>102</td><td>4</td><td>28</td></tr> <tr><td>27</td><td>58</td><td>12</td><td>112</td><td>108</td></tr> <tr><td>82</td><td>75</td><td>68</td><td>49</td><td>40</td></tr> </table>	60	80	18	118	39	67	2	115	32	99	79	101	102	4	28	27	58	12	112	108	82	75	68	49	40
38	125	50	79	34																																																																																																			
75	14	116	94	77																																																																																																			
76	104	107	109	78																																																																																																			
39	71	24	74	46																																																																																																			
32	100	28	25	64																																																																																																			
38	125	50	79	34																																																																																																			
75	14	116	94	77																																																																																																			
76	104	107	109	78																																																																																																			
39	71	24	74	46																																																																																																			
32	100	28	25	64																																																																																																			
111	34	59	6	104																																																																																																			
22	89	114	53	36																																																																																																			
31	95	7	109	74																																																																																																			
90	91	11	66	56																																																																																																			
61	5	124	81	44																																																																																																			
60	80	18	118	39																																																																																																			
67	2	115	32	99																																																																																																			
79	101	102	4	28																																																																																																			
27	58	12	112	108																																																																																																			
82	75	68	49	40																																																																																																			

State Akhir

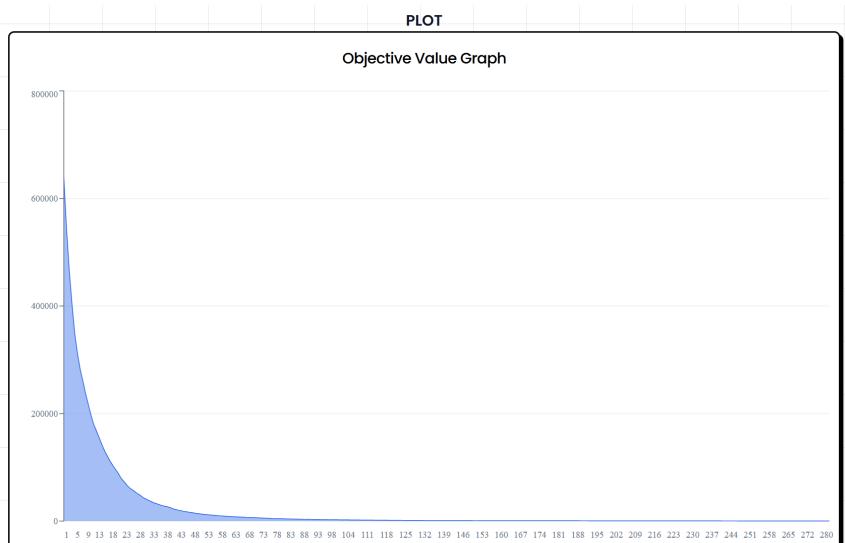
FINAL																																																																																																							
Objective Value: 188																																																																																																							
<table border="1"> <tr><td>78</td><td>84</td><td>119</td><td>26</td><td>9</td></tr> <tr><td>83</td><td>41</td><td>14</td><td>105</td><td>71</td></tr> <tr><td>46</td><td>50</td><td>48</td><td>97</td><td>73</td></tr> <tr><td>3</td><td>47</td><td>110</td><td>70</td><td>86</td></tr> <tr><td>106</td><td>92</td><td>24</td><td>19</td><td>76</td></tr> </table>	78	84	119	26	9	83	41	14	105	71	46	50	48	97	73	3	47	110	70	86	106	92	24	19	76	<table border="1"> <tr><td>17</td><td>107</td><td>30</td><td>123</td><td>37</td></tr> <tr><td>29</td><td>64</td><td>57</td><td>72</td><td>94</td></tr> <tr><td>121</td><td>69</td><td>96</td><td>8</td><td>23</td></tr> <tr><td>93</td><td>54</td><td>98</td><td>25</td><td>45</td></tr> <tr><td>55</td><td>21</td><td>33</td><td>88</td><td>116</td></tr> </table>	17	107	30	123	37	29	64	57	72	94	121	69	96	8	23	93	54	98	25	45	55	21	33	88	116	<table border="1"> <tr><td>51</td><td>10</td><td>87</td><td>42</td><td>125</td></tr> <tr><td>113</td><td>120</td><td>16</td><td>52</td><td>15</td></tr> <tr><td>35</td><td>1</td><td>62</td><td>100</td><td>117</td></tr> <tr><td>103</td><td>63</td><td>85</td><td>43</td><td>20</td></tr> <tr><td>13</td><td>122</td><td>65</td><td>77</td><td>38</td></tr> </table>	51	10	87	42	125	113	120	16	52	15	35	1	62	100	117	103	63	85	43	20	13	122	65	77	38	<table border="1"> <tr><td>111</td><td>34</td><td>59</td><td>6</td><td>104</td></tr> <tr><td>22</td><td>89</td><td>114</td><td>53</td><td>36</td></tr> <tr><td>31</td><td>95</td><td>7</td><td>109</td><td>74</td></tr> <tr><td>90</td><td>91</td><td>11</td><td>66</td><td>56</td></tr> <tr><td>61</td><td>5</td><td>124</td><td>81</td><td>44</td></tr> </table>	111	34	59	6	104	22	89	114	53	36	31	95	7	109	74	90	91	11	66	56	61	5	124	81	44
78	84	119	26	9																																																																																																			
83	41	14	105	71																																																																																																			
46	50	48	97	73																																																																																																			
3	47	110	70	86																																																																																																			
106	92	24	19	76																																																																																																			
17	107	30	123	37																																																																																																			
29	64	57	72	94																																																																																																			
121	69	96	8	23																																																																																																			
93	54	98	25	45																																																																																																			
55	21	33	88	116																																																																																																			
51	10	87	42	125																																																																																																			
113	120	16	52	15																																																																																																			
35	1	62	100	117																																																																																																			
103	63	85	43	20																																																																																																			
13	122	65	77	38																																																																																																			
111	34	59	6	104																																																																																																			
22	89	114	53	36																																																																																																			
31	95	7	109	74																																																																																																			
90	91	11	66	56																																																																																																			
61	5	124	81	44																																																																																																			



Percobaan 3

Objective Value dan Durasi	
	<p>Result</p> <p>Best Objective : 231</p> <p>Total Iteration : 281</p> <p>Duration Search : 1273.84 ms</p>

