Tugas Besar 1

IF3070 Dasar Inteligensi Artifisial

Pencarian Solusi *Diagonal Magic Cube* dengan *Local Search*

Dosen Pengampu: Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

Oleh:

Syakira Fildza Nazhifan	18222012
Nicolaas Heru Dreandachrista	18222050
Ardra Rafif Sahasika	18222134
Timothy Haposan Simanjuntak	18222137

Kelompok 37

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Deskripsi Persoalan

Tugas ini mengharuskan peserta menyelesaikan permasalahan *Diagonal Magic Cube* berukuran 5x5x5. *Diagonal Magic Cube* adalah sebuah kubus yang tersusun dari angka-angka unik dari 1 hingga n^3 (dalam kasus ini 1 hingga 125, karena n = 5) tanpa pengulangan. Kubus ini harus memenuhi beberapa properti khusus yang berkaitan dengan *Magic Number*.

Magic Number adalah suatu nilai khusus yang menjadi target jumlah angka-angka dalam kubus, tetapi angka ini tidak termasuk di dalam angka-angka yang harus dimasukkan ke dalam kubus. Properti yang harus dipenuhi oleh *Magic Cube* adalah:

1. Jumlah angka-angka dalam setiap baris pada kubus harus sama dengan *Magic Number*.
2. Jumlah angka-angka dalam setiap kolom pada kubus harus sama dengan *Magic Number*.
3. Jumlah angka-angka dalam setiap tiang (*depth/level*) pada kubus harus sama dengan *Magic Number*.
4. Jumlah angka-angka dalam setiap diagonal ruang (diagonal 3D) pada kubus harus sama dengan *Magic Number*.
5. Jumlah angka-angka dalam setiap diagonal pada potongan bidang 2D dari kubus juga harus sama dengan *Magic Number*.

Tugas peserta adalah untuk memulai dari initial state berupa susunan acak angka 1 hingga 125 di dalam kubus 5x5x5. Dalam setiap iterasi algoritma *local search*, peserta diperbolehkan melakukan satu langkah, yaitu menukar posisi dua angka pada kubus. Untuk *genetic algorithm*, peserta boleh menukar lebih dari dua angka dalam satu iterasi.

Tujuan utama dari tugas ini adalah untuk mencari solusi yang memenuhi semua properti dari *Diagonal Magic Cube*, di mana semua jumlah pada baris, kolom, tiang, diagonal, dan potongan bidang 2D sama dengan *Magic Number*. Peserta akan mengimplementasikan tiga algoritma *local search* untuk menyelesaikan masalah ini, di antaranya:

1. *Hill-climbing with Sideways Move*
2. *Simulated Annealing*
3. *Genetic Algorithm*

Peserta juga diminta untuk menjelaskan *objective function* yang digunakan, menggambarkan bagaimana setiap algoritma bekerja untuk mencari solusi yang *feasible*, serta memberikan analisis mengenai algoritma mana yang terbaik dan mengapa.

Pembahasan

A. Pemilihan Objective Function

Magic cube 5x5x5 adalah perpanjangan tiga dimensi dari *magic square* dengan sebuah prinsip yaitu penjumlahan antara angka di *row, column, level,* dan *diagonal* adalah sama. Cara untuk menghitung *magic constant* adalah sebagai berikut:

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

Keterangan:

1. M adalah *magic constant*
2. N adalah dimensi dari *magic cube*

Berdasarkan formula tersebut, didapatkan *magic constant*/ konstanta ajaib untuk dimensi 5x5x5 adalah 315. Lalu, *magic constant* ini akan digunakan dalam menentukan fungsi objektif. Fungsi objektif yang akan digunakan adalah fungsi yang mencoba meminimalisasi selisih antara jumlah setiap *row, column, level,* dan *diagonal* dengan konstanta ajaib yaitu 315. Berikut adalah formula yang akan kami gunakan

1. Menghitung error pada setiap baris:

$$f(x)\ row = \sum_{i=1}^{5} \sum_{k=1}^{5} \left| \sum_{j=1}^{5} x[i][j][k] - 315 \right|$$

2. Menghitung error pada setiap kolom:

$$f(x)\ column = \sum_{j=1}^{5} \sum_{k=1}^{5} \left| \sum_{i=1}^{5} x[i][j][k] - 315 \right|$$

3. Menghitung error pada setiap level (vertikal):

$$f(x)\ level = \sum_{j=1}^{5} \sum_{j=1}^{5} \left| \sum_{k=1}^{5} x[i][j][k] - 315 \right|$$