Plot Objective Value terhadap Iterasi



B. Hill-Climbing with Sideways Move

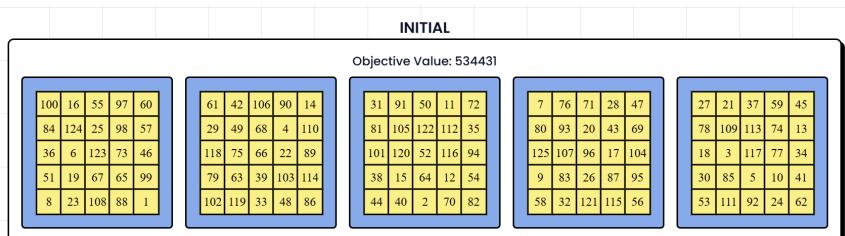
Percobaan 1: Max Iteration 10

Objective Value dan Durasi

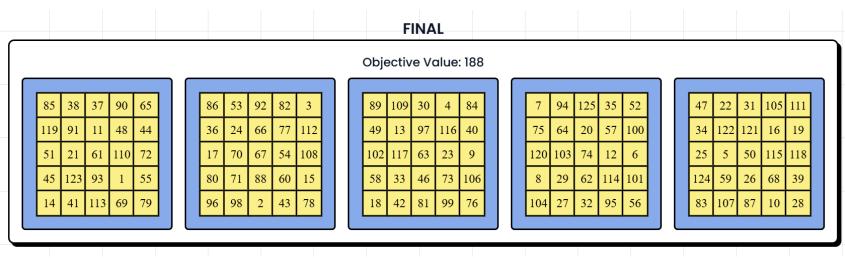
Result

Best Objective : 188
 Total Iteration : 424
 Duration Search : 1916.973 ms

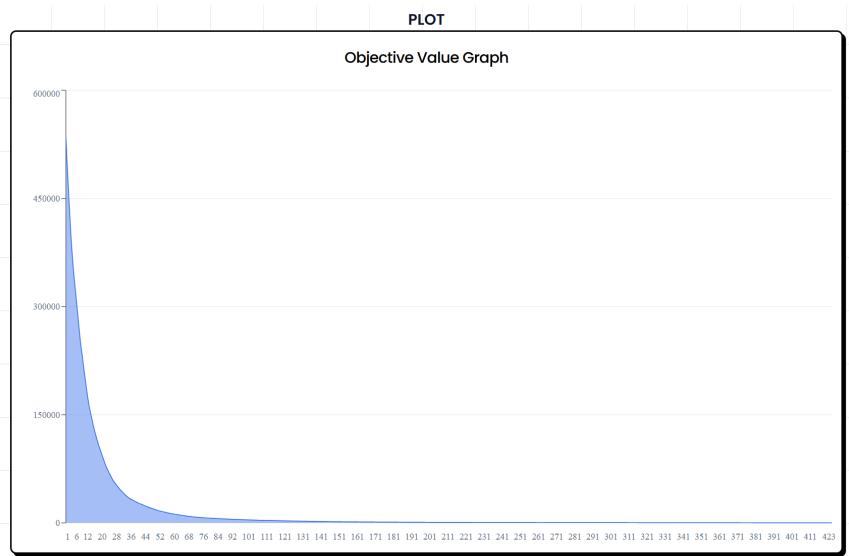
State Awal



State Akhir



Plot Objective Value terhadap Iterasi



Percobaan 2: Max Iteration 100

Objective Value dan Durasi

Result
Best Objective : 198
Total Iteration : 353
Duration Search : 904.084 ms

State Awal

INITIAL

Objective Value: 865138

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

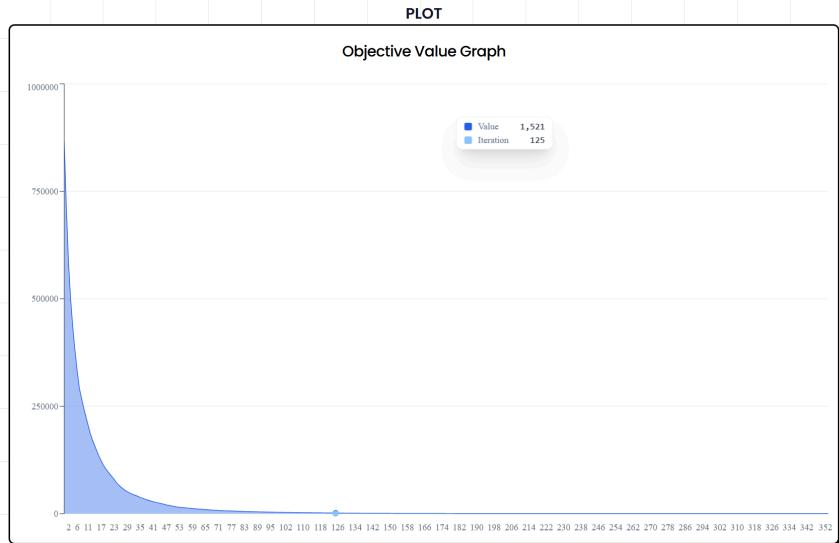
State Akhir

FINAL

Objective Value: 198

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

Plot Objective Value terhadap Iterasi



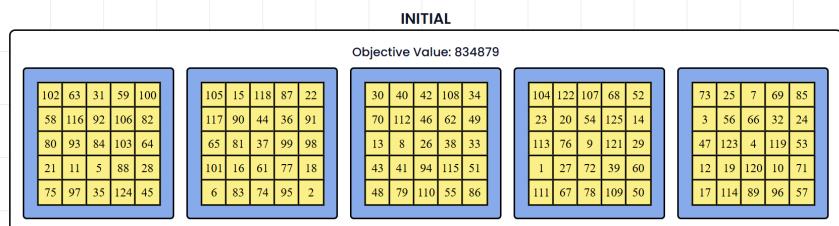
Percobaan 3: Max Iteration 1000

Objective Value dan Durasi

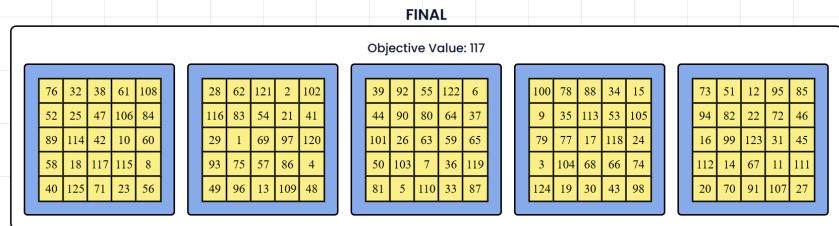
Result

Best Objective : 117
Total Iteration : 370
Duration Search : 952.468 ms

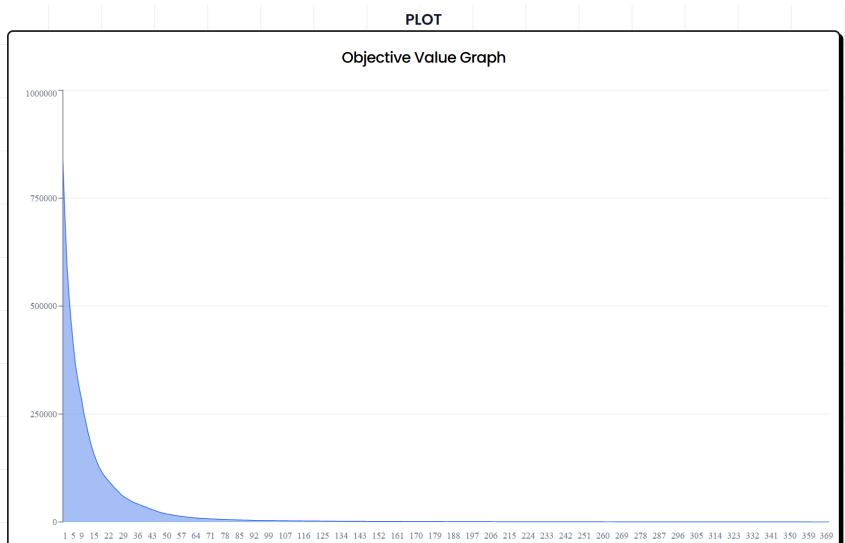
State Awal



State Akhir



Plot Objective Value terhadap Iterasi



C. Hill-Climbing with Random Restart

Percobaan 1: 5 Restart

Objective Value dan Durasi

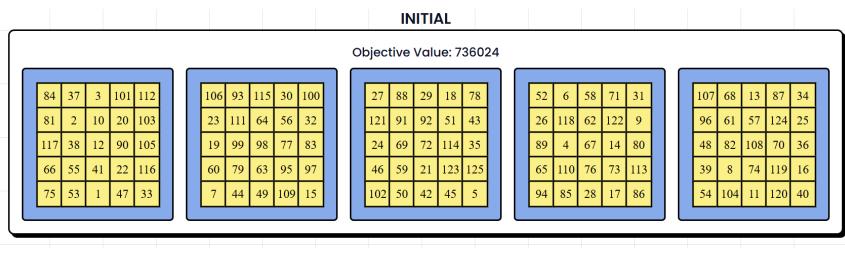
Random Restart Hill Climbing

Best Objective : 195

Restart Iteration : 5

Duration Search : 3380.24 ms

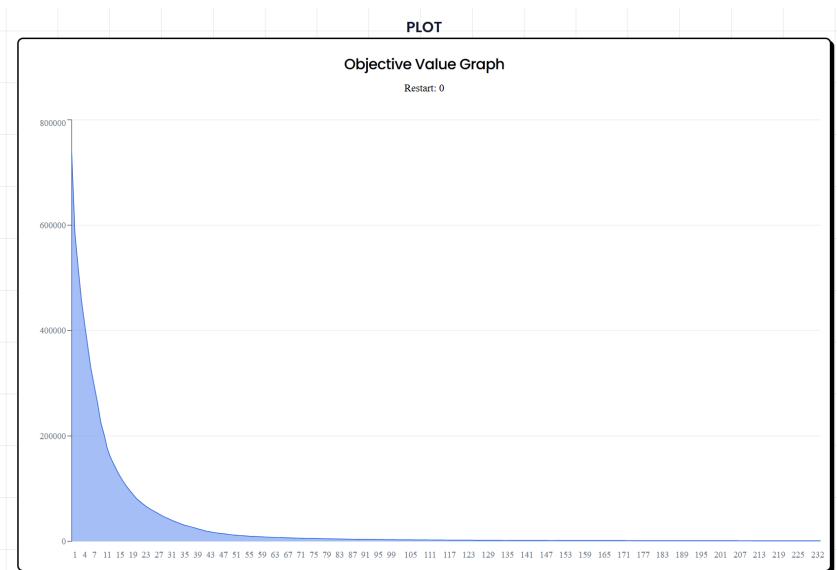
State Awal Iterasi



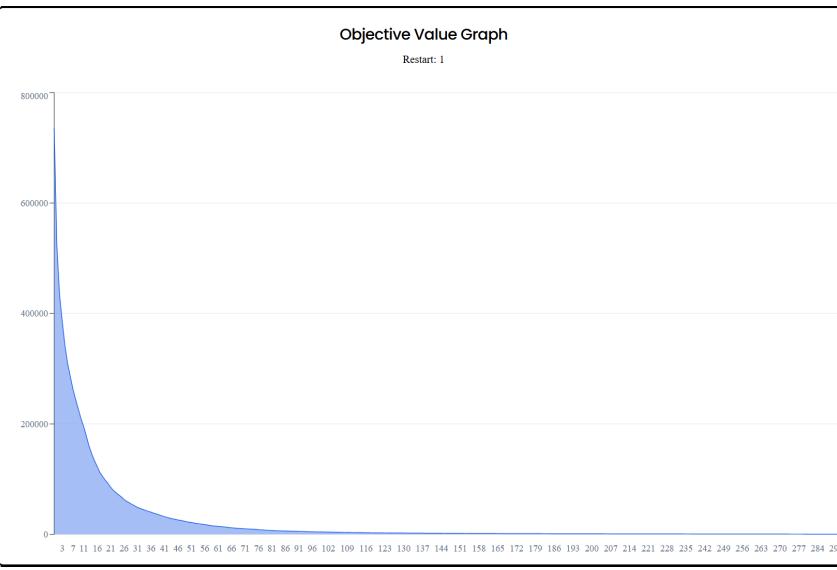
State Akhir Iterasi

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

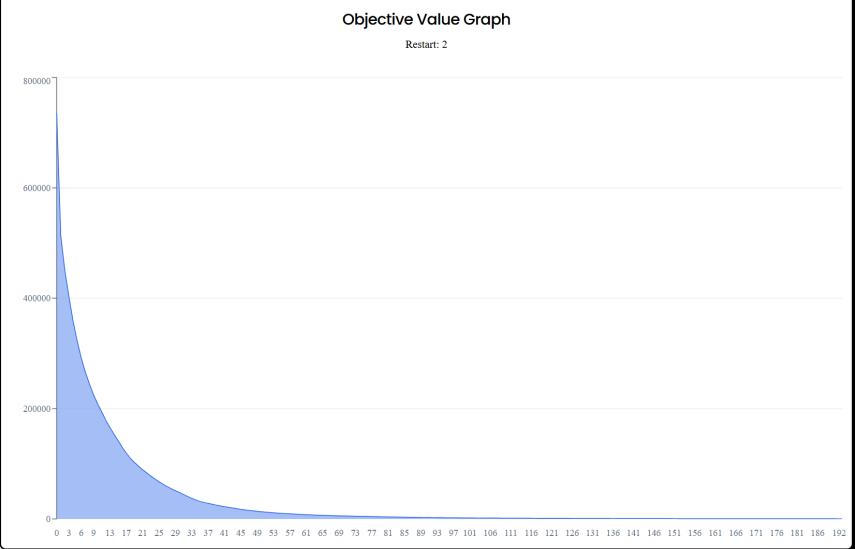
Plot Objective Value terhadap Iterasi 1



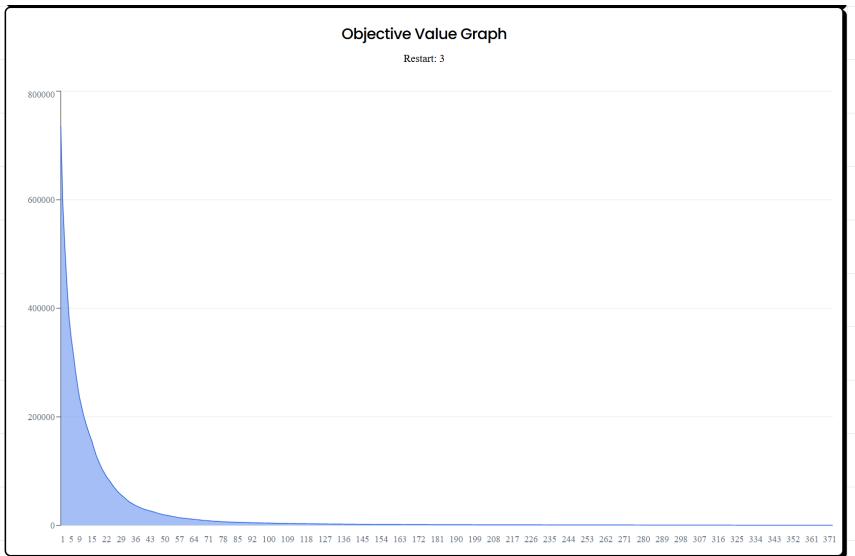
Plot Objective Value terhadap Iterasi 2



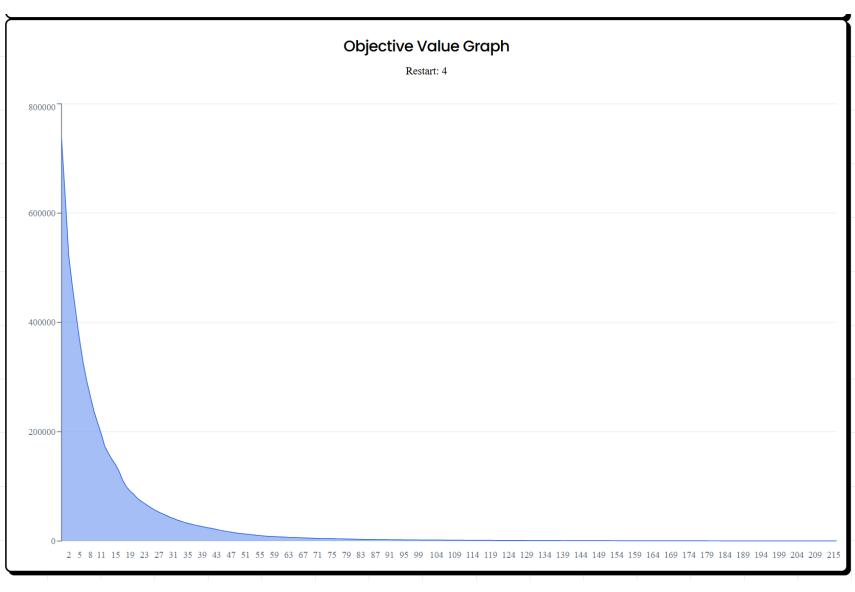
Plot Objective Value terhadap Iterasi 3



Plot Objective Value terhadap Iterasi 4



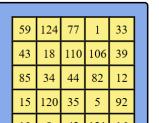
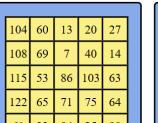
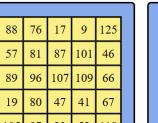
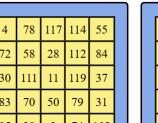
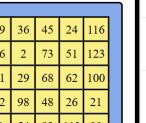
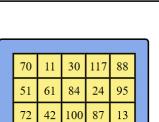
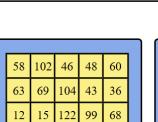
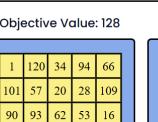
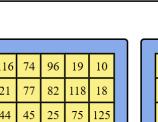
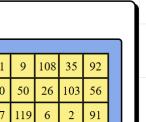
Plot Objective Value terhadap Iterasi 5



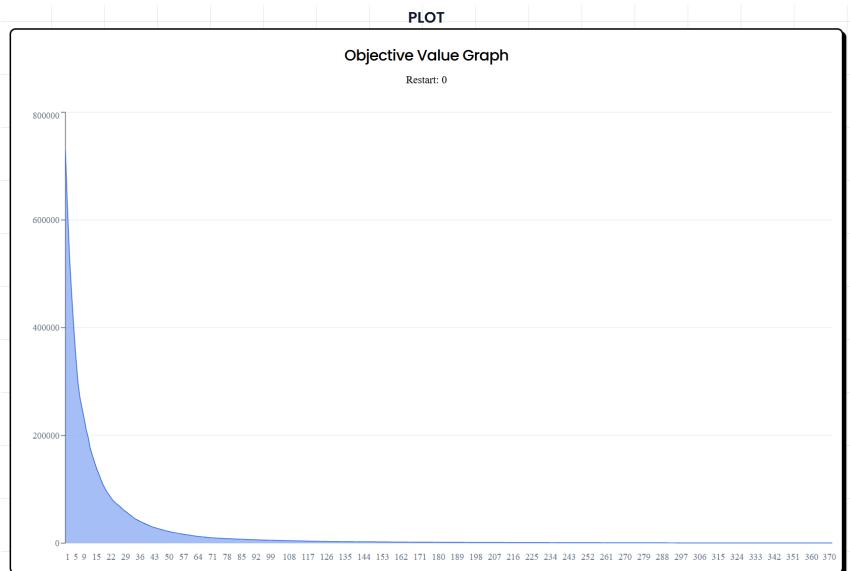
Iteration untuk setiap restart	Iteration for restart 0 : 233
	Iteration for restart 1 : 293
	Iteration for restart 2 : 193
	Iteration for restart 3 : 372
	Iteration for restart 4 : 216

Percobaan 2: 6 Restart

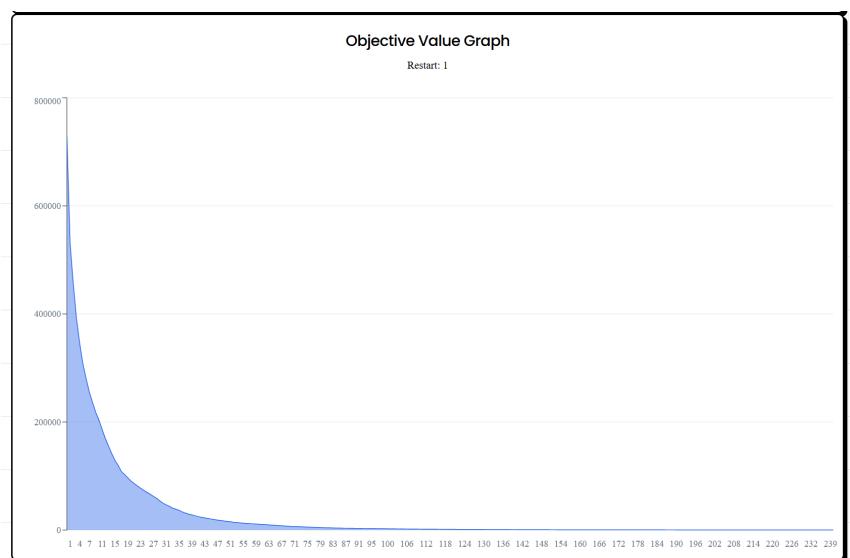
Objective Value dan Durasi	<p>Random Restart Hill Climbing</p> <p>Best Objective : 128</p> <p>Restart Iteration : 6</p> <p>Duration Search : 4197.566 ms</p>
----------------------------	--

State Awal Iterasi	INITIAL Objective Value: 730263     
State Akhir Iterasi	FINAL Objective Value: 128     

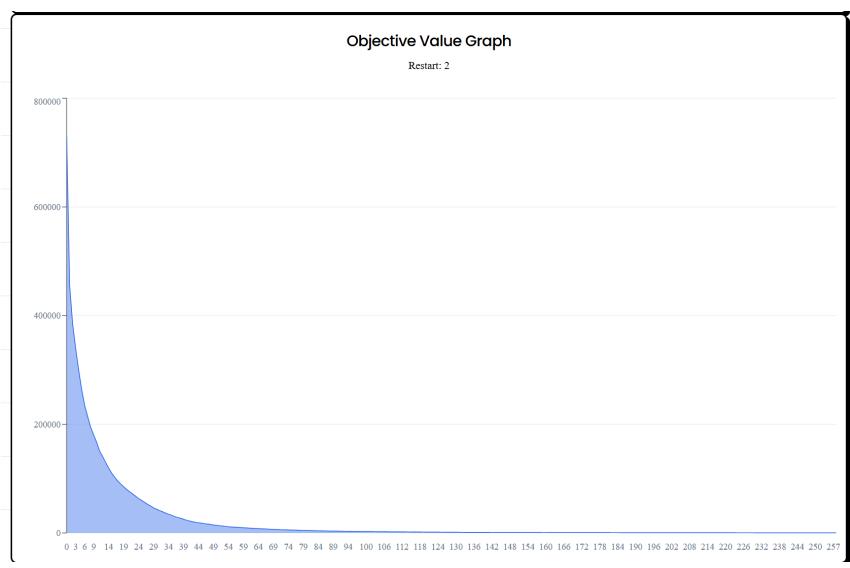
Plot Objective Value terhadap Iterasi 1



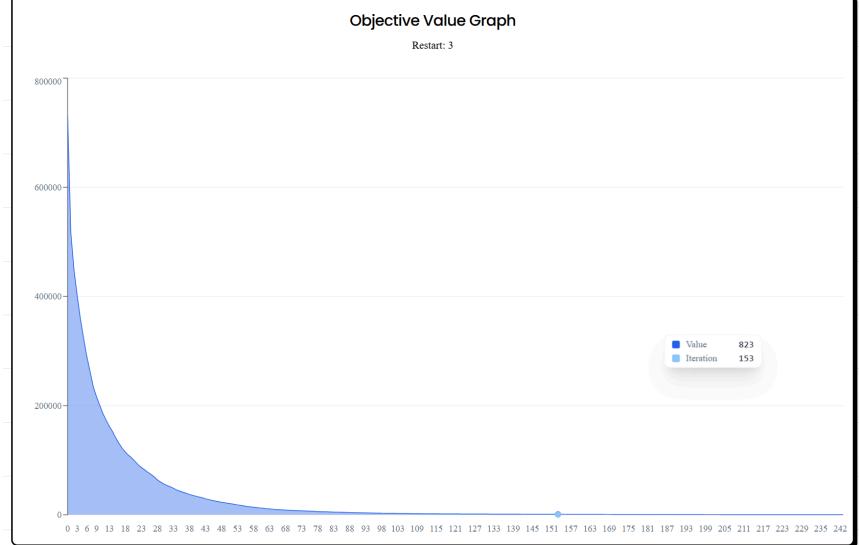
Plot Objective Value terhadap Iterasi 2



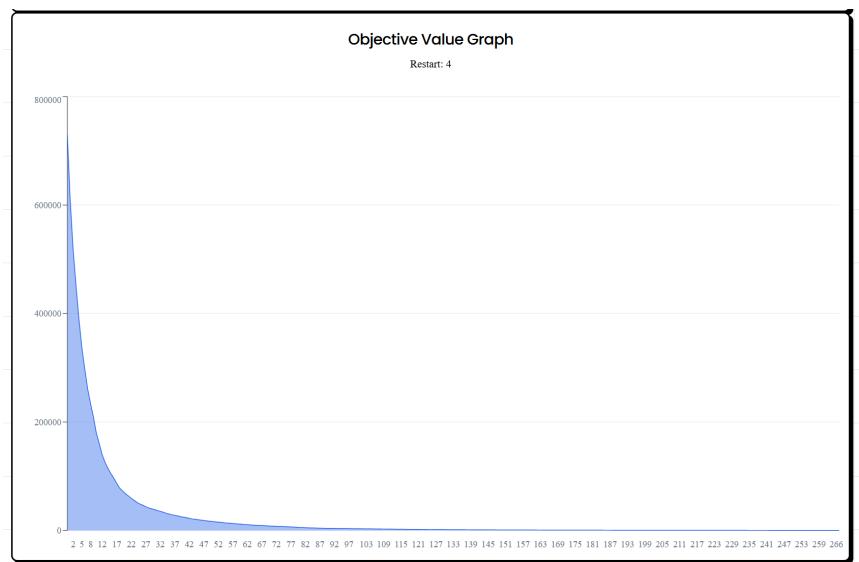
Plot Objective Value terhadap Iterasi 3



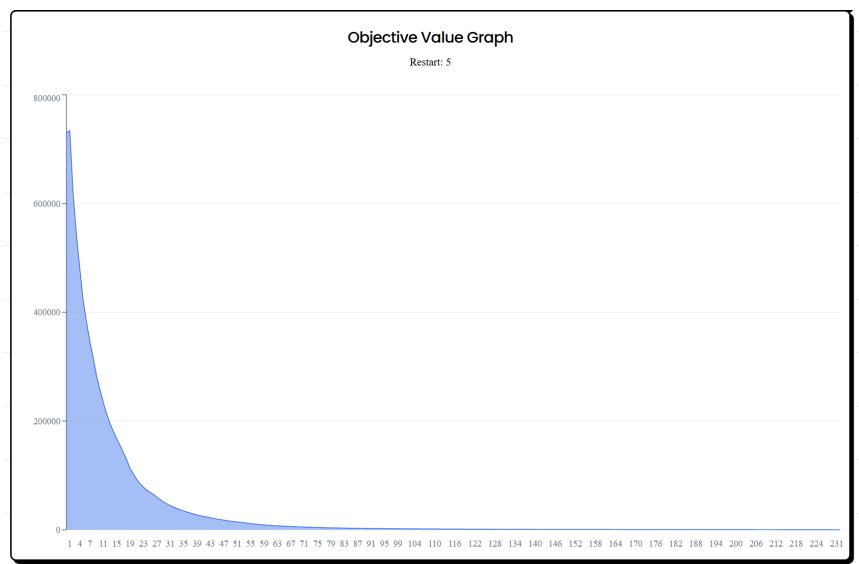
Plot Objective Value terhadap Iterasi 4



Plot Objective Value terhadap Iterasi 5



Plot Objective Value terhadap Iterasi 6



Iteration untuk setiap restart

Iteration for restart 0 : 371

Iteration for restart 1 : 240

Iteration for restart 2 : 258

Iteration for restart 3 : 243

Iteration for restart 4 : 267

Iteration for restart 5 : 232

Percobaan 3: 7 Restart

Objective Value dan Durasi

Random Restart Hill Climbing

Best Objective : 148

Restart Iteration : 7

Duration Search : 4990.691 ms

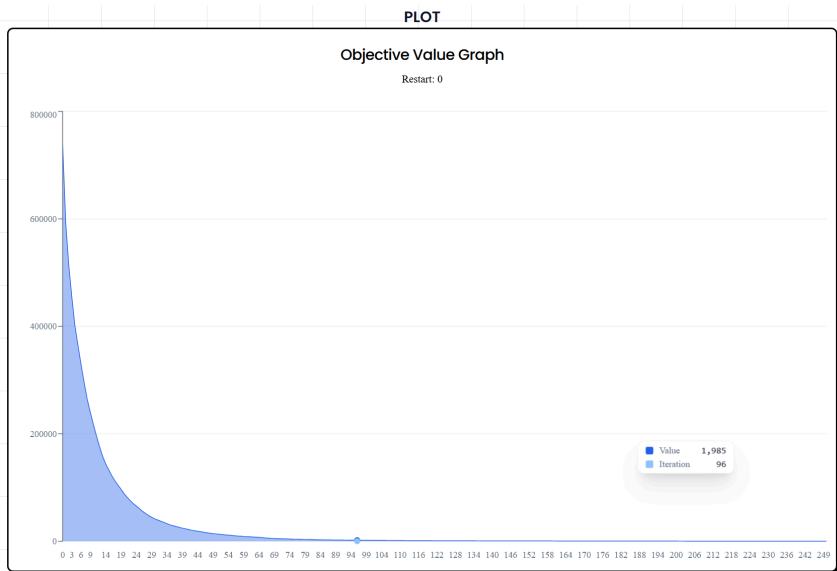
State Awal Iterasi

INITIAL									
Objective Value: 736276									
67	125	93	122	116	24	118	95	103	6
14	54	25	112	86	88	12	72	66	50
23	108	21	106	96	36	10	121	34	80
46	119	69	52	17	100	27	5	42	90
73	33	31	79	91	97	2	113	109	39
81	7	82	37	65	58	57	26	63	45
77	83	32	123	104	30	16	62	84	1
44	98	9	105	68	44	98	9	105	68
117	114	89	56	18	74	20	99	35	29
53	101	124	78	59	40	22	94	13	49
120	15	48	51	3	71	55	115	75	47
111	38	43	8	70	76	4	107	110	61
102	60	41	85	28	19	87	11	64	92
71	55	115	75	47					

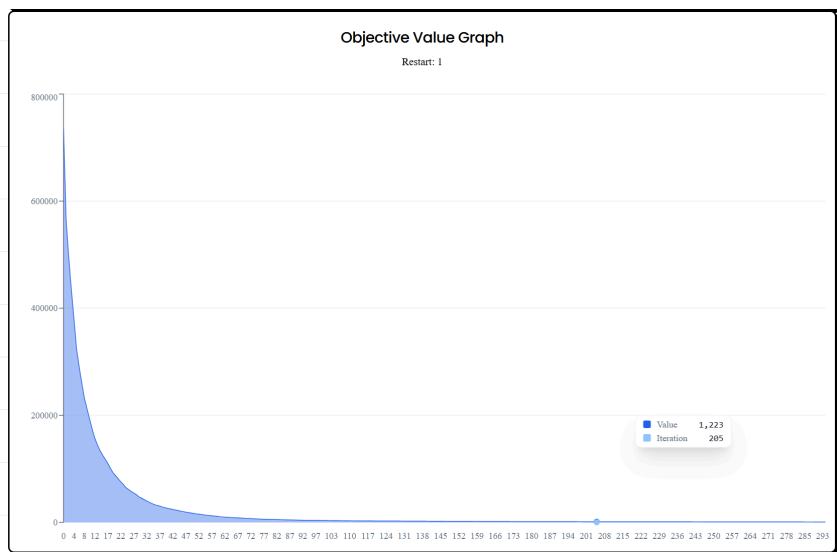
State Akhir Iterasi

FINAL									
Objective Value: 148									
63	4	97	121	29	11	55	83	96	69
25	84	21	80	105	109	104	1	54	48
108	103	35	8	62	53	10	119	14	118
47	100	120	31	18	114	45	2	78	77
72	26	41	75	101	27	102	111	74	3
107	36	42	6	124	57	38	71	59	91
16	94	64	92	49	93	24	116	68	13
43	122	22	90	37	43	122	22	90	37
98	112	5	82	20	79	40	117	61	17
15	51	67	115	66	39	81	70	12	113
23	65	7	125	95	86	30	56	44	99
88	32	85	33	76					

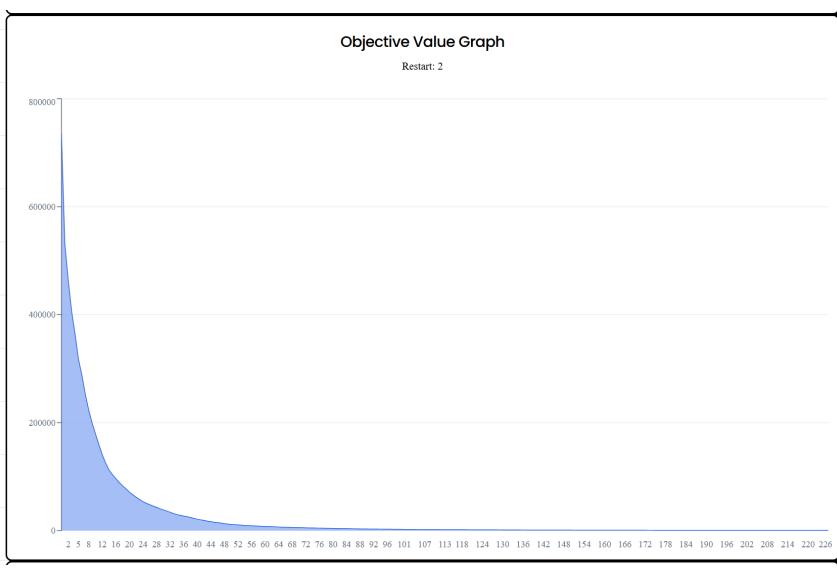
Plot Objective Value terhadap Iterasi 1



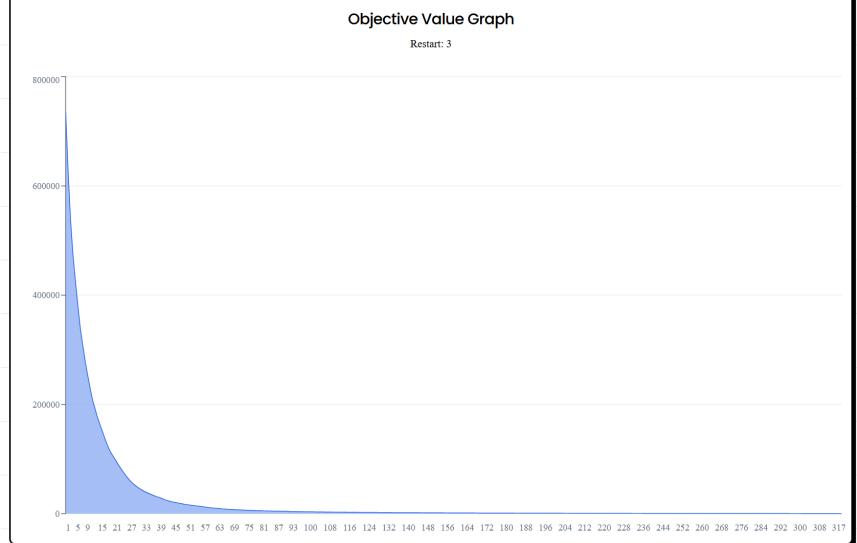
Plot Objective Value terhadap Iterasi 2



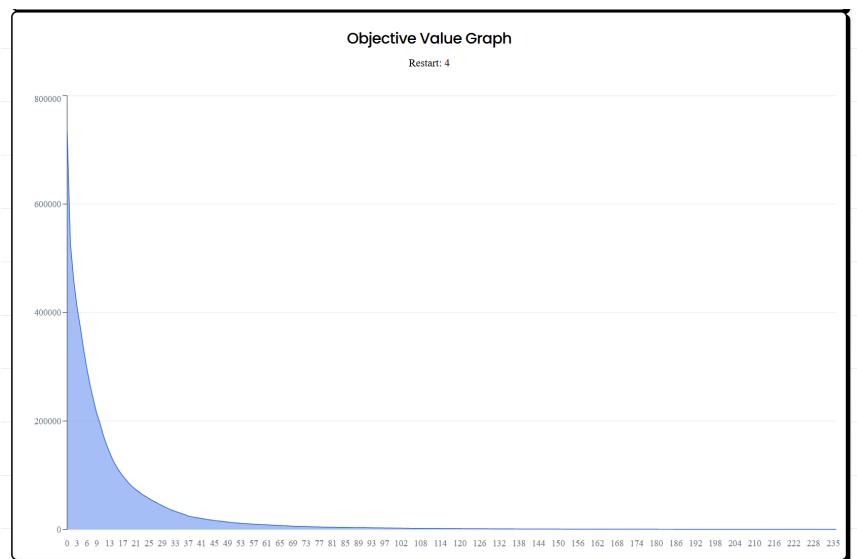
Plot Objective Value terhadap Iterasi 3



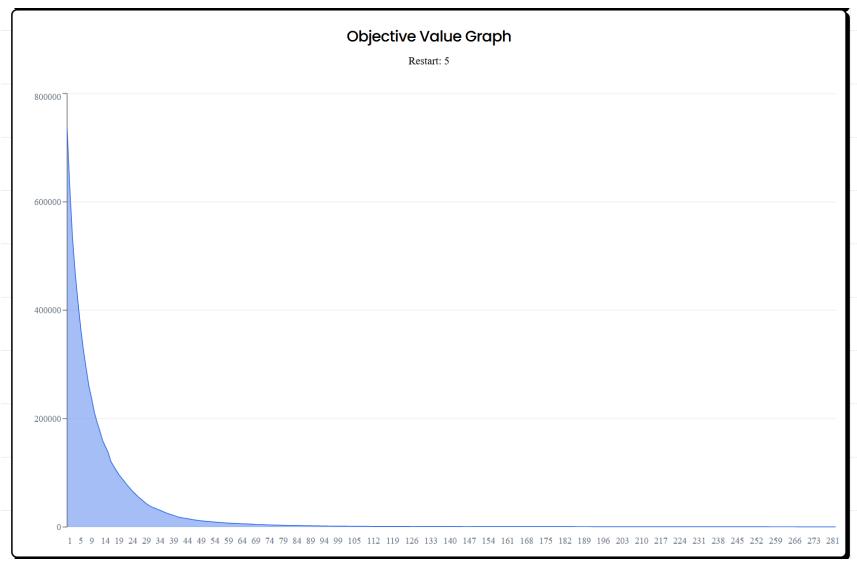
Plot Objective Value terhadap Iterasi 4



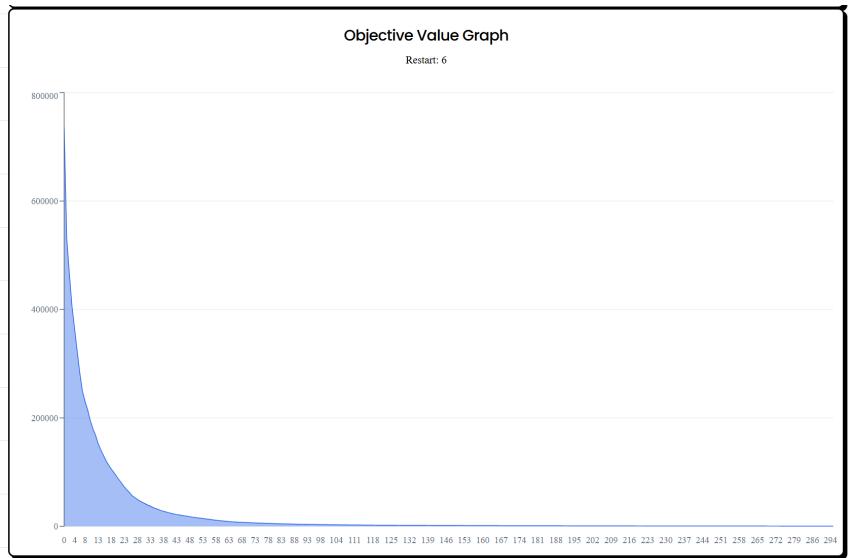
Plot Objective Value terhadap Iterasi 5



Plot Objective Value terhadap Iterasi 5



Plot Objective Value terhadap Iterasi 6



Iteration untuk setiap restart

Iteration for restart 0 : 250

Iteration for restart 1 : 294

Iteration for restart 2 : 227

Iteration for restart 3 : 318

Iteration for restart 4 : 236

Iteration for restart 5 : 282

Iteration for restart 6 : 295

D. Stochastic Hill-Climbing

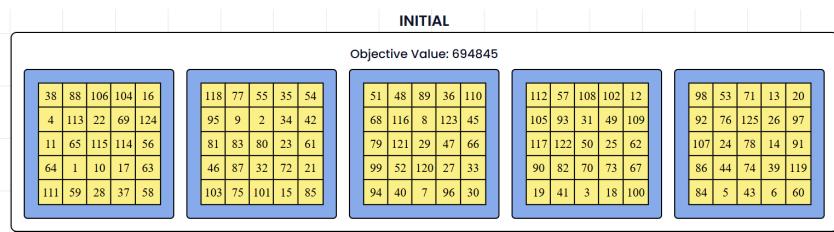
Percobaan 1

Objective Value dan Durasi

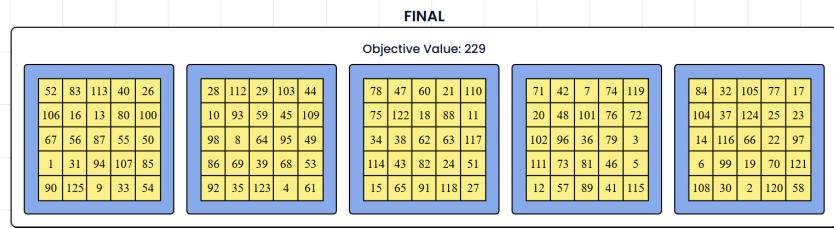
Result

Best Objective : 229
Total Iteration : 800000
Duration Search : 1118.706 ms

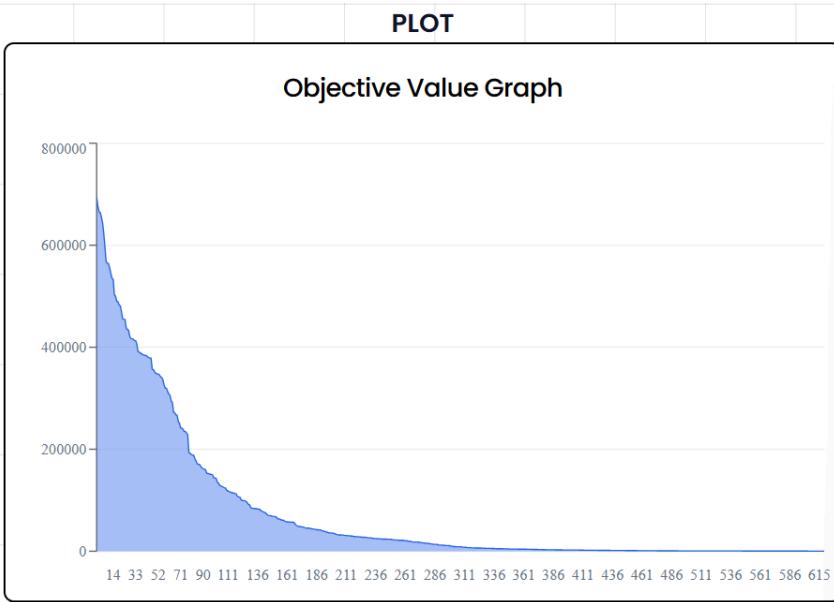
State Awal



State Akhir



Plot Objective Value terhadap Iterasi



Percobaan 2

Objective Value dan Durasi

Result

Best Objective : 205
Total Iteration : 800000
Duration Search : 1122.417 ms

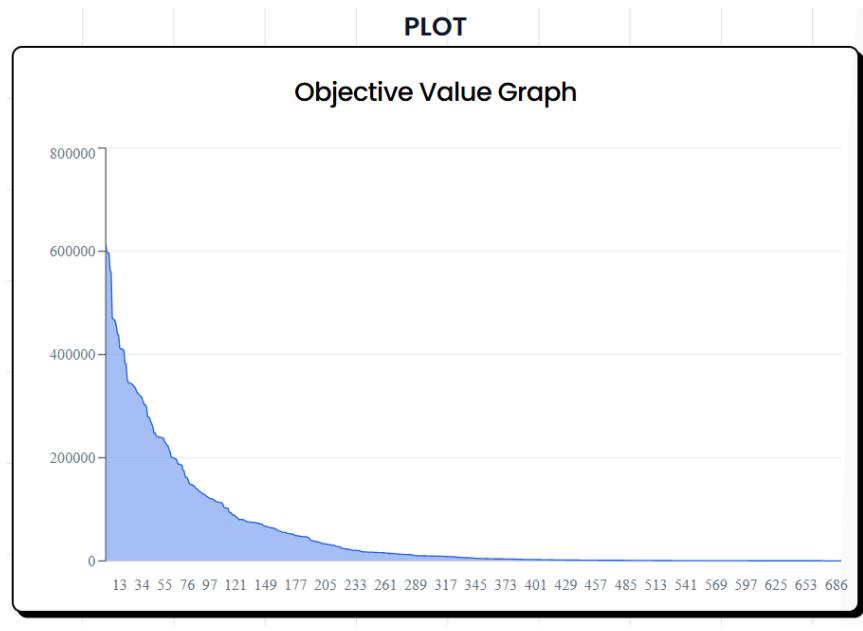
State Awal

INITIAL									
Objective Value: 613775									
110	26	108	53	36	49	125	67	69	84
31	77	111	7	88	71	39	17	34	100
50	97	51	55	5	117	43	99	124	32
104	29	14	90	52	30	45	101	107	95
11	57	96	81	93	116	62	119	42	87
64	21	94	54	65	56	47	70	28	18
61	102	113	9	79	16	85	66	83	38
91	72	15	76	58	6	121	27	63	74
33	12	37	48	73	120	22	2	114	10
89	92	25	105	123	3	20	78	106	118
82	75	8	80	44	19	112	23	24	122
60	86	40	115	41	35	98	109	46	4

State Akhir

FINAL									
Objective Value: 205									
87	28	34	119	48	17	112	99	64	24
29	59	114	4	109	79	60	11	55	111
10	123	90	71	22	80	15	103	101	16
92	78	62	12	70	49	43	8	93	122
97	26	18	107	66	91	86	95	3	41
54	19	67	94	83	98	47	77	88	6
37	125	63	14	72	121	76	65	25	30
106	84	1	7	117	5	46	42	96	124
82	45	113	68	9	69	13	38	108	85
23	58	105	75	53	35	116	57	56	51
50	73	115	33	44	31	61	74	110	39
27	104	2	100	81	118	36	21	20	120
89	40	102	52	32					

Plot Objective Value terhadap Iterasi



Percobaan 3

Objective Value dan Durasi

Result

Best Objective : 302
Total Iteration : 800000
Duration Search : 1125.872 ms

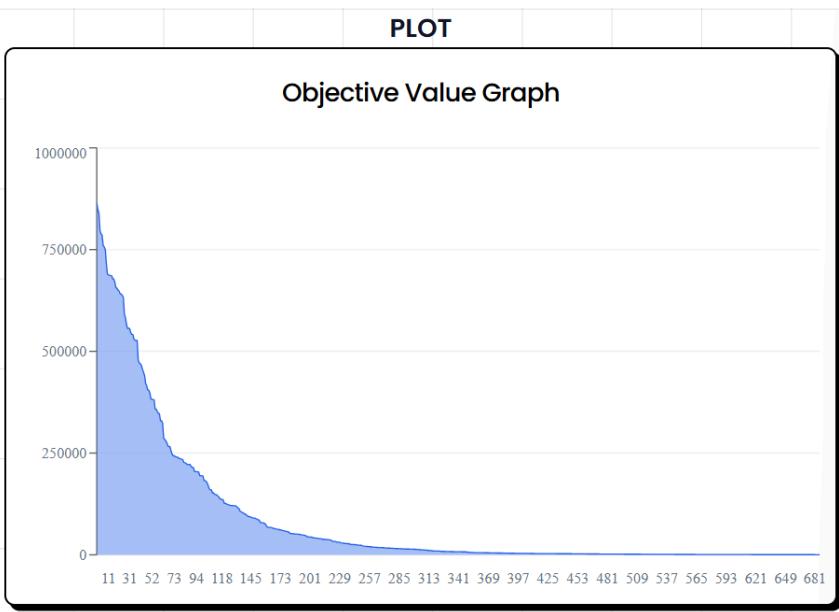
State Awal

INITIAL											
Objective Value: 618070											
45	99	79	58	109							
89	70	87	9	2							
26	57	29	95	40							
105	69	78	43	15							
38	56	34	62	48							
61	32	64	121	115							
8	97	18	23	81							
112	82	66	1	19							
14	55	30	85	106							
124	41	63	71	50							
123	20	60	13	37							
90	36	72	119	46							
75	101	76	52	22							
93	33	74	51	98							
49	53	12	25	42							
103	21	108	118	111							
122	80	24	47	114							
44	4	17	39	35							
67	73	16	84	7							
100	77	113	120	28							
68	116	6	107	117							
10	94	11	31	104							
5	3	88	54	91							
65	102	96	125	86							
92	59	110	27	83							

State Akhir

FINAL											
Objective Value: 535											
78	62	15	71	89							
87	102	51	45	28							
18	93	26	125	50							
42	57	98	39	80							
88	1	124	33	70							
19	65	48	113	69							
25	66	118	23	85							
109	106	96	2	4							
56	47	20	83	108							
105	30	34	97	49							
122	43	119	24	8							
7	5	86	101	117							
58	36	38	116	67							
114	112	44	35	10							
13	120	27	40	115							
74	32	91	11	107							
73	64	54	55	68							
9	63	79	61	103							
100	72	53	75	14							
60	84	37	111	22							
21	110	41	95	46							
123	77	6	94	17							
121	16	76	12	90							
3	29	99	82	104							
52	81	92	31	59							