4. Menghitung error pada diagonal ruang:

$$f(x)\ space\ diagonal = \left| \sum_{i=1}^{5} x[i][i][i] - 315 \right| + \left| \sum_{i=1}^{5} x[i][i][6 - i] - 315 \right|$$
$$+ \left| \sum_{i=1}^{5} x[6 - i][6 - i][i] - 315 \right| + \left| \sum_{i=1}^{5} x[6 - i][6 - i][6 - i] - 315 \right|$$

5. Menghitung error pada diagonal bidang potongan sejajar sumbu i:

$$f(x)\ diagonal\ i = \sum_{i=1}^{5} \left| \sum_{j=1}^{5} x[i][j][j] - 315 \right| + \sum_{i=1}^{5} \left| \sum_{j=1}^{5} x[i][j][6 - j] - 315 \right|$$

6. Menghitung error pada diagonal bidang potongan sejajar sumbu j:

$$f(x)\ diagonal\ j\ =\ \sum_{j=1}^{5}\left|\sum_{i=1}^{5} x[i][j][i]\ -\ 315\right| + \sum_{i=1}^{5}\left|\sum_{j=1}^{5} x[i][j][6\ -\ i]\ -\ 315\right|$$

7. Menghitung error pada diagonal bidang potongan sejajar sumbu k:

$$f(x)\ diagonal\ k\ =\ \sum_{j=1}^{5}\left|\sum_{i=1}^{5} x[i][i][k]\ -\ 315\right| + \sum_{i=1}^{5}\left|\sum_{j=1}^{5} x[i][6\ -\ i][k]\ -\ 315\right|$$

Total Objective Function

$$f(x)\ all\ =\ f(x)\ row\ +\ f(x)\ column\ +\ f(x)\ level\ +\ f(x)\ space\ diagonal$$
$$+\ f(x)\ diagonal\ i\ +\ f(x)\ diagonal\ j\ +\ f(x)\ diagonal\ k$$

Keterangan:

1. f(x) adalah fungsi yang memiliki nilai total dari selisih absolut antara jumlah elemen dengan konstanta ajaib
2. Indeks i, j, k, adalah penanda posisi elemen-elemen dalam kubus 3 dimensi yaitu baris, kolom, dan level.
3. $x[i][j][k]$ merujuk pada elemen spesifik pada posisi i, j, k
4. Tanda mutlak/absolut digunakan sebagai inisiasi penghindaran angka negatif pada pengurangan nilai di dalam tanda tersebut.

B. Penjelasan Implementasi Algoritma Local Search

B.1 Hill-Climbing with Sideways Move

C.1.1 Deskripsi Fungsi/Kelas

Tipe	Nama Fungsi	Deskripsi
Function	main	Fungsi utama program. Menginisialisasi kubus, mengevaluasi nya, dan secara iteratif meningkatkan fitness kubus menggunakan algoritma sideways hill climbing untuk meminimalkan kesalahan. Menampilkan kondisi awal dan akhir serta mencatat performa.
Function	initialize_cube	Menginisialisasi kubus 3D (5x5x5) dengan angka unik dari 1 hingga TOTAL_NUMBERS, disusun secara acak. Menggunakan metode pengacakan untuk mengisi kubus dengan angka-angka secara acak.
Function	print_cube	Menampilkan kubus 5x5x5 dalam format yang mudah dibaca, lengkap dengan garis pembatas dan judul untuk setiap lapisan (slice). Berguna untuk visualisasi kondisi terkini kubus.
Function	evaluate	Menghitung fitness score dari kubus, yaitu selisih dari target “magic number” untuk setiap baris, kolom, dan diagonal. Digunakan untuk menentukan seberapa dekat kubus dengan konfigurasi ideal.
Function	swap	Menukar dua nilai integer, digunakan untuk menghasilkan tetangga baru dalam proses hill climbing.
Function	generate_all_neighbors	Menghasilkan dan mengevaluasi semua kemungkinan tetangga dari kubus saat ini dengan menukar elemen-elemen. Memperbarui konfigurasi terbaik jika ditemukan tetangga yang lebih baik atau setara (dengan mempertimbangkan pergerakan sideways).
Function	copy_cube	Menyalin isi dari satu kubus ke kubus lain. Berguna untuk menjaga kondisi kubus saat ini dan yang terbaik selama proses optimisasi.
Function	Konstanta	N mendefinisikan ukuran kubus; TOTAL_NUMBERS adalah jumlah total nilai unik dalam kubus; MAGIC_NUMBERadalah target jumlah untuk setiap baris, kolom, dan diagonal; MAX_SIDEWAYS_MOVES adalah batas

		maksimal pergerakan sideways yang diperbolehkan sebelum berhenti.