Plot Objective Value terhadap Iterasi

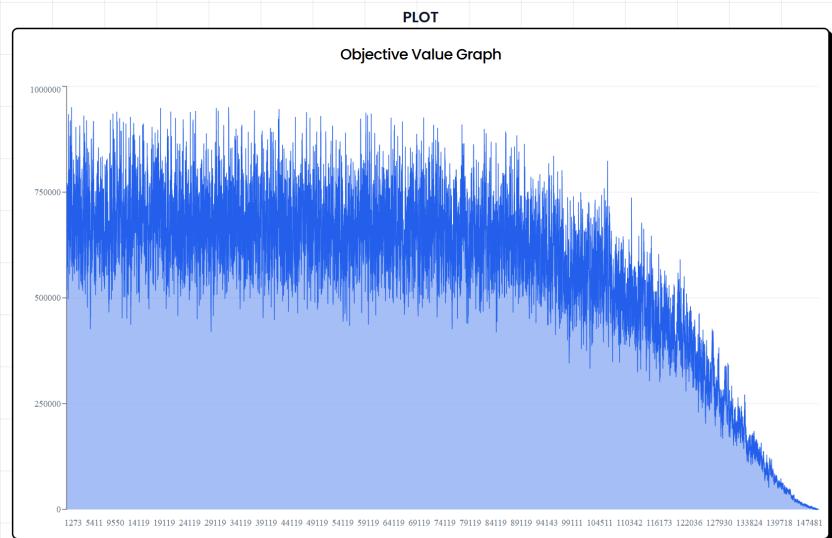


E. Simulated Annealing

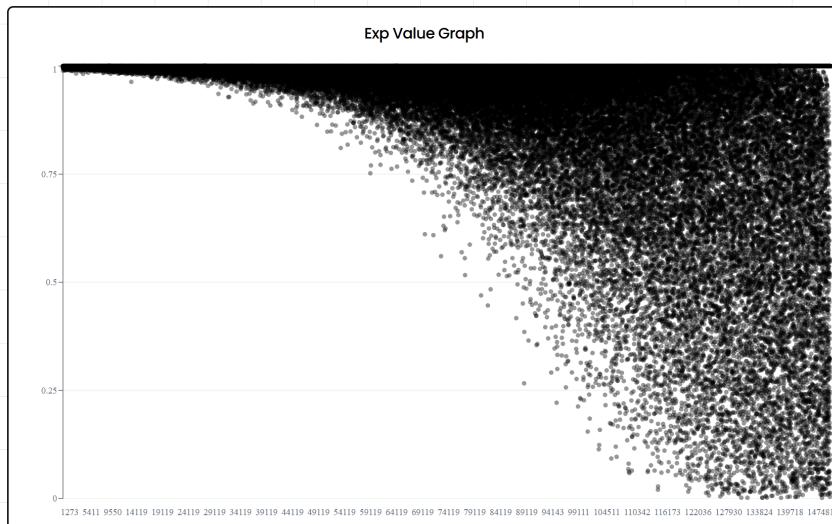
Percobaan 1

Objective Value dan Durasi	<p style="text-align: center;">Result</p> <p>Stuck Frequency: 73836</p> <p>Best Objective : 84</p> <p>Total Iteration : 412348</p> <p>Duration Search : 622.29 ms</p>																																																																																																																													
State Awal	<p style="text-align: center;">INITIAL</p> <p>Objective Value: 630414</p> <p>The initial state shows a 5x5 grid of numbers from 1 to 125. The objective value is 630414.</p> <table border="1"><tr><td>107</td><td>89</td><td>16</td><td>22</td><td>77</td></tr><tr><td>96</td><td>31</td><td>88</td><td>48</td><td>30</td></tr><tr><td>36</td><td>100</td><td>66</td><td>104</td><td>63</td></tr><tr><td>78</td><td>44</td><td>70</td><td>33</td><td>24</td></tr><tr><td>61</td><td>94</td><td>80</td><td>53</td><td>38</td></tr></table> <table border="1"><tr><td>34</td><td>106</td><td>64</td><td>29</td><td>46</td></tr><tr><td>50</td><td>1</td><td>10</td><td>42</td><td>93</td></tr><tr><td>51</td><td>4</td><td>3</td><td>45</td><td>86</td></tr><tr><td>13</td><td>7</td><td>112</td><td>91</td><td>49</td></tr><tr><td>90</td><td>75</td><td>95</td><td>60</td><td>79</td></tr></table> <table border="1"><tr><td>114</td><td>102</td><td>123</td><td>59</td><td>83</td></tr><tr><td>103</td><td>72</td><td>82</td><td>57</td><td>76</td></tr><tr><td>9</td><td>47</td><td>116</td><td>56</td><td>105</td></tr><tr><td>115</td><td>6</td><td>119</td><td>27</td><td>12</td></tr><tr><td>98</td><td>17</td><td>121</td><td>125</td><td>41</td></tr></table> <table border="1"><tr><td>74</td><td>101</td><td>32</td><td>26</td><td>124</td></tr><tr><td>54</td><td>120</td><td>21</td><td>39</td><td>92</td></tr><tr><td>19</td><td>25</td><td>58</td><td>113</td><td>2</td></tr><tr><td>52</td><td>69</td><td>43</td><td>62</td><td>40</td></tr><tr><td>111</td><td>55</td><td>11</td><td>18</td><td>118</td></tr></table> <table border="1"><tr><td>68</td><td>97</td><td>85</td><td>109</td><td>67</td></tr><tr><td>65</td><td>37</td><td>73</td><td>35</td><td>5</td></tr><tr><td>71</td><td>14</td><td>23</td><td>8</td><td>122</td></tr><tr><td>15</td><td>84</td><td>87</td><td>99</td><td>108</td></tr><tr><td>20</td><td>110</td><td>28</td><td>117</td><td>81</td></tr></table>	107	89	16	22	77	96	31	88	48	30	36	100	66	104	63	78	44	70	33	24	61	94	80	53	38	34	106	64	29	46	50	1	10	42	93	51	4	3	45	86	13	7	112	91	49	90	75	95	60	79	114	102	123	59	83	103	72	82	57	76	9	47	116	56	105	115	6	119	27	12	98	17	121	125	41	74	101	32	26	124	54	120	21	39	92	19	25	58	113	2	52	69	43	62	40	111	55	11	18	118	68	97	85	109	67	65	37	73	35	5	71	14	23	8	122	15	84	87	99	108	20	110	28	117	81
107	89	16	22	77																																																																																																																										
96	31	88	48	30																																																																																																																										
36	100	66	104	63																																																																																																																										
78	44	70	33	24																																																																																																																										
61	94	80	53	38																																																																																																																										
34	106	64	29	46																																																																																																																										
50	1	10	42	93																																																																																																																										
51	4	3	45	86																																																																																																																										
13	7	112	91	49																																																																																																																										
90	75	95	60	79																																																																																																																										
114	102	123	59	83																																																																																																																										
103	72	82	57	76																																																																																																																										
9	47	116	56	105																																																																																																																										
115	6	119	27	12																																																																																																																										
98	17	121	125	41																																																																																																																										
74	101	32	26	124																																																																																																																										
54	120	21	39	92																																																																																																																										
19	25	58	113	2																																																																																																																										
52	69	43	62	40																																																																																																																										
111	55	11	18	118																																																																																																																										
68	97	85	109	67																																																																																																																										
65	37	73	35	5																																																																																																																										
71	14	23	8	122																																																																																																																										
15	84	87	99	108																																																																																																																										
20	110	28	117	81																																																																																																																										
State Akhir	<p style="text-align: center;">FINAL</p> <p>Objective Value: 84</p> <p>The final state shows a 5x5 grid of numbers from 1 to 125. The objective value is 84.</p> <table border="1"><tr><td>48</td><td>73</td><td>56</td><td>107</td><td>30</td></tr><tr><td>47</td><td>22</td><td>122</td><td>18</td><td>106</td></tr><tr><td>49</td><td>20</td><td>102</td><td>92</td><td>52</td></tr><tr><td>90</td><td>85</td><td>4</td><td>76</td><td>60</td></tr><tr><td>80</td><td>115</td><td>31</td><td>21</td><td>67</td></tr></table> <table border="1"><tr><td>23</td><td>1</td><td>78</td><td>89</td><td>125</td></tr><tr><td>25</td><td>111</td><td>50</td><td>17</td><td>112</td></tr><tr><td>119</td><td>123</td><td>26</td><td>34</td><td>12</td></tr><tr><td>40</td><td>39</td><td>118</td><td>104</td><td>15</td></tr><tr><td>108</td><td>43</td><td>42</td><td>71</td><td>51</td></tr></table> <table border="1"><tr><td>65</td><td>84</td><td>103</td><td>5</td><td>58</td></tr><tr><td>82</td><td>46</td><td>88</td><td>96</td><td>3</td></tr><tr><td>57</td><td>14</td><td>63</td><td>105</td><td>75</td></tr><tr><td>87</td><td>74</td><td>32</td><td>41</td><td>81</td></tr><tr><td>24</td><td>97</td><td>28</td><td>69</td><td>99</td></tr></table> <table border="1"><tr><td>124</td><td>38</td><td>19</td><td>98</td><td>35</td></tr><tr><td>83</td><td>27</td><td>44</td><td>68</td><td>93</td></tr><tr><td>10</td><td>86</td><td>94</td><td>13</td><td>113</td></tr><tr><td>91</td><td>110</td><td>45</td><td>33</td><td>36</td></tr><tr><td>8</td><td>54</td><td>114</td><td>101</td><td>37</td></tr></table> <table border="1"><tr><td>55</td><td>120</td><td>59</td><td>16</td><td>66</td></tr><tr><td>77</td><td>109</td><td>11</td><td>116</td><td>2</td></tr><tr><td>79</td><td>72</td><td>29</td><td>70</td><td>64</td></tr><tr><td>9</td><td>7</td><td>117</td><td>61</td><td>121</td></tr><tr><td>95</td><td>6</td><td>100</td><td>53</td><td>62</td></tr></table>	48	73	56	107	30	47	22	122	18	106	49	20	102	92	52	90	85	4	76	60	80	115	31	21	67	23	1	78	89	125	25	111	50	17	112	119	123	26	34	12	40	39	118	104	15	108	43	42	71	51	65	84	103	5	58	82	46	88	96	3	57	14	63	105	75	87	74	32	41	81	24	97	28	69	99	124	38	19	98	35	83	27	44	68	93	10	86	94	13	113	91	110	45	33	36	8	54	114	101	37	55	120	59	16	66	77	109	11	116	2	79	72	29	70	64	9	7	117	61	121	95	6	100	53	62
48	73	56	107	30																																																																																																																										
47	22	122	18	106																																																																																																																										
49	20	102	92	52																																																																																																																										
90	85	4	76	60																																																																																																																										
80	115	31	21	67																																																																																																																										
23	1	78	89	125																																																																																																																										
25	111	50	17	112																																																																																																																										
119	123	26	34	12																																																																																																																										
40	39	118	104	15																																																																																																																										
108	43	42	71	51																																																																																																																										
65	84	103	5	58																																																																																																																										
82	46	88	96	3																																																																																																																										
57	14	63	105	75																																																																																																																										
87	74	32	41	81																																																																																																																										
24	97	28	69	99																																																																																																																										
124	38	19	98	35																																																																																																																										
83	27	44	68	93																																																																																																																										
10	86	94	13	113																																																																																																																										
91	110	45	33	36																																																																																																																										
8	54	114	101	37																																																																																																																										
55	120	59	16	66																																																																																																																										
77	109	11	116	2																																																																																																																										
79	72	29	70	64																																																																																																																										
9	7	117	61	121																																																																																																																										
95	6	100	53	62																																																																																																																										

Plot Objective Value terhadap Iterasi



Plot $e^{\Delta E/T}$ terhadap iterasi



Percobaan 2

Objective Value dan Durasi

Result

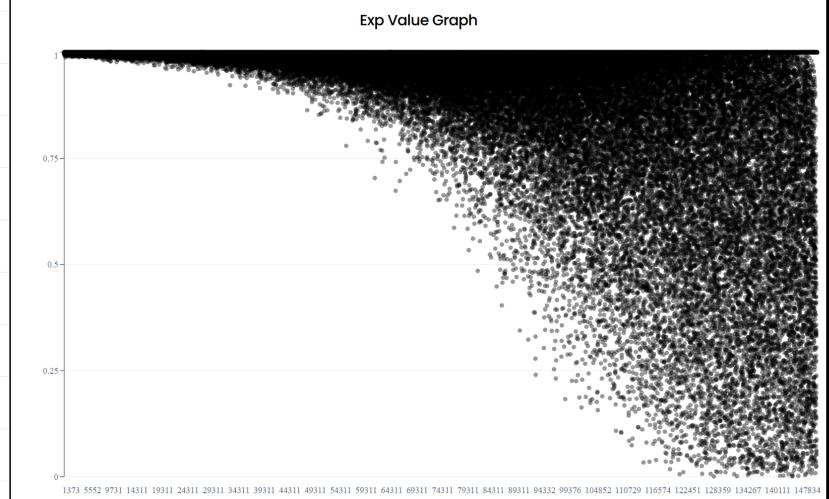
Stuck Frequency : 73819
Best Objective : 101
Total Iteration : 412348
Duration Search : 625.446 ms

State Awal	<p style="text-align: center;">INITIAL</p> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>35</td><td>94</td><td>69</td><td>63</td><td>107</td></tr> <tr><td>45</td><td>79</td><td>98</td><td>92</td><td>125</td></tr> <tr><td>87</td><td>112</td><td>77</td><td>8</td><td>33</td></tr> <tr><td>95</td><td>26</td><td>106</td><td>110</td><td>65</td></tr> <tr><td>114</td><td>82</td><td>85</td><td>66</td><td>25</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>111</td><td>60</td><td>93</td><td>58</td><td>121</td></tr> <tr><td>62</td><td>100</td><td>55</td><td>99</td><td>59</td></tr> <tr><td>72</td><td>71</td><td>81</td><td>118</td><td>18</td></tr> <tr><td>76</td><td>21</td><td>109</td><td>14</td><td>101</td></tr> <tr><td>9</td><td>105</td><td>53</td><td>46</td><td>15</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>80</td><td>83</td><td>48</td><td>103</td><td>97</td></tr> <tr><td>120</td><td>86</td><td>117</td><td>1</td><td>12</td></tr> <tr><td>64</td><td>23</td><td>70</td><td>32</td><td>7</td></tr> <tr><td>3</td><td>52</td><td>20</td><td>119</td><td>40</td></tr> <tr><td>113</td><td>78</td><td>61</td><td>31</td><td>104</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>88</td><td>10</td><td>19</td><td>68</td><td>108</td></tr> <tr><td>49</td><td>39</td><td>13</td><td>116</td><td>115</td></tr> <tr><td>44</td><td>28</td><td>6</td><td>5</td><td>29</td></tr> <tr><td>56</td><td>124</td><td>42</td><td>96</td><td>74</td></tr> <tr><td>47</td><td>11</td><td>90</td><td>41</td><td>2</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>30</td><td>4</td><td>67</td><td>123</td><td>43</td></tr> <tr><td>37</td><td>36</td><td>57</td><td>102</td><td>54</td></tr> <tr><td>89</td><td>34</td><td>16</td><td>84</td><td>51</td></tr> <tr><td>24</td><td>50</td><td>38</td><td>75</td><td>122</td></tr> <tr><td>27</td><td>22</td><td>17</td><td>91</td><td>73</td></tr> </table>	35	94	69	63	107	45	79	98	92	125	87	112	77	8	33	95	26	106	110	65	114	82	85	66	25	111	60	93	58	121	62	100	55	99	59	72	71	81	118	18	76	21	109	14	101	9	105	53	46	15	80	83	48	103	97	120	86	117	1	12	64	23	70	32	7	3	52	20	119	40	113	78	61	31	104	88	10	19	68	108	49	39	13	116	115	44	28	6	5	29	56	124	42	96	74	47	11	90	41	2	30	4	67	123	43	37	36	57	102	54	89	34	16	84	51	24	50	38	75	122	27	22	17	91	73
35	94	69	63	107																																																																																																																										
45	79	98	92	125																																																																																																																										
87	112	77	8	33																																																																																																																										
95	26	106	110	65																																																																																																																										
114	82	85	66	25																																																																																																																										
111	60	93	58	121																																																																																																																										
62	100	55	99	59																																																																																																																										
72	71	81	118	18																																																																																																																										
76	21	109	14	101																																																																																																																										
9	105	53	46	15																																																																																																																										
80	83	48	103	97																																																																																																																										
120	86	117	1	12																																																																																																																										
64	23	70	32	7																																																																																																																										
3	52	20	119	40																																																																																																																										
113	78	61	31	104																																																																																																																										
88	10	19	68	108																																																																																																																										
49	39	13	116	115																																																																																																																										
44	28	6	5	29																																																																																																																										
56	124	42	96	74																																																																																																																										
47	11	90	41	2																																																																																																																										
30	4	67	123	43																																																																																																																										
37	36	57	102	54																																																																																																																										
89	34	16	84	51																																																																																																																										
24	50	38	75	122																																																																																																																										
27	22	17	91	73																																																																																																																										
State Akhir	<p style="text-align: center;">FINAL</p> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>59</td><td>64</td><td>58</td><td>55</td><td>79</td></tr> <tr><td>8</td><td>74</td><td>95</td><td>122</td><td>14</td></tr> <tr><td>100</td><td>45</td><td>52</td><td>90</td><td>27</td></tr> <tr><td>106</td><td>21</td><td>37</td><td>43</td><td>108</td></tr> <tr><td>42</td><td>110</td><td>73</td><td>4</td><td>87</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>118</td><td>46</td><td>15</td><td>24</td><td>112</td></tr> <tr><td>32</td><td>72</td><td>99</td><td>36</td><td>76</td></tr> <tr><td>5</td><td>124</td><td>9</td><td>88</td><td>89</td></tr> <tr><td>56</td><td>53</td><td>82</td><td>102</td><td>22</td></tr> <tr><td>103</td><td>20</td><td>109</td><td>67</td><td>16</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>84</td><td>12</td><td>70</td><td>116</td><td>33</td></tr> <tr><td>31</td><td>49</td><td>107</td><td>3</td><td>125</td></tr> <tr><td>69</td><td>41</td><td>63</td><td>65</td><td>78</td></tr> <tr><td>30</td><td>114</td><td>1</td><td>105</td><td>66</td></tr> <tr><td>101</td><td>98</td><td>75</td><td>28</td><td>13</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>25</td><td>115</td><td>80</td><td>71</td><td>23</td></tr> <tr><td>121</td><td>35</td><td>6</td><td>92</td><td>61</td></tr> <tr><td>47</td><td>44</td><td>97</td><td>17</td><td>111</td></tr> <tr><td>104</td><td>86</td><td>83</td><td>39</td><td>2</td></tr> <tr><td>18</td><td>34</td><td>48</td><td>96</td><td>119</td></tr> </table> <table border="1" style="margin: auto; border-collapse: collapse;"> <tr><td>29</td><td>77</td><td>91</td><td>50</td><td>68</td></tr> <tr><td>123</td><td>85</td><td>7</td><td>62</td><td>38</td></tr> <tr><td>93</td><td>60</td><td>94</td><td>57</td><td>11</td></tr> <tr><td>19</td><td>40</td><td>113</td><td>26</td><td>117</td></tr> <tr><td>51</td><td>54</td><td>10</td><td>120</td><td>81</td></tr> </table>	59	64	58	55	79	8	74	95	122	14	100	45	52	90	27	106	21	37	43	108	42	110	73	4	87	118	46	15	24	112	32	72	99	36	76	5	124	9	88	89	56	53	82	102	22	103	20	109	67	16	84	12	70	116	33	31	49	107	3	125	69	41	63	65	78	30	114	1	105	66	101	98	75	28	13	25	115	80	71	23	121	35	6	92	61	47	44	97	17	111	104	86	83	39	2	18	34	48	96	119	29	77	91	50	68	123	85	7	62	38	93	60	94	57	11	19	40	113	26	117	51	54	10	120	81
59	64	58	55	79																																																																																																																										
8	74	95	122	14																																																																																																																										
100	45	52	90	27																																																																																																																										
106	21	37	43	108																																																																																																																										
42	110	73	4	87																																																																																																																										
118	46	15	24	112																																																																																																																										
32	72	99	36	76																																																																																																																										
5	124	9	88	89																																																																																																																										
56	53	82	102	22																																																																																																																										
103	20	109	67	16																																																																																																																										
84	12	70	116	33																																																																																																																										
31	49	107	3	125																																																																																																																										
69	41	63	65	78																																																																																																																										
30	114	1	105	66																																																																																																																										
101	98	75	28	13																																																																																																																										
25	115	80	71	23																																																																																																																										
121	35	6	92	61																																																																																																																										
47	44	97	17	111																																																																																																																										
104	86	83	39	2																																																																																																																										
18	34	48	96	119																																																																																																																										
29	77	91	50	68																																																																																																																										
123	85	7	62	38																																																																																																																										
93	60	94	57	11																																																																																																																										
19	40	113	26	117																																																																																																																										
51	54	10	120	81																																																																																																																										
Plot Objective Value terhadap Iterasi	<p style="text-align: center;">PLOT</p> <p style="text-align: center;">Objective Value Graph</p>																																																																																																																													
Plot $e^{\Delta E/T}$ terhadap iterasi	<p style="text-align: center;">Exp Value Graph</p>																																																																																																																													

Percobaan 3

Objective Value dan Durasi	<h2>Result</h2> <p>Stuck Frequency : 74106 Best Objective : 58 Total Iteration : 412348 Duration Search : 649.611 ms</p>
State Awal	<p>INITIAL</p> <p>Objective Value: 739916</p> <p>FINAL</p> <p>Objective Value: 58</p> <p>PLOT</p> <p>Objective Value Graph</p>
Plot Objective Value terhadap Iterasi	

Plot $e^{\Delta E/T}$ terhadap iterasi



F. Genetic Algorithm

Percobaan 1 : population : 2 & iteration: 10

Objective Value dan Durasi

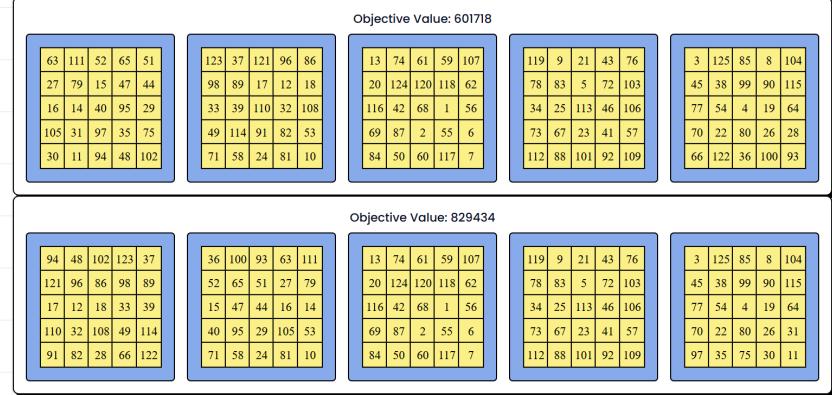
Genetic Algorithm

Best Objective : 601718
 Total Population : 2
 Total Iteration : 10
 Duration Search : 0 ms

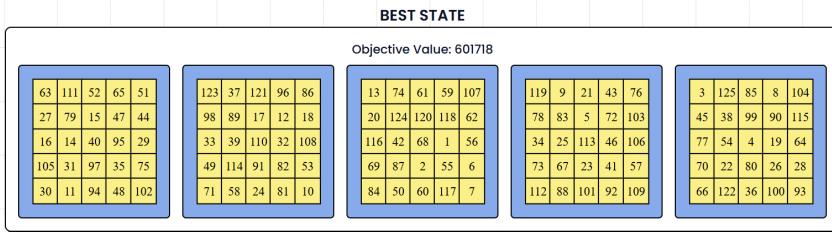
State Awal

Objective Value: 659207				
117 79 11 59 125	67 98 57 3 58	97 66 12 47 87	34 76 45 8 53	82 32 93 64 110
63 39 118 10 50	96 62 21 69 84	54 112 43 114 90	25 42 24 83 74	113 48 17 65 2
75 123 38 20 19	106 18 13 36 49	80 105 121 124 30	33 115 26 120 55	44 91 41 28 100
4 51 68 89 6	86 7 85 35 95	31 107 119 78 109	5 37 94 61 15	116 1 70 14 22
73 101 122 23 9	99 52 27 56 40	72 29 108 102 46	103 81 111 77 88	60 71 104 16 92
Objective Value: 670420				
94 48 102 123 37	36 100 93 63 111	13 74 61 59 107	22 80 26 31 97	103 34 25 113 46
121 96 86 98 89	52 65 51 27 79	20 124 120 118 62	35 75 30 11 55	106 73 67 23 41
17 12 18 33 39	15 47 44 16 14	116 42 68 1 56	6 84 50 60 117	57 112 88 101 92
110 32 108 49 114	40 95 29 105 53	69 87 2 115 77	7 119 9 21 43	109 3 125 85 8
91 82 28 66 122	71 58 24 81 10	54 4 19 64 70	76 78 83 5 72	104 45 38 99 90

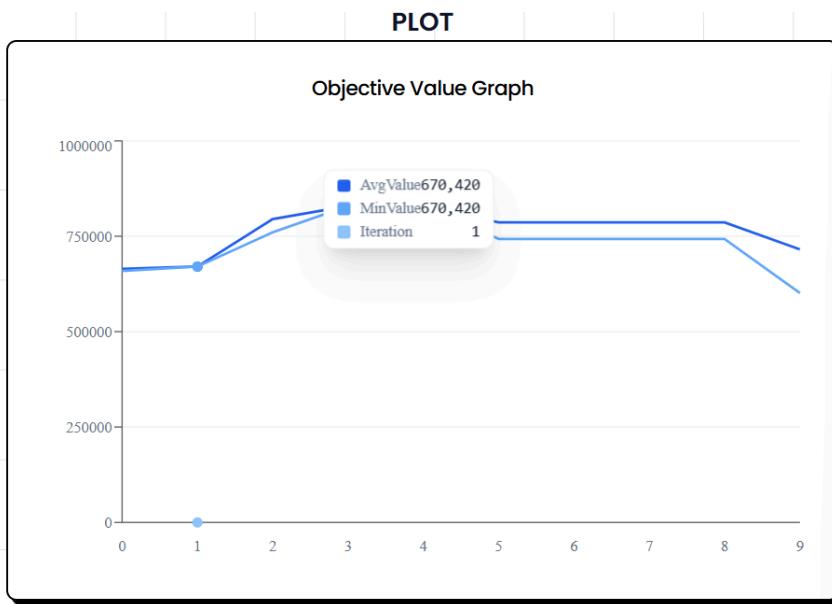
State Akhir



Best State



Plot Objective Value terhadap Iterasi



Percobaan 2 : population : 2 & iteration: 100

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 661511

Total Population : 2

Total Iteration : 100

Duration Search : 0.8 ms

State Awal

Objective Value: 672800									
35	15	64	108	9	121	95	100	82	1
13	54	29	20	23	12	61	83	45	84
124	101	30	63	62	103	34	97	98	50
76	16	88	94	53	7	58	68	122	18
79	41	52	38	59	87	109	85	60	36

Objective Value: 786116									
120	51	70	7	23	96	93	22	40	92
67	109	62	95	121	33	31	106	107	17
124	101	68	115	32	57	119	61	45	81
80	117	6	53	113	116	71	50	27	47
125	91	87	111	14	4	19	98	123	12

State Akhir

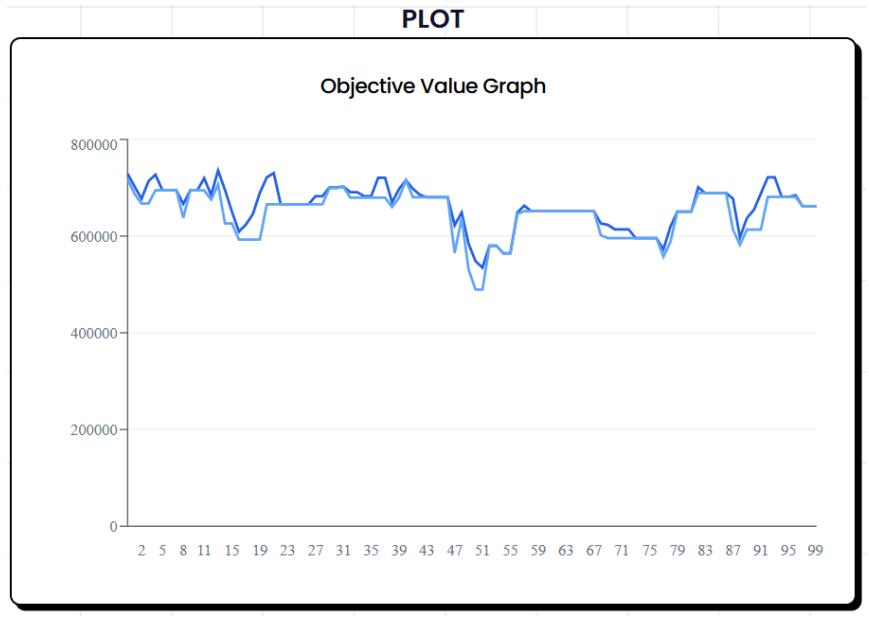
Objective Value: 661511									
59	101	117	52	36	22	41	61	38	79
77	82	107	98	50	1	31	37	34	97
99	91	4	108	62	40	92	33	123	120
8	71	104	105	57	23	67	15	89	66
115	114	110	17	44	64	35	124	68	122

Objective Value: 661511									
59	101	117	52	36	22	41	61	38	79
77	82	107	98	50	1	31	37	34	97
99	91	4	108	62	40	92	33	123	120
8	71	104	105	57	23	67	15	89	66
115	114	110	17	44	64	35	124	68	122

Best State

BEST STATE									
Objective Value: 661511									
59	101	117	52	36	22	41	61	38	79
77	82	107	98	50	1	31	37	34	97
99	91	4	108	62	40	92	33	123	120
8	71	104	105	57	23	67	15	89	66
115	114	110	17	44	64	35	124	68	122

Plot Objective Value terhadap Iterasi



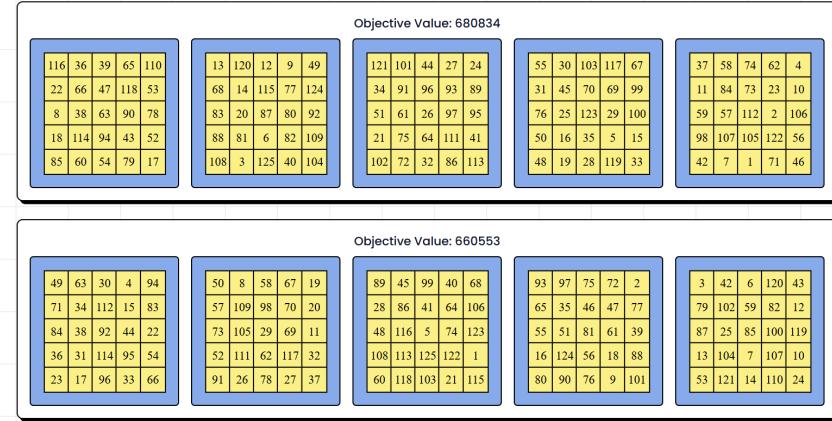
Percobaan 3 : population : 2 & iteration: 1000

Objective Value dan Durasi

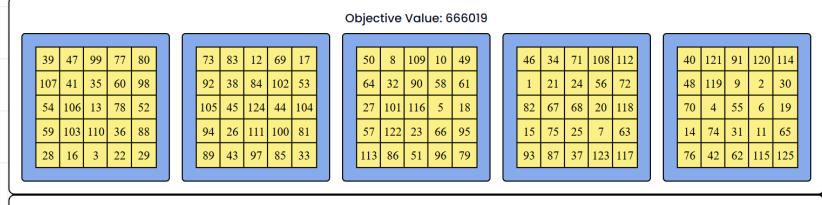
Genetic Algorithm

Best Objective : 666019
Total Population : 2
Total Iteration : 1000
Duration Search : 6.355 ms

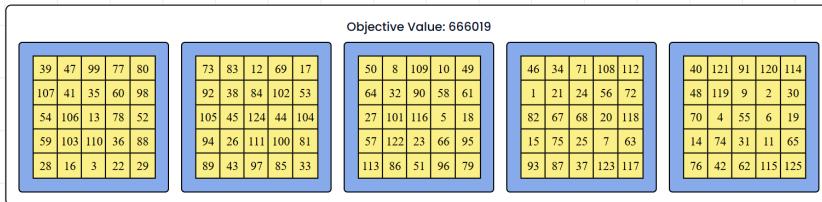
State Awal



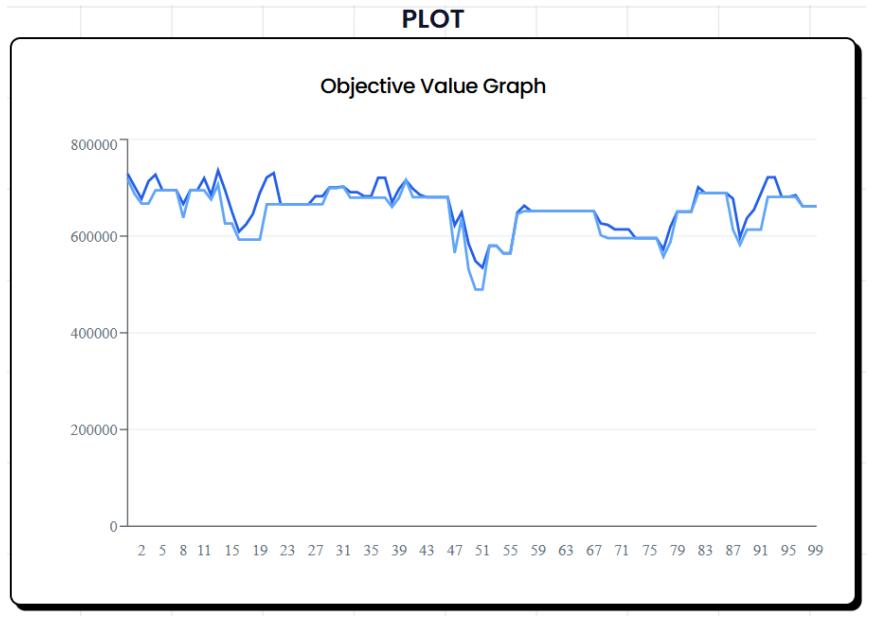
State Akhir



Best State



Plot Objective Value terhadap Iterasi



Percobaan 4: population 3 & iteration 10

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 704699

Total Population : 3

Total Iteration : 10

Duration Search : 0 ms

State Awal

Objective Value: 792237											
48	47	8	32	86							
85	26	90	17	83							
94	45	19	106	38							
56	53	87	120	51							
13	89	25	15	124							
108	59	71	114	103							
29	123	60	118	12							
125	113	91	102	64							
95	46	72	79	75							
14	33	116	41	101							
122	36	37	23	70							
39	78	61	76	57							
44	82	111	74	62							
105	30	34	52	7							
112	27	107	24	117							
104	96	16	6	92							
81	115	55	5	73							
4	88	35	1	69							
93	84	43	54	22							
100	28	68	77	9							
63	99	31	2	18							
65	109	98	58	21							
11	119	20	49	10							
40	50	3	66	80							
67	97	42	121	110							

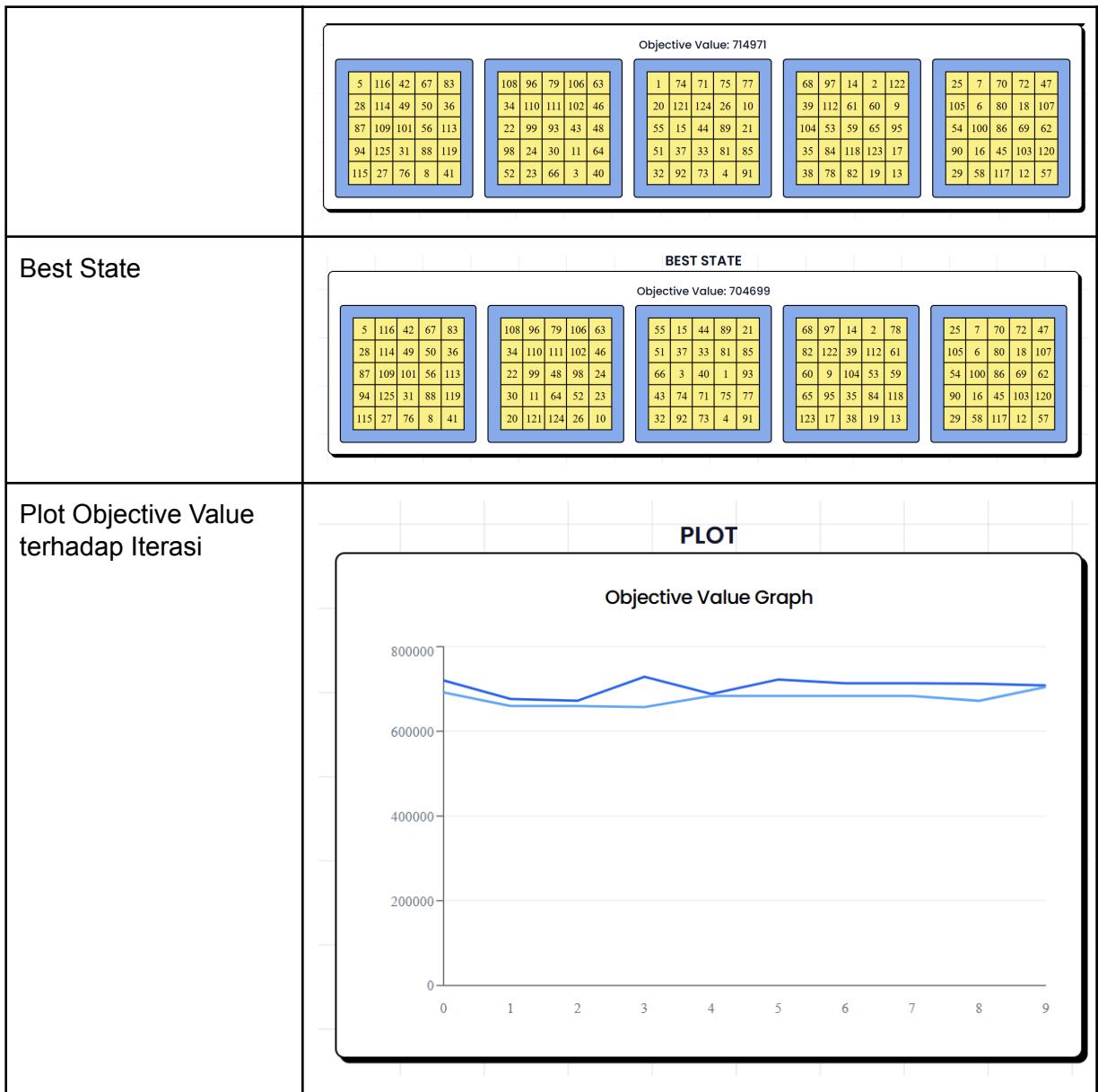
Objective Value: 701895											
114	107	2	123	95							
91	66	78	84	22							
93	68	52	105	104							
74	65	1	19	51							
47	113	8	82	115							
99	97	70	96	21							
36	34	71	76	54							
109	100	33	110	73							
40	37	98	119	32							
101	87	49	46	89							
9	13	121	43	31							
118	53	59	86	62							
61	90	38	4	16							
25	45	35	92	103							
120	29	6	58	117							
12	57	5	116	42							
67	83	28	102	11							
77	81	79	94	106							
18	14	7	112	48							
125	124	30	72	56							
88	26	122	41	63							
75	3	23	64	55							
44	10	69	80	108							
85	27	39	17	60							
20	24	111	15	50							

Objective Value: 772283											
116	67	96	79	106							
63	62	34	110	111							
102	12	46	22	99							
48	24	30	11	64							
52	5	103	23	20							
121	124	83	26	10							
55	15	44	89	21							
51	37	120	33	58							
81	85	29	66	57							
117	3	40	1	16							
93	43	74	71	90							
65	95	35	84	118							
123	17	6	38	78							
82	19	13	25	7							
70	72	47	105	80							
18	107	54	100	86							
69	98	114	49	75							
77	32	92	73	4							
91	68	97	14	2							
122	39	112	50	53							
36	28	87	109	101							
56	45	113	94	125							
59	61	60	9	104							
31	88	119	115	27							
76	42	8	41	108							

State Akhir

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

Objective Value: 704699											
5	116	42	67	83							
28	114	49	50	36							
87	109	101	56	113							
94	125	31	88	119							
115	27	76	8	41							
108	96	79	106	63							
34	110	111	102	46							
22	99	48	98	24							
30	11	64	52	23							
20	121	124	26	10							
55	15	44	89	21							
51	37	33	81	85							
66	3	40	1	93							
43	74	71	75	77							
32	92	73	4	91							
68	97	14	2	78							
82	122	39	112	61							
60	9	104	53	59							
65	95	35	84	118							
123	17	38	19	13							
25	7	70	72	47							
105	6	80	18	107							
54	100	86	69	62							
90	16	45	103	120							
29	58	117	12	57							



Percobaan 5: population 3 & iteration 100

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 621474

Total Population : 3

Total Iteration : 100

Duration Search : 0.581 ms

State Awal

Initial Population

Objective Value: 651742									
111	100	3	52	79					
62	28	8	102	81					
113	89	82	77	68					
33	76	64	45	66					
26	25	67	1	53					
97	12	48	65	87					
86	23	124	59	107					
6	84	49	123	69					
30	78	29	70	98					
37	35	96	51	105					
115	112	43	73	104					
71	38	40	114	9					
85	92	5	60	17					
83	90	116	108	24					
103	75	72	39	54					
36	15	41	110	121					
31	122	80	63	22					
18	109	95	2	42					
7	32	119	14	91					
88	101	61	13	47					
19	74	125	117	16					
4	120	93	46	94					
21	118	56	99	34					
27	106	57	55	58					
10	50	11	20	44					

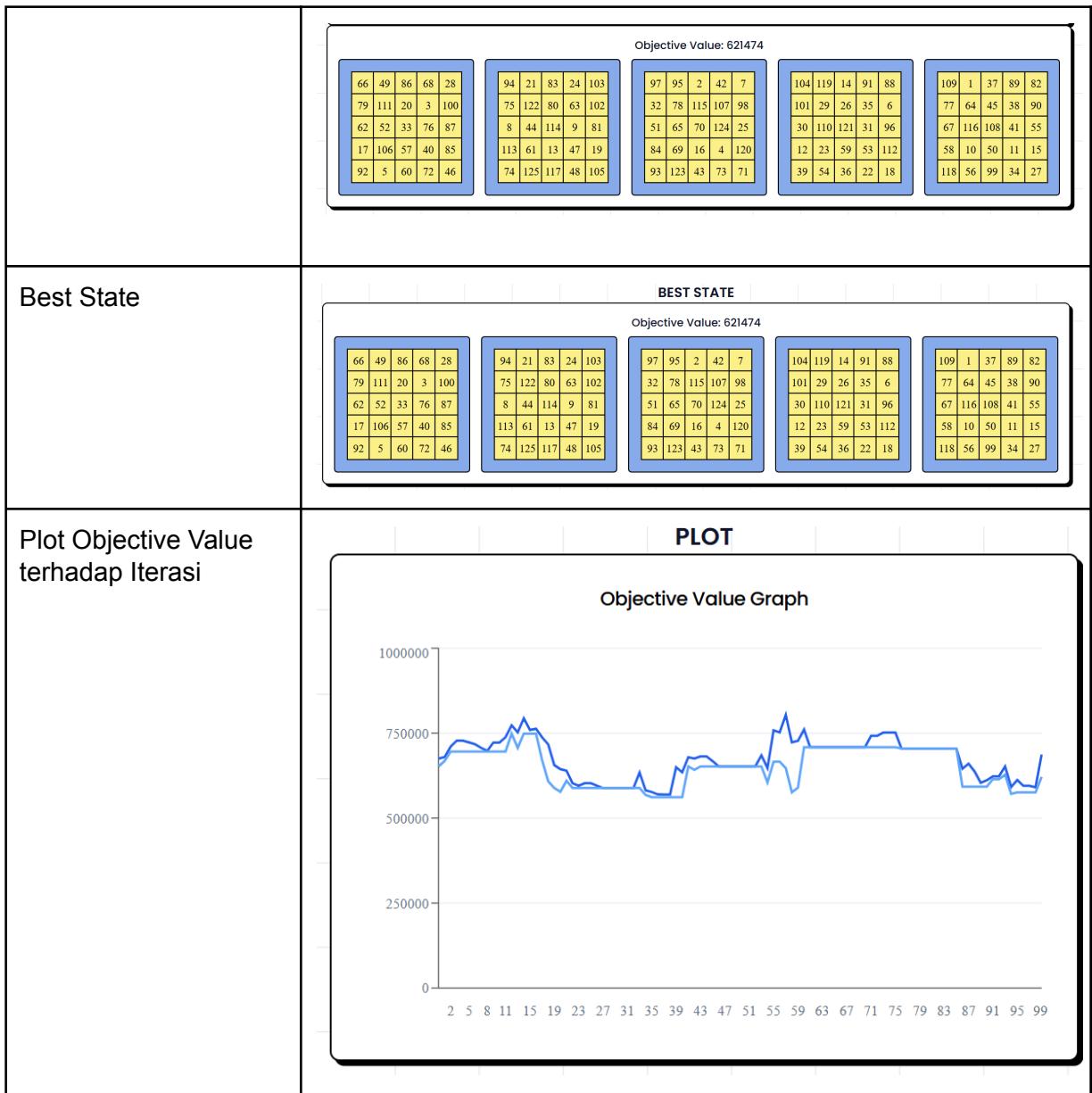
Objective Value: 648734									
118	73	69	92	93					
111	40	26	15	103					
113	119	12	87	14					
32	31	16	78	27					
106	42	122	23	54					
76	39	75	90	20					
53	110	25	5	51					
125	3	2	114	36					
22	9	109	59	46					
29	97	82	30	56					
80	6	100	11	117					
34	71	45	99	4					
38	88	81	62	115					
58	85	60	64	52					
66	89	47	96	124					
70	94	108	77	24					
102	107	35	98	55					
33	19	57	8	7					
72	44	104	43	95					
49	120	101	28	41					
74	91	105	112	123					
121	10	84	68	65					
67	116	63	37	83					
18	79	48	21	86					
61	13	17	50	1					