C.1.2 Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

#define N 5
#define TOTAL_NUMBERS (N * N * N)
#define MAGIC_NUMBER (N * (TOTAL_NUMBERS + 1)) / 2
#define MAX_SIDEWAYS_MOVES 5000  // Limit for sideways moves

// Function prototypes
void initialize_cube(int cube[N][N][N]);
void print_cube(int cube[N][N][N]);
int evaluate(int cube[N][N][N]);
void swap(int *a, int *b);
void generate_all_neighbors(int cube[N][N][N], int best_cube[N][N][N],
int *best_error);
void copy_cube(int src[N][N][N], int dest[N][N][N]);

int main() {
   clock_t start_time = clock();
   srand(time(0));

   int current_cube[N][N][N];
   int best_cube[N][N][N];
   int current_error, best_error;
   int sideways_moves = 0;

   FILE *fptr = fopen("objective_function.txt", "w");
   if (fptr == NULL) {
       printf("Error opening file!\n");
       exit(1);
   }

   // Initialize the cube with random values
   initialize_cube(current_cube);
   current_error = evaluate(current_cube);

   printf("Initial Cube:\n");
   print_cube(current_cube);
   printf("Initial Error: %d\n", current_error);
```

```
    int iterations = 0;
    while (current_error > 0) {
        iterations++;

        // Generate all neighbors and choose the best one
        best_error = current_error;
        generate_all_neighbors(current_cube, best_cube, &best_error);

        // If a better neighbor is found, reset sideways move counter
        if (best_error < current_error) {
            copy_cube(best_cube, current_cube);
            current_error = best_error;
            sideways_moves = 0;  // Reset sideways move counter
        }
        // If no improvement, but the neighbor has the same error
(sideways move)
        else if (best_error == current_error && sideways_moves <
MAX_SIDEWAYS_MOVES) {
            copy_cube(best_cube, current_cube);
            current_error = best_error;
            sideways_moves++;  // Increment sideways move counter
            // printf("Sideways move: %d\n", sideways_moves);
        }
        // If no better neighbor and sideways limit is reached, stop
(local optimum)
        else {
            printf("Reached local optimum or sideways move limit.
Stopping.\n");
            break;
        }

        fprintf(fptr, "%d %d\n", iterations, current_error);


        // Print status every 10 iterations
        if (iterations % 1000 == 0) {
            printf("Iteration %d - Current Error: %d\n", iterations,
current_error);
        }
    }

    printf("Final Cube after %d iterations:\n", iterations);
    print_cube(current_cube);
    printf("Final Error: %d\n", current_error);
    fclose(fptr);

    // Record the end time
    clock_t end_time = clock();
```

```c
    // Calculate the time difference in seconds
    double duration = (double)(end_time - start_time) / CLOCKS_PER_SEC;

    printf("Program execution time: %.2f seconds\n", duration);

    return 0;
}

// Initialize the cube with numbers from 1 to TOTAL_NUMBERS randomly
void initialize_cube(int cube[N][N][N]) {
    int numbers[TOTAL_NUMBERS];
    for (int i = 0; i < TOTAL_NUMBERS; i++) {
        numbers[i] = i + 1;
    }

    // Shuffle the numbers array
    for (int i = TOTAL_NUMBERS - 1; i > 0; i--) {
        int j = rand() % (i + 1);
        swap(&numbers[i], &numbers[j]);
    }

    // Fill the cube with the shuffled numbers
    int idx = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                cube[i][j][k] = numbers[idx++];
            }
        }
    }
}

// Print the 5x5x5 cube
void print_cube(int cube[N][N][N]) {
    for (int i = 0; i < N; i++) {
        printf("Slice %d:\n", i + 1);

        // Print the top border for the slice
        printf("    +");
        for (int k = 0; k < N; k++) {
            printf("-----+");
        }
        printf("\n");

        for (int j = 0; j < N; j++) {
            printf("    | ");
            for (int k = 0; k < N; k++) {
```

```
                printf("%3d | ", cube[i][j][k]);
            }
            printf("\n");

            // Print the row separator
            printf("    +");
            for (int k = 0; k < N; k++) {
                printf("-----+");
            }
            printf("\n");
        }

        printf("\n"); // Add a newline between slices for better
readability
    }
}

// Evaluate the error of the cube (difference from MAGIC_NUMBER)
int evaluate(int cube[N][N][N]) {
    int error = 0;
    int sum;

    // Evaluate rows
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            sum = 0;
            for (int k = 0; k < N; k++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate columns
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
            for (int i = 0; i < N; i++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate pillars
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
```

```
            for (int j = 0; j < N; j++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate main space diagonals
    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][i];
    }
    error += abs(sum - MAGIC_NUMBER);

    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][N - i - 1];
    }
    error += abs(sum - MAGIC_NUMBER);

    // Evaluate diagonals in horizontal (x-y) slices
    for (int i = 0; i < N; i++) {
        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][j];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][N - j - 1];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    // Evaluate diagonals in vertical (y-z) slices
    for (int j = 0; j < N; j++) {
        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[k][j][k];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[N - k - 1][j][k];
        }
        error += abs(sum - MAGIC_NUMBER);
```

```
   }

   // Evaluate diagonals in vertical (x-z) slices
   for (int k = 0; k < N; k++) {
       sum = 0;
       for (int i = 0; i < N; i++) {
           sum += cube[i][i][k];
       }
       error += abs(sum - MAGIC_NUMBER);

       sum = 0;
       for (int i = 0; i < N; i++) {
           sum += cube[i][N - i - 1][k];
       }
       error += abs(sum - MAGIC_NUMBER);
   }

   return error;
}

// Swap two integers
void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}

// Generate all possible neighbors by swapping two elements in the cube
void generate_all_neighbors(int cube[N][N][N], int best_cube[N][N][N],
int *best_error) {
   int temp_cube[N][N][N];
   copy_cube(cube, temp_cube);

   // Try all possible pairs of positions in the cube
   for (int i1 = 0; i1 < N; i1++) {
       for (int j1 = 0; j1 < N; j1++) {
           for (int k1 = 0; k1 < N; k1++) {
               for (int i2 = i1; i2 < N; i2++) {
                   for (int j2 = (i1 == i2 ? j1 + 1 : 0); j2 < N; j2++)
{
                       for (int k2 = (i1 == i2 && j1 == j2 ? k1 + 1 :
0); k2 < N; k2++) {
                           // Swap two elements
                           swap(&temp_cube[i1][j1][k1],
&temp_cube[i2][j2][k2]);

                           // Evaluate the neighbor
                           int neighbor_error = evaluate(temp_cube);
```

```
                           // If this neighbor is better or equal (for
sideways), remember it
                           if (neighbor_error <= *best_error) {
                               copy_cube(temp_cube, best_cube);
                               *best_error = neighbor_error;
                           }

                           // Swap back to restore the original cube
                           swap(&temp_cube[i1][j1][k1],
&temp_cube[i2][j2][k2]);
                       }
                   }
               }
           }
       }
   }
}