Objective Value: 674011									
52	49	62	11	47					
15	50	86	110	109					
45	95	60	73	38					
87	1	125	72	37					
67	108	94	63	58					
103	89	5	118	88					
78	20	76	115	44					
8	85	71	123	93					
9	22	48	119	105					
27	97	29	43	80					
41	3	26	92	99					
14	40	35	21	106					
46	6	2	64	54					
30	57	96	12	66					
23	82	16	91	117					
101	90	74	68	39					
59	53	112	19	107					
98	51	121	65	122					
34	79	111	13	81					
114	61	4	102	28					
120	10	70	100	56					
55	124	83	25	31					
32	17	113	75	42					
84	33	36	24	77					
104	7	116	18	69					

State Akhir

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

Objective Value: 731643									
79	111	20	3	100					
62	52	33	76	87					
17	106	57	40	85					
92	5	60	72	46					
94	21	83	24	103					
31	96	12	23	59					
75	122	80	63	61					
13	47	19	74	125					
117	48	105	97	95					
2	42	7	32	78					
115	107	98	51	65					
70	124	25	84	69					
16	4	120	93	123					
43	73	71	104	119					
14	91	88	101	29					
26	35	6	30	110					
121	53	112	39	54					
36	22	18	109	1					
37	102	8	44	114					
9	81	113	89	82					
77	64	45	38	90					
67	116	108	41	55					
58	10	50	11	15					
118	56	99	34	27					
66	49	86	68	28					



Percobaan 6: population 3 & iteration 1000

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 674749

Total Population : 3

Total Iteration : 1000

Duration Search : 7.601 ms

State Awal

Initial Population

Objective Value: 700514									
73	70	23	45	6					
41	24	119	100	1					
21	7	12	58	92					
13	68	86	4	62					
22	52	34	29	54					
69	95	17	15	33					
82	120	31	14	104					
35	67	78	2	117					
5	99	115	106	75					
88	20	65	83	26					
91	47	37	49	50					
107	125	30	61	38					
59	46	101	123	43					
118	48	72	108	97					
8	110	57	9	87					
74	77	27	53	28					
102	66	25	39	42					
16	40	113	36	93					
122	51	105	85	11					
103	19	114	112	60					
55	79	10	76	44					
71	124	96	3	18					
121	109	63	94	84					
90	116	56	98	32					
64	80	111	89	81					

Objective Value: 828496

Objective Value: 828496									
100	90	1	12	64					
13	125	54	77	84					
55	75	88	76	108					
38	24	9	59	103					
2	104	93	69	99					
34	51	113	81	41					
52	83	26	109	66					
79	98	71	19	73					
3	5	20	121	74					
68	47	67	82	39					
25	112	97	23	115					
7	94	117	57	11					
53	120	37	16	31					
85	119	14	40	96					
42	72	6	46	35					
49	114	102	91	118					
110	124	28	58	62					
65	18	70	15	50					
101	32	33	111	60					
105	10	92	8	123					
116	21	95	122	63					
45	107	56	80	4					
86	43	89	106	17					
29	78	44	87	27					
48	30	22	36	61					

Objective Value: 799367

Objective Value: 799367									
56	54	8	61	23					
65	77	55	53	86					
118	18	109	66	40					
12	14	32	43	26					
124	69	90	123	48					
119	24	49	88	52					
84	78	76	2	4					
92	7	96	62	75					
59	91	115	120	79					
50	93	6	17	39					
44	81	101	3	72					
35	85	27	71	29					
121	63	116	10	16					
36	87	104	57	112					
15	28	21	60	68					
58	99	41	1	9					
38	67	110	105	125					
114	45	94	122	20					
111	89	102	113	98					
103	70	64	19	51					
73	97	108	42	80					
25	74	11	34	107					
33	22	106	82	47					
13	31	117	83	5					
37	100	46	95	30					

State Akhir

Final Population

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

Objective Value: 674749

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

	<p style="text-align: center;">Objective Value: 674749</p> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>42</td><td>52</td><td>27</td><td>115</td><td>71</td></tr> <tr><td>113</td><td>39</td><td>1</td><td>100</td><td>89</td></tr> <tr><td>90</td><td>84</td><td>104</td><td>63</td><td>83</td></tr> <tr><td>7</td><td>45</td><td>6</td><td>123</td><td>106</td></tr> <tr><td>5</td><td>24</td><td>37</td><td>119</td><td>105</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>109</td><td>38</td><td>32</td><td>64</td><td>72</td></tr> <tr><td>67</td><td>17</td><td>99</td><td>122</td><td>117</td></tr> <tr><td>120</td><td>55</td><td>11</td><td>36</td><td>46</td></tr> <tr><td>111</td><td>53</td><td>56</td><td>44</td><td>3</td></tr> <tr><td>47</td><td>75</td><td>16</td><td>60</td><td>61</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>21</td><td>96</td><td>15</td><td>125</td><td>74</td></tr> <tr><td>78</td><td>101</td><td>121</td><td>95</td><td>12</td></tr> <tr><td>68</td><td>66</td><td>93</td><td>26</td><td>28</td></tr> <tr><td>31</td><td>49</td><td>14</td><td>92</td><td>98</td></tr> <tr><td>65</td><td>51</td><td>30</td><td>76</td><td>22</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>124</td><td>97</td><td>91</td><td>59</td><td>103</td></tr> <tr><td>40</td><td>102</td><td>79</td><td>82</td><td>54</td></tr> <tr><td>114</td><td>48</td><td>110</td><td>29</td><td>118</td></tr> <tr><td>57</td><td>77</td><td>107</td><td>8</td><td>20</td></tr> <tr><td>34</td><td>86</td><td>9</td><td>87</td><td>88</td></tr> </table> <table border="1" style="display: inline-table;"> <tr><td>2</td><td>25</td><td>33</td><td>43</td><td>81</td></tr> <tr><td>18</td><td>108</td><td>62</td><td>35</td><td>23</td></tr> <tr><td>4</td><td>85</td><td>50</td><td>70</td><td>73</td></tr> <tr><td>10</td><td>116</td><td>19</td><td>41</td><td>69</td></tr> <tr><td>94</td><td>112</td><td>13</td><td>58</td><td>80</td></tr> </table>	42	52	27	115	71	113	39	1	100	89	90	84	104	63	83	7	45	6	123	106	5	24	37	119	105	109	38	32	64	72	67	17	99	122	117	120	55	11	36	46	111	53	56	44	3	47	75	16	60	61	21	96	15	125	74	78	101	121	95	12	68	66	93	26	28	31	49	14	92	98	65	51	30	76	22	124	97	91	59	103	40	102	79	82	54	114	48	110	29	118	57	77	107	8	20	34	86	9	87	88	2	25	33	43	81	18	108	62	35	23	4	85	50	70	73	10	116	19	41	69	94	112	13	58	80
42	52	27	115	71																																																																																																																										
113	39	1	100	89																																																																																																																										
90	84	104	63	83																																																																																																																										
7	45	6	123	106																																																																																																																										
5	24	37	119	105																																																																																																																										
109	38	32	64	72																																																																																																																										
67	17	99	122	117																																																																																																																										
120	55	11	36	46																																																																																																																										
111	53	56	44	3																																																																																																																										
47	75	16	60	61																																																																																																																										
21	96	15	125	74																																																																																																																										
78	101	121	95	12																																																																																																																										
68	66	93	26	28																																																																																																																										
31	49	14	92	98																																																																																																																										
65	51	30	76	22																																																																																																																										
124	97	91	59	103																																																																																																																										
40	102	79	82	54																																																																																																																										
114	48	110	29	118																																																																																																																										
57	77	107	8	20																																																																																																																										
34	86	9	87	88																																																																																																																										
2	25	33	43	81																																																																																																																										
18	108	62	35	23																																																																																																																										
4	85	50	70	73																																																																																																																										
10	116	19	41	69																																																																																																																										
94	112	13	58	80																																																																																																																										
Best State	<p style="text-align: center;">BEST STATE</p> <p style="text-align: center;">Objective Value: 674749</p> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>85</td><td>50</td><td>70</td><td>73</td><td>10</td></tr> <tr><td>116</td><td>19</td><td>41</td><td>69</td><td>94</td></tr> <tr><td>112</td><td>13</td><td>58</td><td>80</td><td>42</td></tr> <tr><td>7</td><td>45</td><td>6</td><td>123</td><td>106</td></tr> <tr><td>5</td><td>24</td><td>37</td><td>119</td><td>105</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>109</td><td>38</td><td>32</td><td>64</td><td>72</td></tr> <tr><td>67</td><td>17</td><td>99</td><td>122</td><td>117</td></tr> <tr><td>120</td><td>55</td><td>11</td><td>36</td><td>46</td></tr> <tr><td>111</td><td>53</td><td>56</td><td>44</td><td>3</td></tr> <tr><td>47</td><td>75</td><td>16</td><td>60</td><td>61</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>21</td><td>96</td><td>15</td><td>125</td><td>74</td></tr> <tr><td>78</td><td>101</td><td>121</td><td>95</td><td>12</td></tr> <tr><td>68</td><td>66</td><td>93</td><td>26</td><td>28</td></tr> <tr><td>31</td><td>49</td><td>14</td><td>92</td><td>98</td></tr> <tr><td>65</td><td>51</td><td>30</td><td>76</td><td>22</td></tr> </table> <table border="1" style="display: inline-table; margin-right: 10px;"> <tr><td>124</td><td>97</td><td>91</td><td>59</td><td>103</td></tr> <tr><td>40</td><td>102</td><td>79</td><td>82</td><td>54</td></tr> <tr><td>114</td><td>48</td><td>110</td><td>29</td><td>118</td></tr> <tr><td>57</td><td>77</td><td>107</td><td>8</td><td>20</td></tr> <tr><td>34</td><td>86</td><td>9</td><td>87</td><td>88</td></tr> </table> <table border="1" style="display: inline-table;"> <tr><td>2</td><td>25</td><td>33</td><td>43</td><td>81</td></tr> <tr><td>18</td><td>108</td><td>62</td><td>35</td><td>23</td></tr> <tr><td>52</td><td>27</td><td>115</td><td>71</td><td>113</td></tr> <tr><td>39</td><td>1</td><td>100</td><td>89</td><td>90</td></tr> <tr><td>84</td><td>104</td><td>63</td><td>83</td><td>4</td></tr> </table>	85	50	70	73	10	116	19	41	69	94	112	13	58	80	42	7	45	6	123	106	5	24	37	119	105	109	38	32	64	72	67	17	99	122	117	120	55	11	36	46	111	53	56	44	3	47	75	16	60	61	21	96	15	125	74	78	101	121	95	12	68	66	93	26	28	31	49	14	92	98	65	51	30	76	22	124	97	91	59	103	40	102	79	82	54	114	48	110	29	118	57	77	107	8	20	34	86	9	87	88	2	25	33	43	81	18	108	62	35	23	52	27	115	71	113	39	1	100	89	90	84	104	63	83	4
85	50	70	73	10																																																																																																																										
116	19	41	69	94																																																																																																																										
112	13	58	80	42																																																																																																																										
7	45	6	123	106																																																																																																																										
5	24	37	119	105																																																																																																																										
109	38	32	64	72																																																																																																																										
67	17	99	122	117																																																																																																																										
120	55	11	36	46																																																																																																																										
111	53	56	44	3																																																																																																																										
47	75	16	60	61																																																																																																																										
21	96	15	125	74																																																																																																																										
78	101	121	95	12																																																																																																																										
68	66	93	26	28																																																																																																																										
31	49	14	92	98																																																																																																																										
65	51	30	76	22																																																																																																																										
124	97	91	59	103																																																																																																																										
40	102	79	82	54																																																																																																																										
114	48	110	29	118																																																																																																																										
57	77	107	8	20																																																																																																																										
34	86	9	87	88																																																																																																																										
2	25	33	43	81																																																																																																																										
18	108	62	35	23																																																																																																																										
52	27	115	71	113																																																																																																																										
39	1	100	89	90																																																																																																																										
84	104	63	83	4																																																																																																																										
Plot Objective Value terhadap Iterasi	<p style="text-align: center;">PLOT</p> <p style="text-align: center;">Objective Value Graph</p> <p>The graph illustrates the search space of the Genetic Algorithm. The objective value starts around 750,000 and fluctuates significantly as the algorithm explores different solutions across 1000 iterations.</p>																																																																																																																													

Percobaan 7: population 4 & iteration 10

Objective Value dan Durasi	<p style="text-align: center;">Genetic Algorithm</p> <p>Best Objective : 615157</p> <p>Total Population : 4</p> <p>Total Iteration : 10</p> <p>Duration Search : 0.182 ms</p>
----------------------------	--

State Awal

Initial Population

Objective Value: 674124

17	95	49	41	58
14	16	2	92	118
57	114	116	53	46
66	94	23	15	119
27	109	67	106	88

111	61	68	45	86
93	24	84	47	97
11	113	99	35	59
100	30	71	36	81
13	51	48	32	18

37	101	1	96	72
90	25	120	82	122
21	125	33	98	85
87	115	80	55	77
108	29	112	65	124

62	10	78	123	22
76	6	60	40	28
63	9	19	50	7
69	105	39	107	75
64	74	20	42	103

117	91	4	43	34
70	56	83	110	38
12	73	121	31	3
54	104	26	102	8
89	5	52	79	44

6	75	56	111	34
93	15	104	14	62
21	29	27	116	26
57	24	97	3	41
45	79	72	100	5

32	120	124	53	51
16	48	84	18	112
40	55	103	35	36
105	85	99	22	67
50	39	107	33	80

10	23	4	17	1
25	78	59	91	117
113	61	12	31	49
19	64	123	58	65
92	9	44	125	114

95	77	11	119	46
63	47	81	101	30
60	13	93	20	89
37	106	52	88	54
28	90	76	109	121

42	70	118	71	68
115	110	74	43	8
87	108	7	66	38
96	69	2	102	94
82	98	122	86	73

49	125	117	64	98
71	97	52	75	110
17	5	80	29	34
37	84	118	122	89
3	59	24	51	101

65	103	33	30	58
46	50	9	70	92
55	120	7	62	18
74	12	60	123	124
113	53	61	102	104

4	16	28	82	23
38	48	43	91	96
94	31	112	13	121
32	107	15	68	45
57	67	39	11	100

108	21	54	2	27
44	73	42	116	86
79	36	6	22	77
114	88	63	25	106
76	81	10	111	115

78	35	95	99	41
85	14	105	66	93
72	47	26	87	1
109	83	56	19	8
40	119	90	20	69

19	51	73	36	98
79	28	81	76	16
54	23	26	48	77
49	12	46	29	64
110	89	123	101	109

66	33	99	93	111
15	120	72	37	106
53	118	2	84	125
92	25	71	121	83
80	34	32	56	70

114	44	102	4	22
67	50	41	69	104
65	31	43	96	78
86	62	112	60	59
1	5	20	24	27

74	21	100	10	47
7	75	8	97	117
11	119	46	63	81
101	60	20	89	37
106	52	88	54	28

113	13	61	18	116
82	40	52	38	30
107	68	63	124	119
9	35	17	58	91
88	55	103	42	87

57	2	16	116	53
94	15	27	67	45
24	84	97	59	100
30	71	36	13	51
48	32	18	96	120

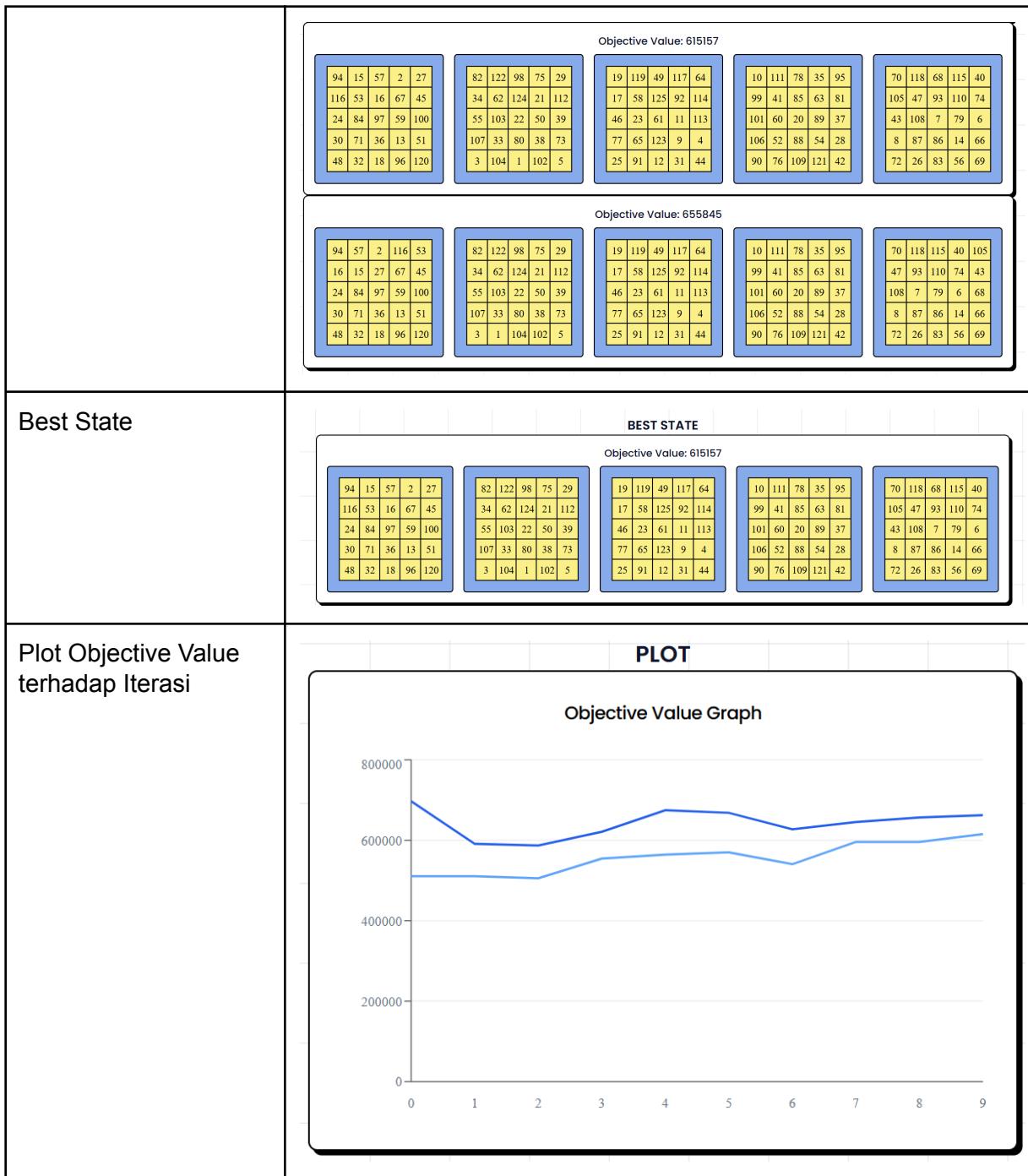
82	122	98	75	29
34	62	124	21	112
55	103	22	50	39
107	33	80	38	73
3	104	102	5	111

35	99	41	85	10
23	4	17	1	25
78	91	117	113	61
12	31	49	19	64
123	58	65	92	9

44	125	114	95	77
11	119	46	63	81
101	60	20	89	37
106	52	88	54	28
90	76	109	121	42

70	118	68	115	40

<tbl_r cells="5" ix="4" maxcspan="1" maxrspan



Percobaan 8: population 4 & iteration 100

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 627547

Total Population : 4

Total Iteration : 100

Duration Search : 1.021 ms

State Awal

Initial Population

Objective Value: 731764

52	44	115	120	2
71	98	5	88	3
82	77	51	123	101
70	87	104	61	46
58	9	16	83	84

64	110	107	62	117
49	20	11	17	100
74	103	66	109	93
23	76	29	89	86
96	68	28	8	34

33	27	37	59	113
108	55	24	54	38
56	69	122	119	90
39	32	118	48	43
12	18	50	81	31

111	125	57	1	41
47	60	85	99	114
75	92	21	40	80
26	4	65	94	116
42	15	13	72	105

22	10	35	25	14
124	102	106	19	63
95	73	97	6	121
91	79	78	30	36
67	7	112	53	45

49	65	2	45	77
105	61	109	76	21
25	64	70	55	54
9	87	111	125	74
104	110	51	5	113

124	29	75	100	41
86	115	16	94	3
47	27	107	43	71
89	50	1	68	91
13	82	7	32	40

11	114	28	42	119
20	30	92	17	24
102	117	103	95	15
73	44	22	26	120
48	83	106	39	10