// Copy one cube to another
void copy_cube(int src[N][N][N], int dest[N][N][N]) {
   for (int i = 0; i < N; i++) {
       for (int j = 0; j < N; j++) {
           for (int k = 0; k < N; k++) {
               dest[i][j][k] = src[i][j][k];
           }
       }
   }
}
```

B.2 Simulated Annealing

B.2.1 Deskripsi Fungsi/Kelas

Tipe	Nama Fungsi	Deskripsi
Fungsi	main()	Fungsi utama untuk menjalankan algoritma dan skema keseluruhan
Fungsi	initialize_cube(int cube[N][N][N])	Menginisiasi kubus 3D dengan angka random dar 1 hingga TOTAL_NUMBER.
Fungsi	print_cube(int cube[N][N][N])	Melakukan display kubus 3D dengan format terstruktur dan pembagian tiap lapisan
Fungsi	evaluate(int	Menghitung objective function dengan

	cube[N][N][N])	konsep “error”
Fungsi	swap(int *a, int *b)	Menukar dua nilai integer
Fungsi	generate_random_neighbor(int cube[N][N][N], int new_cube[N][N][N])	Membuat kubus tetangga acak dengan menyalin kubus saat ini dan melakukan pertukaran dua elemen secara acak di dalamnya.
Fungsi	copy_cube(int src[N][N][N], int dest[N][N][N])	Menyalin nilai dari satu kubus ke kubus lainnya
Fungsi	acceptance_probability(int current_error, int new_error, double temperature)	Menghitung probabilitas penerimaan untuk solusi yang lebih buruk berdasarkan perbedaan nilai error dan suhu saat ini. Jika solusi baru lebih buruk, fungsi ini menggunakan rumus eksponensial untuk menentukan penerimaan solusi tersebut sesuai dengan konsep simulated annealing.

B.2.2 Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define N 5
#define TOTAL_NUMBERS (N * N * N)
#define MAGIC_NUMBER (N * (TOTAL_NUMBERS + 1)) / 2
#define INITIAL_TEMPERATURE 1000.0
#define FINAL_TEMPERATURE 0.001
#define ALPHA 0.999
#define MAX_ITERATIONS 100000
#define STUCK_THRESHOLD 100  // Define how many iterations of
no improvement is considered 'stuck'

// Function prototypes
void initialize_cube(int cube[N][N][N]);
void print_cube(int cube[N][N][N]);
int evaluate(int cube[N][N][N]);
void swap(int *a, int *b);
void generate_random_neighbor(int cube[N][N][N], int
new_cube[N][N][N]);
void copy_cube(int src[N][N][N], int dest[N][N][N]);
double acceptance_probability(int current_error, int
```

```c
new_error, double temperature);

int main() {
    clock_t start_time = clock();
    srand(time(0));

    int current_cube[N][N][N];
    int new_cube[N][N][N];
    int current_error, new_error;
    double temperature = INITIAL_TEMPERATURE;

    // Initialize the cube with random values
    initialize_cube(current_cube);
    current_error = evaluate(current_cube);

    printf("Initial Cube:\n");
    print_cube(current_cube);
    printf("Initial Error: %d\n", current_error);

    int iterations = 0;
    int stuck_count = 0;  // Counter for local optima "stuck"
cases
    int no_improvement_count = 0;  // To detect prolonged
lack of improvement

    // Array to store acceptance probability values for
plotting
    double acceptance_probs[MAX_ITERATIONS];

    while (temperature > FINAL_TEMPERATURE) {
        iterations++;

        // Generate a random neighbor
        generate_random_neighbor(current_cube, new_cube);
        new_error = evaluate(new_cube);

        double prob = acceptance_probability(current_error,
new_error, temperature);
        acceptance_probs[iterations - 1] = prob;  // Store
the probability for plotting

        // Accept the new solution if it's better, or with a
probability if worse
        if (new_error < current_error || prob >
((double)rand() / RAND_MAX)) {
            copy_cube(new_cube, current_cube);
```

```
            current_error = new_error;
            no_improvement_count = 0;  // Reset count as
we've found improvement
        } else {
            no_improvement_count++;
        }

        // Check if we've been stuck in local optima for a
while
        if (no_improvement_count >= STUCK_THRESHOLD) {
            stuck_count++;
            no_improvement_count = 0;  // Reset to avoid
repeated counting
        }

        // Cool down the temperature
        temperature *= ALPHA;

        // Print status every 1000 iterations
        if (iterations % 10000 == 0) {
            printf("Iteration %d - Current Error: %d -
Temperature: %.2f\n", iterations, current_error,
temperature);
        }
    }

    printf("Final Cube after %d iterations:\n", iterations);
    print_cube(current_cube);
    printf("Final Error: %d\n", current_error);
    printf("Total stuck occurrences (local optima): %d\n",
stuck_count);

    // Plotting acceptance probability (to be done outside C
or by exporting data)
    FILE *fptr = fopen("acceptance_probs.txt", "w");
    for (int i = 0; i < iterations; i++) {
        fprintf(fptr, "%d %f\n", i + 1, acceptance_probs[i]);
    }
    fclose(fptr);
    printf("Acceptance probabilities saved to
'acceptance_probs.txt'.\n");

    // Record the end time
    clock_t end_time = clock();

    // Calculate the time difference in seconds
```

```c
    double duration = (double)(end_time - start_time) /
CLOCKS_PER_SEC;

    printf("Program execution time: %.2f seconds\n",
duration);
    return 0;
}

// Initialize the cube with numbers from 1 to TOTAL_NUMBERS
randomly
void initialize_cube(int cube[N][N][N]) {
    int numbers[TOTAL_NUMBERS];
    for (int i = 0; i < TOTAL_NUMBERS; i++) {
        numbers[i] = i + 1;
    }

    // Shuffle the numbers array
    for (int i = TOTAL_NUMBERS - 1; i > 0; i--) {
        int j = rand() % (i + 1);
        swap(&numbers[i], &numbers[j]);
    }

    // Fill the cube with the shuffled numbers
    int idx = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                cube[i][j][k] = numbers[idx++];
            }
        }
    }
}

// Print the 5x5x5 cube
void print_cube(int cube[N][N][N]) {
    for (int i = 0; i < N; i++) {
        printf("Slice %d:\n", i + 1);

        // Print the top border for the slice
        printf("    +");
        for (int k = 0; k < N; k++) {
            printf("-----+");
        }
        printf("\n");

        for (int j = 0; j < N; j++) {
```

```
            printf("    | ");
            for (int k = 0; k < N; k++) {
                printf("%3d | ", cube[i][j][k]);
            }
            printf("\n");

            // Print the row separator
            printf("    +");
            for (int k = 0; k < N; k++) {
                printf("-----+");
            }
            printf("\n");
        }

        printf("\n"); // Add a newline between slices for
better readability
    }
}

// Evaluate the error of the cube (difference from
MAGIC_NUMBER)
int evaluate(int cube[N][N][N]) {
    int error = 0;
    int sum;

    // Evaluate rows
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            sum = 0;
            for (int k = 0; k < N; k++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate columns
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
            for (int i = 0; i < N; i++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }
```

```
    // Evaluate pillars
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
            for (int j = 0; j < N; j++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate main space diagonals
    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][i];
    }
    error += abs(sum - MAGIC_NUMBER);

    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][N - i - 1];
    }
    error += abs(sum - MAGIC_NUMBER);

    // Evaluate diagonal in slices
// Evaluate diagonals in horizontal (x-y) slices
    for (int i = 0; i < N; i++) {
        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][j];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][N - j - 1];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    // Evaluate diagonals in vertical (y-z) slices
    for (int j = 0; j < N; j++) {
        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[k][j][k];
```

```
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[N - k - 1][j][k];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    // Evaluate diagonals in vertical (x-z) slices
    for (int k = 0; k < N; k++) {
        sum = 0;
        for (int i = 0; i < N; i++) {
            sum += cube[i][i][k];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int i = 0; i < N; i++) {
            sum += cube[i][N - i - 1][k];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    return error;
}

// Swap two integers
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Generate a random neighbor by swapping two elements in the
cube
void generate_random_neighbor(int cube[N][N][N], int
new_cube[N][N][N]) {
    copy_cube(cube, new_cube);

    // Randomly select two positions to swap
    int i1 = rand() % N, j1 = rand() % N, k1 = rand() % N;
    int i2 = rand() % N, j2 = rand() % N, k2 = rand() % N;

    // Swap two elements
```

```
    swap(&new_cube[i1][j1][k1], &new_cube[i2][j2][k2]);
}

// Copy one cube to another
void copy_cube(int src[N][N][N], int dest[N][N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                dest[i][j][k] = src[i][j][k];
            }
        }
    }
}

// Calculate the acceptance probability for worse solutions
double acceptance_probability(int current_error, int
new_error, double temperature) {
    if (new_error < current_error) {
        return 1.0;
    }
    return exp((double)(current_error - new_error) /
temperature);
}
```

B.3 Genetic Algorithm

B.3.1 Deskripsi Fungsi/Kelas

Tipe	Nama Fungsi/Kelas	Deskripsi
Kelas	Individual	Merepresentasikan individu dalam populasi: ● *cube*: Matriks 3D dengan dimensi 5x5x5 (int[N][N][N]) ● *fitness*: Skor fitness dari individu
Fungsi	initialize_cube(int cube[N][N][N])	Menginisialisasi sebuah *cube* dengan nilai unik dan acak dari 1 hingga N^3
Fungsi	evaluate(int cube[N][N][N])	Menghitung nilai objektif
Fungsi	generate_population(Individual population[], int population_size)	Menginisialisasi populasi individu secara acak dan menghitung *fitness* untuk setiap individu
Fungsi	generate_individual(Indi	Memilih individu terbaik dari beberapa

	vidual population[], int population_size)	individu secara acak
Fungsi	crossover(Individual *parent1, Individual *parent2, Individual *child)	Melakukan *crossover* untuk menghasilkan anak baru dari dua *parent* menggunakan metode *cycle crossover*
Fungsi	mutation(Individual *individual)	Melakukan mutasi pada suatu individu dengan menukar nilai dalam *cube*
Fungsi	find_best_individual(Individual population[], int population_size)	Menentukan individu terbaik dari suatu populasi
Fungsi	main()	Fungsi utama yang menjalankan algoritma dengan mengatur parameter (jumlah iterasi dan jumlah populasi), mengeksekusi fungsi algoritma genetika, dan melakukan *output* hasil akhir *cube*

B.3.2 Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>
#include <math.h>

#define N 5                                                   // Size
of the cube
#define TOTAL_NUMBERS (N * N * N)                             // Total
number of cubes
#define MAGIC_NUMBER (N * (TOTAL_NUMBERS + 1)) / 2  // Magic
number of cubes

typedef struct {
    int cube[N][N][N];  // The N x N x N cube
    int fitness;        // Fitness score
} Individual;

// Swap two integers
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Initialize the cube with numbers from 1 to N^3 randomly
```

```c
void initialize_cube(int cube[N][N][N]) {
    int numbers[TOTAL_NUMBERS];
    for (int i = 0; i < TOTAL_NUMBERS; i++) {
        numbers[i] = i + 1;
    }

    // Shuffle the numbers array
    for (int i = TOTAL_NUMBERS - 1; i > 0; i--) {
        int j = rand() % (i + 1);
        swap(&numbers[i], &numbers[j]);
    }