14	58	108	101	96
4	116	36	121	79
18	81	31	99	72
123	69	34	80	85
19	97	33	62	12

53	67	84	122	59
52	60	23	66	112
63	6	57	93	8
88	46	78	35	38
56	98	118	90	37

125	34	107	74	53
108	103	83	23	29
48	50	62	115	45
54	112	13	18	77
82	78	6	19	121

114	63	79	120	40
69	111	4	8	33
101	58	95	41	109
49	97	57	110	21
68	99	46	106	39

16	84	24	7	5
124	70	116	30	9
92	43	56	36	66
113	28	117	42	76
25	90	122	105	89

64	17	72	32	3
87	104	38	44	86
71	22	37	75	52
96	67	35	20	59
55	123	98	15	47

31	85	118	119	11
100	12	73	94	65
27	93	102	14	81
10	60	80	26	2
1	91	51	88	61

25	75	105	7	8
124	62	125	98	19
66	4	35	71	118
76	60	103	11	65
116	90	49	79	94

38	3	68	73	123
40	1	30	58	55
104	113	101	93	26
112	42	15	88	77
122	100	16	41	107

48	6	52	74	43
2	36	61	56	89
45	32	9	117	39
10	83	24	99	23
108	63	114	110	5

64	106	51	80	18
119	50	81	84	87
47	20	27	95	102
29	46	33	44	67
86	21	22	121	91

78	54	12	53	85
82	37	72	14	69
59	120	96	115	57
109	34	70	28	111
92	17	31	97	13

State Akhir

Final Population

Objective Value: 759314

17	8	3	124	47
60	70	52	44	115
88	69	68	19	29
116	82	39	72	93
104	58	64	4	96

74	66	25	14	65
94	13	90	103	51
84	123	101	49	95
73	31	32	9	27
85	117	59	54	125

83	24	63	12	1
41	61	46	38	113
111	57	120	107	92
26	106	40	10	35
108	102	18	81	75

11	50	36	67	99
114	110	100	2	71
98	5	16	42	15
122	89	76	20	97
6	121	91	79	78

30	86	28	34	33
77	37	62	23	55
80	56	21	119	109
118	48	43	105	7
112	53	22	45	87

16	67	99	80	56
21	86	28	34	33
77	37	62	119	42
15	122	89	76	20
97	6	121	91	79

78	30	23	109	118
48	43	105	7	112
53	22	45	87	55
88	69	68	19	29
116	82	39	72	93

104	58	64	4	96
74	66	25	14	65
94	13	90	103	51
84	123	101	49	95
73	31	32	9	27

85	117	59	54	125
83	24	63	12	1
41	61	46	38	113
111	57	120	107	92
26	106	40	10	35

108	102	18	81	75
11	50	36	17	8
3	124	47	60	70
52	44	115	114	110
100	2	71	98	5

17	8	3	124	47
60	70	52	44	115
114	110	100	2	71
98	5	16	42	15
122	89	76	20	97

6	121	91	23	55
88	69	68	58	64
4	96	74	66	25
14	65	94	13	90
103	51	84	123	101

49	95	73	31	32
9	27	19	29	116
82	39	72	93	104
86	28	34	33	77
37	62	85	117	59

54	125	83	24	63
12	1	41	61	46
38	113	111	57	120
107	92	26	106	40
10	35	108	102	18

81	75	11	50	79
78	30	36	67	99
80	56	21	119	109
118	48	43	105	7
112	53	22	45	87

17	8	3	124	47
60	70	52	44	115
114	110	100	2	71
98	5	16	69	68
19	29	116	82	39

72	93	42	15	122
89	76	20	97	6
121	91	86	28	34
33	77	37	62	23
55	88	104	58	64

4	96	74	66	25
14	65	94	13	90
103	51	84	123	101
49	95	73	31	32
9	27	85	117	59

54	125	83	24	63
12	1	41	61	46
38	113	111	57	120
107	92	26	106	40
10	35	108	102	18

81	75	11	50	79
78	30	36	67	99
80	56	21	119	109
118	48	43	105	7
112	53	22	45	87

Best State

BEST STATE

Objective Value: 627547

17	8	3	124	47
60	70	52	44	115
114	110	100	2	71
98	5	16	42	15
122	89	76	20	97

6	121	91	23	55
88	69	68	58	64
4	96	74	66	25
14	65	94	13	90
103	51	84	123	101

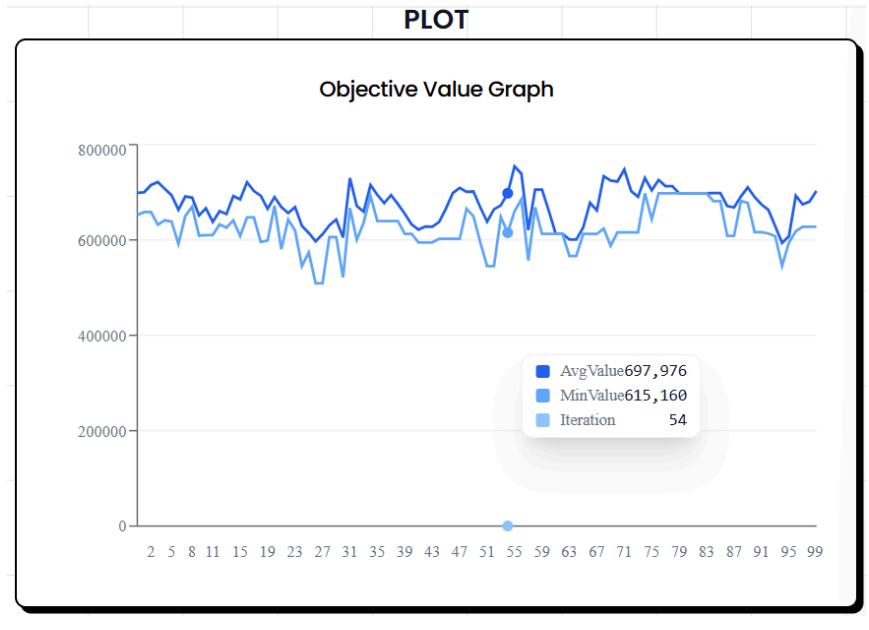
49	95	73	31	32
9	27	19	29	116
82	39	72	93	104
86	28	34	33	77
37	62	85	117	59

54	125	83	24	63
12	1	41	61	46
38	113	111	57	120
107	92	26	106	40
10	35	108	102	18

81	75	11	50	79

<tbl_r cells="

Plot Objective Value terhadap Iterasi



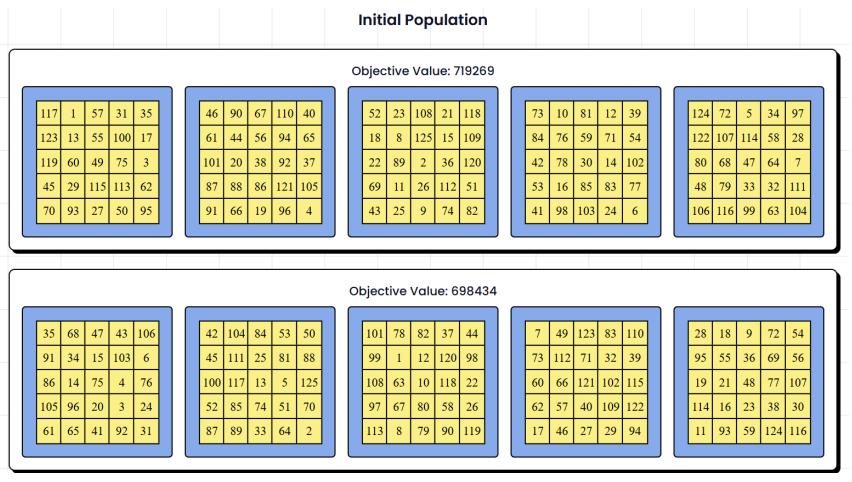
Percobaan 9: population 4 & iteration 1000

Objective Value dan Durasi

Genetic Algorithm

Best Objective : 556801
Total Population : 4
Total Iteration : 1000
Duration Search : 10.975 ms

State Awal



	<p style="text-align: center;">Objective Value: 678371</p> <table border="1" style="margin: auto;"> <tr><td>51</td><td>37</td><td>50</td><td>85</td><td>73</td></tr> <tr><td>26</td><td>115</td><td>31</td><td>32</td><td>18</td></tr> <tr><td>62</td><td>76</td><td>47</td><td>29</td><td>11</td></tr> <tr><td>112</td><td>103</td><td>54</td><td>45</td><td>27</td></tr> <tr><td>101</td><td>16</td><td>108</td><td>121</td><td>107</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>84</td><td>67</td><td>93</td><td>109</td><td>23</td></tr> <tr><td>104</td><td>125</td><td>10</td><td>91</td><td>68</td></tr> <tr><td>61</td><td>78</td><td>105</td><td>43</td><td>90</td></tr> <tr><td>52</td><td>100</td><td>14</td><td>38</td><td>69</td></tr> <tr><td>122</td><td>118</td><td>81</td><td>33</td><td>7</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>24</td><td>40</td><td>44</td><td>19</td><td>3</td></tr> <tr><td>48</td><td>123</td><td>9</td><td>89</td><td>102</td></tr> <tr><td>60</td><td>72</td><td>79</td><td>63</td><td>116</td></tr> <tr><td>15</td><td>4</td><td>17</td><td>124</td><td>56</td></tr> <tr><td>75</td><td>70</td><td>5</td><td>66</td><td>12</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>65</td><td>82</td><td>86</td><td>39</td><td>59</td></tr> <tr><td>64</td><td>77</td><td>88</td><td>96</td><td>111</td></tr> <tr><td>120</td><td>58</td><td>55</td><td>41</td><td>94</td></tr> <tr><td>6</td><td>21</td><td>49</td><td>113</td><td>57</td></tr> <tr><td>80</td><td>92</td><td>34</td><td>97</td><td>95</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>46</td><td>36</td><td>28</td><td>117</td><td>8</td></tr> <tr><td>110</td><td>42</td><td>22</td><td>119</td><td>1</td></tr> <tr><td>98</td><td>87</td><td>2</td><td>99</td><td>106</td></tr> <tr><td>74</td><td>53</td><td>114</td><td>13</td><td>25</td></tr> <tr><td>30</td><td>71</td><td>35</td><td>83</td><td>20</td></tr> </table>	51	37	50	85	73	26	115	31	32	18	62	76	47	29	11	112	103	54	45	27	101	16	108	121	107	84	67	93	109	23	104	125	10	91	68	61	78	105	43	90	52	100	14	38	69	122	118	81	33	7	24	40	44	19	3	48	123	9	89	102	60	72	79	63	116	15	4	17	124	56	75	70	5	66	12	65	82	86	39	59	64	77	88	96	111	120	58	55	41	94	6	21	49	113	57	80	92	34	97	95	46	36	28	117	8	110	42	22	119	1	98	87	2	99	106	74	53	114	13	25	30	71	35	83	20
51	37	50	85	73																																																																																																																										
26	115	31	32	18																																																																																																																										
62	76	47	29	11																																																																																																																										
112	103	54	45	27																																																																																																																										
101	16	108	121	107																																																																																																																										
84	67	93	109	23																																																																																																																										
104	125	10	91	68																																																																																																																										
61	78	105	43	90																																																																																																																										
52	100	14	38	69																																																																																																																										
122	118	81	33	7																																																																																																																										
24	40	44	19	3																																																																																																																										
48	123	9	89	102																																																																																																																										
60	72	79	63	116																																																																																																																										
15	4	17	124	56																																																																																																																										
75	70	5	66	12																																																																																																																										
65	82	86	39	59																																																																																																																										
64	77	88	96	111																																																																																																																										
120	58	55	41	94																																																																																																																										
6	21	49	113	57																																																																																																																										
80	92	34	97	95																																																																																																																										
46	36	28	117	8																																																																																																																										
110	42	22	119	1																																																																																																																										
98	87	2	99	106																																																																																																																										
74	53	114	13	25																																																																																																																										
30	71	35	83	20																																																																																																																										
	<p style="text-align: center;">Objective Value: 620316</p> <table border="1" style="margin: auto;"> <tr><td>26</td><td>47</td><td>66</td><td>114</td><td>81</td></tr> <tr><td>41</td><td>65</td><td>31</td><td>32</td><td>120</td></tr> <tr><td>84</td><td>43</td><td>97</td><td>72</td><td>119</td></tr> <tr><td>106</td><td>77</td><td>122</td><td>62</td><td>102</td></tr> <tr><td>75</td><td>57</td><td>85</td><td>4</td><td>2</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>10</td><td>115</td><td>53</td><td>46</td><td>118</td></tr> <tr><td>15</td><td>29</td><td>56</td><td>27</td><td>28</td></tr> <tr><td>107</td><td>104</td><td>95</td><td>70</td><td>87</td></tr> <tr><td>93</td><td>22</td><td>9</td><td>113</td><td>99</td></tr> <tr><td>58</td><td>101</td><td>116</td><td>82</td><td>40</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>83</td><td>96</td><td>80</td><td>92</td><td>74</td></tr> <tr><td>24</td><td>125</td><td>55</td><td>30</td><td>6</td></tr> <tr><td>35</td><td>18</td><td>89</td><td>71</td><td>112</td></tr> <tr><td>52</td><td>88</td><td>45</td><td>69</td><td>36</td></tr> <tr><td>50</td><td>59</td><td>11</td><td>73</td><td>23</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>17</td><td>44</td><td>63</td><td>124</td><td>42</td></tr> <tr><td>123</td><td>60</td><td>16</td><td>67</td><td>7</td></tr> <tr><td>64</td><td>25</td><td>68</td><td>21</td><td>5</td></tr> <tr><td>33</td><td>100</td><td>105</td><td>117</td><td>94</td></tr> <tr><td>103</td><td>61</td><td>91</td><td>1</td><td>110</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>48</td><td>38</td><td>121</td><td>79</td><td>76</td></tr> <tr><td>8</td><td>78</td><td>109</td><td>19</td><td>86</td></tr> <tr><td>12</td><td>54</td><td>49</td><td>13</td><td>14</td></tr> <tr><td>111</td><td>3</td><td>20</td><td>37</td><td>98</td></tr> <tr><td>51</td><td>90</td><td>108</td><td>39</td><td>34</td></tr> </table>	26	47	66	114	81	41	65	31	32	120	84	43	97	72	119	106	77	122	62	102	75	57	85	4	2	10	115	53	46	118	15	29	56	27	28	107	104	95	70	87	93	22	9	113	99	58	101	116	82	40	83	96	80	92	74	24	125	55	30	6	35	18	89	71	112	52	88	45	69	36	50	59	11	73	23	17	44	63	124	42	123	60	16	67	7	64	25	68	21	5	33	100	105	117	94	103	61	91	1	110	48	38	121	79	76	8	78	109	19	86	12	54	49	13	14	111	3	20	37	98	51	90	108	39	34
26	47	66	114	81																																																																																																																										
41	65	31	32	120																																																																																																																										
84	43	97	72	119																																																																																																																										
106	77	122	62	102																																																																																																																										
75	57	85	4	2																																																																																																																										
10	115	53	46	118																																																																																																																										
15	29	56	27	28																																																																																																																										
107	104	95	70	87																																																																																																																										
93	22	9	113	99																																																																																																																										
58	101	116	82	40																																																																																																																										
83	96	80	92	74																																																																																																																										
24	125	55	30	6																																																																																																																										
35	18	89	71	112																																																																																																																										
52	88	45	69	36																																																																																																																										
50	59	11	73	23																																																																																																																										
17	44	63	124	42																																																																																																																										
123	60	16	67	7																																																																																																																										
64	25	68	21	5																																																																																																																										
33	100	105	117	94																																																																																																																										
103	61	91	1	110																																																																																																																										
48	38	121	79	76																																																																																																																										
8	78	109	19	86																																																																																																																										
12	54	49	13	14																																																																																																																										
111	3	20	37	98																																																																																																																										
51	90	108	39	34																																																																																																																										