    // Fill the cube with the shuffled numbers
    int idx = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                cube[i][j][k] = numbers[idx++];
            }
        }
    }
}

// Evaluate objective function
int evaluate(int cube[N][N][N]) {
    int error = 0;
    int sum;

    // Evaluate rows
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            sum = 0;
            for (int k = 0; k < N; k++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate columns
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
            for (int i = 0; i < N; i++) {
                sum += cube[i][j][k];
            }
```

```
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate pillars
    for (int i = 0; i < N; i++) {
        for (int k = 0; k < N; k++) {
            sum = 0;
            for (int j = 0; j < N; j++) {
                sum += cube[i][j][k];
            }
            error += abs(sum - MAGIC_NUMBER);
        }
    }

    // Evaluate main space diagonals
    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][i];
    }
    error += abs(sum - MAGIC_NUMBER);

    sum = 0;
    for (int i = 0; i < N; i++) {
        sum += cube[i][i][N - i - 1];
    }
    error += abs(sum - MAGIC_NUMBER);

    // Evaluate diagonal in slices
// Evaluate diagonals in horizontal (x-y) slices
    for (int i = 0; i < N; i++) {
        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][j];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int j = 0; j < N; j++) {
            sum += cube[i][j][N - j - 1];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    // Evaluate diagonals in vertical (y-z) slices
    for (int j = 0; j < N; j++) {
```

```
        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[k][j][k];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int k = 0; k < N; k++) {
            sum += cube[N - k - 1][j][k];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    // Evaluate diagonals in vertical (x-z) slices
    for (int k = 0; k < N; k++) {
        sum = 0;
        for (int i = 0; i < N; i++) {
            sum += cube[i][i][k];
        }
        error += abs(sum - MAGIC_NUMBER);

        sum = 0;
        for (int i = 0; i < N; i++) {
            sum += cube[i][N - i - 1][k];
        }
        error += abs(sum - MAGIC_NUMBER);
    }

    return error;
}

// Generate a population with random individuals
void generate_population(Individual population[], int
population_size) {

    // Loop to initialize the population
    for (int i = 0; i < population_size; i++) {
        initialize_cube(population[i].cube);
        population[i].fitness = evaluate(population[i].cube);
    }
}

// Generate individual
int generate_individual(Individual population[], int
population_size) {
```

```
    // Initialize best individual by generating random
individual from population
    int best_individual = rand() % population_size;

    // Loop to find best individual
    for (int i = 1; i < 5; i++) {

        // Initialize competitor individiual by generating
another random individual from population
        int competitor_individual = rand() % population_size;

        // Check if competitor individual's fitness is less
than the best individual's fitness
        if (population[competitor_individual].fitness <
population[best_individual].fitness) {
            best_individual = competitor_individual;
        }
    }
    return best_individual;
}

// Do a crossover
void crossover(Individual *parent1, Individual *parent2,
Individual *child) {
    int map[N * N * N] = {0};
    int idx = 0;

    // Initialize child's value with -1
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                child->cube[i][j][k] = -1;
            }
        }
    }

    // Copy a cycle of values from parent1
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {

                // If a child's value is -1
                if (child->cube[i][j][k] == -1) {

                    // Store parent1 & child value in a
temporary variable
```

```
                    int temp = parent1->cube[i][j][k];
                    child->cube[i][j][k] = temp;
                    map[temp - 1] = 1;
                }
            }
        }
    }

    // Fill the rest from parent2 without duplicates
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {

                // If child's value is -1
                if (child->cube[i][j][k] == -1) {

                    // Store parent2's value in a temporary
variable
                    int temp = parent2->cube[i][j][k];


                    if (!map[temp - 1]) {
                        child->cube[i][j][k] = temp;
                        map[temp - 1] = 1;
                    } else {

                        // Find unused value and insert to
the array
                        for (int m = 0; m < N * N * N; m++) {
                            if (!map[m]) {
                                child->cube[i][j][k] = m + 1;
                                map[m] = 1;
                                break;
                            }
                        }
                    }
                }
            }
        }
    }

    // Evaluate child's fitness
    child->fitness = evaluate(child->cube);
}

// Do a mutation
```

```c
void mutation(Individual *individual) {

    // Initialize the mutation rate & total mutation
    double initial_mutation_rate = 0.05;
    int total_mutation = N * N * N * initial_mutation_rate;

    // Loop to do a mutation
    for (int m = 0; m < total_mutation; m++) {

        // Randomize the child
        int i1 = rand() % N, j1 = rand() % N, k1 = rand() %
N;
        int i2 = rand() % N, j2 = rand() % N, k2 = rand() %
N;