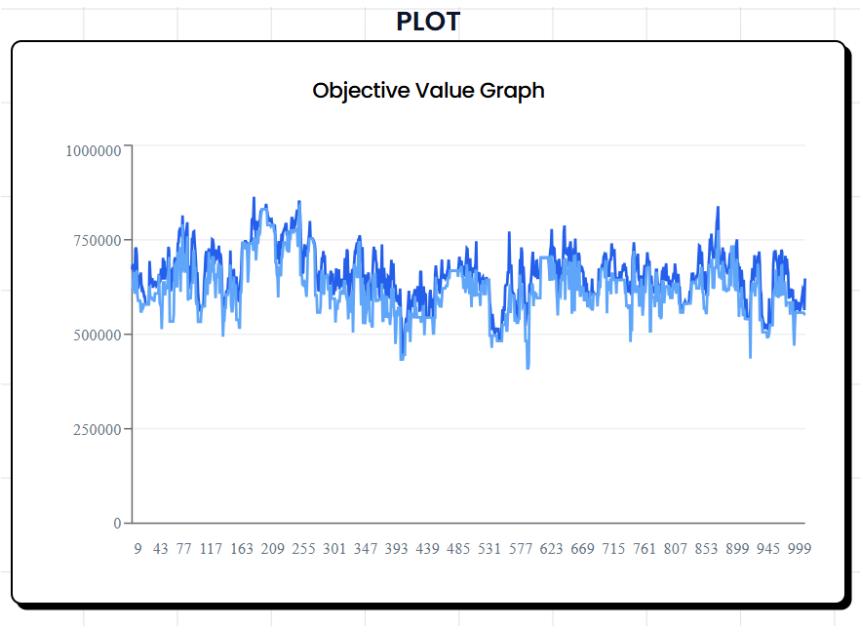
State Akhir

	<p style="text-align: center;">Final Population</p> <p style="text-align: center;">Objective Value: 556801</p> <table border="1" style="margin: auto;"> <tr><td>95</td><td>8</td><td>78</td><td>109</td><td>73</td></tr> <tr><td>116</td><td>53</td><td>34</td><td>31</td><td>122</td></tr> <tr><td>87</td><td>91</td><td>123</td><td>10</td><td>19</td></tr> <tr><td>22</td><td>84</td><td>57</td><td>26</td><td>62</td></tr> <tr><td>93</td><td>17</td><td>54</td><td>49</td><td>28</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>45</td><td>107</td><td>13</td><td>65</td><td>43</td></tr> <tr><td>15</td><td>115</td><td>47</td><td>117</td><td>82</td></tr> <tr><td>75</td><td>68</td><td>12</td><td>92</td><td>50</td></tr> <tr><td>61</td><td>27</td><td>32</td><td>96</td><td>52</td></tr> <tr><td>81</td><td>104</td><td>18</td><td>46</td><td>58</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>51</td><td>102</td><td>55</td><td>7</td><td>88</td></tr> <tr><td>71</td><td>4</td><td>63</td><td>64</td><td>89</td></tr> <tr><td>56</td><td>111</td><td>112</td><td>14</td><td>119</td></tr> <tr><td>125</td><td>120</td><td>24</td><td>113</td><td>66</td></tr> <tr><td>29</td><td>80</td><td>124</td><td>85</td><td>36</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>76</td><td>83</td><td>67</td><td>2</td><td>48</td></tr> <tr><td>114</td><td>60</td><td>108</td><td>40</td><td>3</td></tr> <tr><td>94</td><td>25</td><td>105</td><td>72</td><td>33</td></tr> <tr><td>102</td><td>55</td><td>7</td><td>88</td><td>71</td></tr> <tr><td>4</td><td>63</td><td>64</td><td>89</td><td>84</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>121</td><td>1</td><td>37</td><td>30</td><td>59</td></tr> <tr><td>59</td><td>100</td><td>70</td><td>38</td><td>98</td></tr> <tr><td>38</td><td>98</td><td>5</td><td>11</td><td>103</td></tr> <tr><td>118</td><td>74</td><td>42</td><td>77</td><td>39</td></tr> <tr><td>16</td><td>21</td><td>53</td><td>34</td><td>31</td></tr> </table>	95	8	78	109	73	116	53	34	31	122	87	91	123	10	19	22	84	57	26	62	93	17	54	49	28	45	107	13	65	43	15	115	47	117	82	75	68	12	92	50	61	27	32	96	52	81	104	18	46	58	51	102	55	7	88	71	4	63	64	89	56	111	112	14	119	125	120	24	113	66	29	80	124	85	36	76	83	67	2	48	114	60	108	40	3	94	25	105	72	33	102	55	7	88	71	4	63	64	89	84	121	1	37	30	59	59	100	70	38	98	38	98	5	11	103	118	74	42	77	39	16	21	53	34	31
95	8	78	109	73																																																																																																																										
116	53	34	31	122																																																																																																																										
87	91	123	10	19																																																																																																																										
22	84	57	26	62																																																																																																																										
93	17	54	49	28																																																																																																																										
45	107	13	65	43																																																																																																																										
15	115	47	117	82																																																																																																																										
75	68	12	92	50																																																																																																																										
61	27	32	96	52																																																																																																																										
81	104	18	46	58																																																																																																																										
51	102	55	7	88																																																																																																																										
71	4	63	64	89																																																																																																																										
56	111	112	14	119																																																																																																																										
125	120	24	113	66																																																																																																																										
29	80	124	85	36																																																																																																																										
76	83	67	2	48																																																																																																																										
114	60	108	40	3																																																																																																																										
94	25	105	72	33																																																																																																																										
102	55	7	88	71																																																																																																																										
4	63	64	89	84																																																																																																																										
121	1	37	30	59																																																																																																																										
59	100	70	38	98																																																																																																																										
38	98	5	11	103																																																																																																																										
118	74	42	77	39																																																																																																																										
16	21	53	34	31																																																																																																																										
	<p style="text-align: center;">Objective Value: 792100</p> <table border="1" style="margin: auto;"> <tr><td>69</td><td>106</td><td>9</td><td>97</td><td>23</td></tr> <tr><td>101</td><td>122</td><td>87</td><td>91</td><td>123</td></tr> <tr><td>10</td><td>19</td><td>22</td><td>17</td><td>54</td></tr> <tr><td>49</td><td>28</td><td>45</td><td>107</td><td>13</td></tr> <tr><td>65</td><td>61</td><td>27</td><td>32</td><td>96</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>52</td><td>81</td><td>104</td><td>18</td><td>46</td></tr> <tr><td>58</td><td>110</td><td>35</td><td>41</td><td>20</td></tr> <tr><td>86</td><td>79</td><td>78</td><td>109</td><td>73</td></tr> <tr><td>116</td><td>43</td><td>15</td><td>115</td><td>47</td></tr> <tr><td>117</td><td>82</td><td>75</td><td>68</td><td>57</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>26</td><td>62</td><td>93</td><td>56</td><td>111</td></tr> <tr><td>112</td><td>14</td><td>119</td><td>125</td><td>120</td></tr> <tr><td>24</td><td>113</td><td>66</td><td>29</td><td>80</td></tr> <tr><td>124</td><td>85</td><td>36</td><td>76</td><td>83</td></tr> <tr><td>67</td><td>2</td><td>48</td><td>114</td><td>60</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>108</td><td>40</td><td>3</td><td>94</td><td>90</td></tr> <tr><td>99</td><td>44</td><td>25</td><td>105</td><td>72</td></tr> <tr><td>33</td><td>12</td><td>92</td><td>50</td><td>51</td></tr> <tr><td>102</td><td>55</td><td>7</td><td>88</td><td>71</td></tr> <tr><td>4</td><td>63</td><td>64</td><td>89</td><td>84</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>6</td><td>121</td><td>1</td><td>37</td><td>30</td></tr> <tr><td>59</td><td>100</td><td>70</td><td>95</td><td>8</td></tr> <tr><td>38</td><td>98</td><td>5</td><td>11</td><td>103</td></tr> <tr><td>118</td><td>74</td><td>42</td><td>77</td><td>39</td></tr> <tr><td>16</td><td>21</td><td>53</td><td>34</td><td>31</td></tr> </table>	69	106	9	97	23	101	122	87	91	123	10	19	22	17	54	49	28	45	107	13	65	61	27	32	96	52	81	104	18	46	58	110	35	41	20	86	79	78	109	73	116	43	15	115	47	117	82	75	68	57	26	62	93	56	111	112	14	119	125	120	24	113	66	29	80	124	85	36	76	83	67	2	48	114	60	108	40	3	94	90	99	44	25	105	72	33	12	92	50	51	102	55	7	88	71	4	63	64	89	84	6	121	1	37	30	59	100	70	95	8	38	98	5	11	103	118	74	42	77	39	16	21	53	34	31
69	106	9	97	23																																																																																																																										
101	122	87	91	123																																																																																																																										
10	19	22	17	54																																																																																																																										
49	28	45	107	13																																																																																																																										
65	61	27	32	96																																																																																																																										
52	81	104	18	46																																																																																																																										
58	110	35	41	20																																																																																																																										
86	79	78	109	73																																																																																																																										
116	43	15	115	47																																																																																																																										
117	82	75	68	57																																																																																																																										
26	62	93	56	111																																																																																																																										
112	14	119	125	120																																																																																																																										
24	113	66	29	80																																																																																																																										
124	85	36	76	83																																																																																																																										
67	2	48	114	60																																																																																																																										
108	40	3	94	90																																																																																																																										
99	44	25	105	72																																																																																																																										
33	12	92	50	51																																																																																																																										
102	55	7	88	71																																																																																																																										
4	63	64	89	84																																																																																																																										
6	121	1	37	30																																																																																																																										
59	100	70	95	8																																																																																																																										
38	98	5	11	103																																																																																																																										
118	74	42	77	39																																																																																																																										
16	21	53	34	31																																																																																																																										

	<p style="text-align: center;">Objective Value: 655966</p> <table border="1" style="margin: auto;"> <tr><td>123</td><td>10</td><td>19</td><td>22</td><td>17</td></tr> <tr><td>54</td><td>49</td><td>28</td><td>45</td><td>107</td></tr> <tr><td>13</td><td>65</td><td>61</td><td>27</td><td>32</td></tr> <tr><td>96</td><td>52</td><td>81</td><td>104</td><td>18</td></tr> <tr><td>46</td><td>58</td><td>86</td><td>79</td><td>78</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>109</td><td>73</td><td>116</td><td>43</td><td>15</td></tr> <tr><td>115</td><td>47</td><td>117</td><td>82</td><td>75</td></tr> <tr><td>68</td><td>12</td><td>92</td><td>50</td><td>51</td></tr> <tr><td>95</td><td>8</td><td>38</td><td>102</td><td>55</td></tr> <tr><td>7</td><td>88</td><td>71</td><td>4</td><td>90</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>99</td><td>44</td><td>63</td><td>64</td><td>89</td></tr> <tr><td>84</td><td>57</td><td>26</td><td>62</td><td>93</td></tr> <tr><td>56</td><td>111</td><td>112</td><td>14</td><td>119</td></tr> <tr><td>125</td><td>120</td><td>24</td><td>113</td><td>66</td></tr> <tr><td>29</td><td>80</td><td>124</td><td>85</td><td>36</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>76</td><td>83</td><td>67</td><td>2</td><td>48</td></tr> <tr><td>114</td><td>60</td><td>108</td><td>40</td><td>3</td></tr> <tr><td>94</td><td>25</td><td>105</td><td>72</td><td>33</td></tr> <tr><td>69</td><td>106</td><td>9</td><td>97</td><td>23</td></tr> <tr><td>101</td><td>6</td><td>121</td><td>1</td><td>37</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>30</td><td>59</td><td>100</td><td>70</td><td>110</td></tr> <tr><td>35</td><td>41</td><td>20</td><td>98</td><td>5</td></tr> <tr><td>11</td><td>103</td><td>118</td><td>74</td><td>42</td></tr> <tr><td>77</td><td>39</td><td>16</td><td>21</td><td>53</td></tr> <tr><td>34</td><td>31</td><td>122</td><td>87</td><td>91</td></tr> </table>	123	10	19	22	17	54	49	28	45	107	13	65	61	27	32	96	52	81	104	18	46	58	86	79	78	109	73	116	43	15	115	47	117	82	75	68	12	92	50	51	95	8	38	102	55	7	88	71	4	90	99	44	63	64	89	84	57	26	62	93	56	111	112	14	119	125	120	24	113	66	29	80	124	85	36	76	83	67	2	48	114	60	108	40	3	94	25	105	72	33	69	106	9	97	23	101	6	121	1	37	30	59	100	70	110	35	41	20	98	5	11	103	118	74	42	77	39	16	21	53	34	31	122	87	91
123	10	19	22	17																																																																																																																										
54	49	28	45	107																																																																																																																										
13	65	61	27	32																																																																																																																										
96	52	81	104	18																																																																																																																										
46	58	86	79	78																																																																																																																										
109	73	116	43	15																																																																																																																										
115	47	117	82	75																																																																																																																										
68	12	92	50	51																																																																																																																										
95	8	38	102	55																																																																																																																										
7	88	71	4	90																																																																																																																										
99	44	63	64	89																																																																																																																										
84	57	26	62	93																																																																																																																										
56	111	112	14	119																																																																																																																										
125	120	24	113	66																																																																																																																										
29	80	124	85	36																																																																																																																										
76	83	67	2	48																																																																																																																										
114	60	108	40	3																																																																																																																										
94	25	105	72	33																																																																																																																										
69	106	9	97	23																																																																																																																										
101	6	121	1	37																																																																																																																										
30	59	100	70	110																																																																																																																										
35	41	20	98	5																																																																																																																										
11	103	118	74	42																																																																																																																										
77	39	16	21	53																																																																																																																										
34	31	122	87	91																																																																																																																										
	<p style="text-align: center;">Objective Value: 585501</p> <table border="1" style="margin: auto;"> <tr><td>95</td><td>8</td><td>53</td><td>34</td><td>31</td></tr> <tr><td>122</td><td>87</td><td>91</td><td>123</td><td>10</td></tr> <tr><td>25</td><td>105</td><td>72</td><td>33</td><td>19</td></tr> <tr><td>22</td><td>17</td><td>54</td><td>49</td><td>28</td></tr> <tr><td>45</td><td>107</td><td>13</td><td>65</td><td>61</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>27</td><td>32</td><td>96</td><td>52</td><td>81</td></tr> <tr><td>104</td><td>18</td><td>46</td><td>58</td><td>110</td></tr> <tr><td>35</td><td>41</td><td>20</td><td>86</td><td>79</td></tr> <tr><td>78</td><td>109</td><td>73</td><td>116</td><td>43</td></tr> <tr><td>15</td><td>115</td><td>47</td><td>117</td><td>82</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>75</td><td>68</td><td>12</td><td>92</td><td>50</td></tr> <tr><td>51</td><td>102</td><td>55</td><td>7</td><td>88</td></tr> <tr><td>71</td><td>4</td><td>63</td><td>64</td><td>89</td></tr> <tr><td>84</td><td>57</td><td>26</td><td>62</td><td>93</td></tr> <tr><td>56</td><td>111</td><td>112</td><td>14</td><td>119</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>125</td><td>120</td><td>24</td><td>113</td><td>66</td></tr> <tr><td>29</td><td>80</td><td>124</td><td>85</td><td>36</td></tr> <tr><td>76</td><td>83</td><td>67</td><td>2</td><td>48</td></tr> <tr><td>114</td><td>60</td><td>108</td><td>40</td><td>3</td></tr> <tr><td>94</td><td>90</td><td>99</td><td>44</td><td>6</td></tr> </table> <table border="1" style="margin: auto;"> <tr><td>121</td><td>1</td><td>37</td><td>69</td><td>106</td></tr> <tr><td>9</td><td>97</td><td>23</td><td>101</td><td>30</td></tr> <tr><td>59</td><td>100</td><td>70</td><td>38</td><td>98</td></tr> <tr><td>5</td><td>11</td><td>103</td><td>118</td><td>74</td></tr> <tr><td>39</td><td>16</td><td>21</td><td>42</td><td>77</td></tr> </table>	95	8	53	34	31	122	87	91	123	10	25	105	72	33	19	22	17	54	49	28	45	107	13	65	61	27	32	96	52	81	104	18	46	58	110	35	41	20	86	79	78	109	73	116	43	15	115	47	117	82	75	68	12	92	50	51	102	55	7	88	71	4	63	64	89	84	57	26	62	93	56	111	112	14	119	125	120	24	113	66	29	80	124	85	36	76	83	67	2	48	114	60	108	40	3	94	90	99	44	6	121	1	37	69	106	9	97	23	101	30	59	100	70	38	98	5	11	103	118	74	39	16	21	42	77
95	8	53	34	31																																																																																																																										
122	87	91	123	10																																																																																																																										
25	105	72	33	19																																																																																																																										
22	17	54	49	28																																																																																																																										
45	107	13	65	61																																																																																																																										
27	32	96	52	81																																																																																																																										
104	18	46	58	110																																																																																																																										
35	41	20	86	79																																																																																																																										
78	109	73	116	43																																																																																																																										
15	115	47	117	82																																																																																																																										
75	68	12	92	50																																																																																																																										
51	102	55	7	88																																																																																																																										
71	4	63	64	89																																																																																																																										
84	57	26	62	93																																																																																																																										
56	111	112	14	119																																																																																																																										
125	120	24	113	66																																																																																																																										
29	80	124	85	36																																																																																																																										
76	83	67	2	48																																																																																																																										
114	60	108	40	3																																																																																																																										
94	90	99	44	6																																																																																																																										
121	1	37	69	106																																																																																																																										
9	97	23	101	30																																																																																																																										
59	100	70	38	98																																																																																																																										
5	11	103	118	74																																																																																																																										
39	16	21	42	77																																																																																																																										

	BEST STATE Objective Value: 556801							-----	----	-----	-----	-----		95	8	78	109	73		116	53	34	31	122		87	91	123	10	19		22	84	57	26	62		93	17	54	49	28								----	-----	----	-----	----		45	107	13	65	43		15	115	47	117	82		75	68	12	92	50		61	27	32	96	52		81	104	18	46	58								-----	-----	-----	-----	-----		51	102	55	7	88		71	4	63	64	89		56	111	112	14	119		125	120	24	113	66		29	80	124	85	36								-----	-----	-----	----	----		76	83	67	2	48		114	60	108	40	3		94	25	105	72	33		69	106	9	97	23		101	6	121	1	37								----	-----	-----	--------	-----		30	59	100	70	110		35	41	20	98	5		11	103	118	74	42		77	39	16	21</td		

Plot Objective Value terhadap Iterasi



BAB V

ANALISIS DAN PEMBAHASAN

Berdasarkan eksperimen yang dilakukan terhadap setiap algoritma pencarian lokal pada bagian sebelumnya, berikut ini disajikan tabel nilai *objective value* dari state terbaik yang diperoleh oleh setiap algoritma:

Algoritma	Objective Value	Average Time (ms)
Steepest Ascent Hill Climbing	188	1198,485
Hill-climbing with Sideway	117	1257,841
Hill-climbing with Random Restart	128	3380 – 4990 (tergantung banyak restart)
Stochastic Hill-climbing	205	1122,331
Simulated Annealing	58	632,449
Genetic Algorithm	556801	0 – 6000 (tergantung banyak restart)

		iterasi)
--	--	----------

Berdasarkan data pada tabel tersebut, terlihat bahwa algoritma *Simulated Annealing* memiliki *objective value* paling mendekati *global objective value*, yaitu 0. Algoritma ini juga selalu konsisten memberikan hasil terbaik di antara algoritma lainnya. Hal ini terjadi karena mekanisme probabilistiknya yang memungkinkan perpindahan ke *neighbor* dengan *objective value* yang lebih buruk berdasarkan temperatur yang perlahan-lahan menurun. Dengan karakteristik ini, *Simulated Annealing* dapat keluar dari jebakan *local maxima*, sehingga memperbesar kemungkinan mencapai solusi yang lebih optimal. Selain itu, algoritma ini hanya perlu menghasilkan *neighbor* secara acak sehingga waktu eksekusinya relatif cepat. Meski demikian, algoritma ini memerlukan waktu *convergence* yang cukup lama karena harus menjalankan sejumlah besar iterasi, khususnya pada temperatur rendah. Peningkatan pada *cooling rate* juga dapat meningkatkan kualitas solusi, tetapi hal ini akan meningkatkan biaya komputasi karena jumlah iterasi yang lebih banyak.

Algoritma *Hill-climbing with Sideway* menempati posisi kedua dalam hal *objective value* terbaik. Algoritma ini bekerja dengan berpindah ke *state neighbor* yang memiliki nilai *objective value* lebih kecil atau sama dengan *state* saat ini, sehingga dapat mengatasi kendala *local maxima*. Strategi ini memberikan fleksibilitas lebih dibandingkan *Steepest Ascent* karena memungkinkan pergerakan pada nilai yang sama (*sideway move*), tetapi terdapat risiko algoritma terjebak di *local plateau*—suatu wilayah solusi di mana nilai *objective value* stagnan—sehingga algoritma berpotensi berjalan tanpa batas. Untuk mengatasi hal ini, diperlukan pembatasan jumlah pergerakan *sideway* secara berurutan. Secara umum, algoritma ini memiliki waktu eksekusi yang lebih lama dibandingkan *Simulated Annealing* karena mengevaluasi semua *neighbor*.

Selanjutnya, algoritma *Hill-climbing with Random Restart* menduduki posisi ketiga dalam pencapaian *objective value*. Algoritma ini memperbaiki kelemahan *Steepest Ascent* melalui strategi *random restart*, yakni memulai pencarian dari beberapa titik awal yang berbeda secara acak. Hal ini meningkatkan kemungkinan menemukan solusi yang lebih baik, terutama dalam kasus masalah yang memiliki banyak *local maxima*. Namun, kelemahan algoritma ini adalah ketergantungannya pada *random initial state*, yang membuat waktu *convergence* sangat bervariasi tergantung jumlah *restart* dan karakteristik dari *state* awal yang dipilih.

Algoritma *Steepest Ascent Hill Climbing* menghasilkan *objective value* terbaik keempat dan memiliki waktu *convergence* paling cepat di antara algoritma yang diuji. Algoritma ini selalu memilih *neighbor* dengan *objective value* terbaik pada setiap iterasi, yang membuatnya cenderung lebih efisien dari segi waktu. Namun, algoritma ini mudah terjebak di *local maxima* karena tidak memiliki mekanisme untuk keluar dari jebakan tersebut, sehingga kinerjanya terbatas pada struktur lanskap solusi.

Pada posisi selanjutnya, *Stochastic Hill-climbing* memiliki *objective value* yang relatif buruk karena pergerakannya yang cenderung acak, sehingga kurang konsisten dalam menemukan solusi yang optimal. Algoritma ini hanya berpindah ke *neighbor* dengan *objective value* lebih kecil secara acak, sehingga memiliki kemungkinan tinggi terjebak pada *local maxima*. Namun, dari segi waktu

eksekusi, algoritma ini relatif cepat karena menggunakan mekanisme pembangkitan *neighbor* secara acak.

Genetic Algorithm (GA) memiliki karakteristik unik dibandingkan algoritma pencarian lokal lainnya, terutama karena pendekatan berbasis populasi dan prinsip evolusi yang diusungnya. Dengan menggunakan mekanisme crossover dan mutation, GA mampu mengeksplorasi ruang solusi yang lebih luas dan memperkenalkan variasi di setiap generasi baru. Namun, algoritma ini tidak selalu menghasilkan populasi dengan objective value yang optimal, karena proses crossover dan mutation dilakukan secara acak. Selain itu, performa GA sangat bergantung pada jenis masalah yang ingin dipecahkan. Dalam kasus tertentu, algoritma ini mungkin kurang efektif. Hal ini terlihat jelas pada tabel di atas, dimana GA memiliki best objective value yang paling buruk.

Hal lain yang perlu diperhatikan adalah kesulitan untuk mencapai *global objective value* karena ruang pencarian dalam masalah ini sangat besar, yaitu 125!, sementara hanya ada satu konfigurasi solusi yang benar (dengan beberapa susunan lainnya yang simetris atau identik dalam perspektif kubus yang berbeda). Ruang pencarian yang besar ini juga menyebabkan adanya banyak *local maxima* yang dapat menjebak algoritma pencarian lokal, sehingga algoritma rentan berhenti pada solusi yang tidak optimal.

KESIMPULAN

Berdasarkan eksperimen dan analisis yang telah dilakukan, algoritma pencarian lokal yang menunjukkan kinerja terbaik dalam hal *objective value* dan waktu eksekusi adalah *Simulated Annealing*. Algoritma ini secara konsisten mampu menemukan solusi mendekati optimal dalam waktu yang relatif cepat, menjadikannya pilihan yang efisien untuk berbagai jenis masalah pencarian lokal. Di sisi lain, algoritma dengan performa terburuk dari segi *objective value* dalam percobaan ini adalah *Genetic Algorithm*. Hal ini disebabkan oleh sifat *crossover* dan *mutation* yang acak, yang membuat GA cenderung mengalami fluktuasi performa tergantung pada masalah yang dipecahkan dan parameter yang digunakan.

r

REFERENSI

- E. W. Weisstein, "Magic Cube," *MathWorld*. [Online]. Available: <https://mathworld.wolfram.com/MagicCube.html>. [Accessed: Oct. 2, 2024].
- E. W. Weisstein, "Perfect Magic Cube," *MathWorld*. [Online]. Available: <https://mathworld.wolfram.com/PerfectMagicCube.html>. [Accessed: Oct. 2, 2024].
- Planète Maths, "Perfect Magic Cubes." [Online]. Available: <http://www.multimagie.com/English/Perfectcubes.htm>. [Accessed: Oct. 2, 2024].
- S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson Higher Education, 2019.
- Dr. Masayu Leylia Khodra, S.T, M.T., "Artificial Intelligence," Class lecture, 2024. [PowerPoint slides].

PEMBAGIAN TUGAS

NIM	Task
13522068	Server (State, Genetic Algorithm, API setup, Handler for Genetic Algorithm), Report
13522093	Clients (UI) + Video Player, Report
13522098	Server (Hill Climbing, Simulated Annealing, Handler for Hill Climbing ada Simulated Annealing)
13522118	Clients (UI) + Video Player, Report