        // Swapping two individuals
        int temp = individual->cube[i1][j1][k1];
        individual->cube[i1][j1][k1] =
individual->cube[i2][j2][k2];
        individual->cube[i2][j2][k2] = temp;
    }

    // Evaluate an individual's fitness
    individual->fitness = evaluate(individual->cube);
}

// Find the best individual in the population
int find_best_individual(Individual population[], int
population_size) {

    // Best individual initialization
    int best_individual = 0;

    // Loop to find the best individual
    for (int i = 1; i < population_size; i++) {
        if (population[i].fitness <
population[best_individual].fitness) {
            best_individual = i;
        }
    }
    return best_individual;
}

// Main function
int main() {
    srand(time(NULL));  // Seed the random number generator
```

```
with the current time

    int iterations = 100;        // Set number of iterations
    int population_size = 100;   // Set population size
    int cube[N][N][N];           // Initialize the 5x5x5 cube

    // Initialize 2 individuals with different population
    Individual population[population_size];
    Individual new_population[population_size];

    // Generate population
    generate_population(population, population_size);

    // Initialize iteration counter, timer, and
best_individual
    int counter = 0;
    clock_t start = clock();
    int best_individual = find_best_individual(population,
population_size);

    // Print the initial state
    printf("Initial Cube:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                printf("%3d ",
population[best_individual].cube[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }
    printf("Initial Fitness: %d\n",
population[best_individual].fitness);
    printf("Population Size: %d\n", population_size);
    printf("Iterations: %d\n", iterations);

    // Main loop
    while (counter < iterations) {

        // Loop to generate new population
        #pragma omp parallel for
        for (int i = 0; i < 2; i++) {
            new_population[i] = population[best_individual];
        }
```

```
        // Loop to do crossover & mutation
        #pragma omp parallel for
        for (int i = 2; i < population_size; i++) {

            // Initialize parents
            int parent1_idx = generate_individual(population,
population_size);
            int parent2_idx = generate_individual(population,
population_size);

            // Crossover & mutation
            crossover(&population[parent1_idx],
&population[parent2_idx], &new_population[i]);
            mutation(&new_population[i]);
        }

        // Loop to generate population
        #pragma omp parallel for
        for (int i = 0; i < population_size; i++) {
            population[i] = new_population[i];
        }

        // Find another best individual, and the number of
iteration will increase
        best_individual = find_best_individual(population,
population_size);
        counter++;

        // If fitness of a best individual from a population
reaches zero, the program will stop
        if (population[best_individual].fitness == 0) {
            break;
        }
    }

    // Timer ends
    clock_t end = clock();
    printf("Genetic Algorithm: Iterations=%d, Time=%.2f
seconds, Best Cost=%d\n", counter, (double)(end - start) /
CLOCKS_PER_SEC, population[best_individual].fitness);

    // Generate the final & optimized cube
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                cube[i][j][k] =
```

```
population[best_individual].cube[i][j][k];
                }
            }
        }

    printf("Optimized Cube:\n");
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                printf("%3d ", cube[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }

    return 0;
}
```

C. Hasil Eksperimen dan Analisis

C.1 Hill-Climbing with Sideways Move

C.1.1 Visualisasi

Percobaan 1	
Initial State	Initial Error : 6533

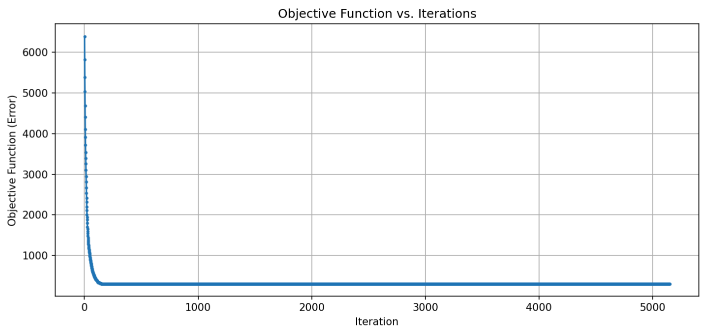
	`` Initial Cube: Slice 1: +-----+-----+-----+-----+-----+ 	39	83	73	107	87 | +-----+-----+-----+-----+-----+ 	51	97	125	48	4 | +-----+-----+-----+-----+-----+ 	98	22	53	50	38 | +-----+-----+-----+-----+-----+ 	61	88	37	30	33 | +-----+-----+-----+-----+-----+ 	12	66	111	99	19 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ 	11	114	47	2	110 | +-----+-----+-----+-----+-----+ 	46	70	63	65	49 | +-----+-----+-----+-----+-----+ 	41	75	92	69	119 | +-----+-----+-----+-----+-----+ 	113	122	101	7	34 | +-----+-----+-----+-----+-----+ 	109	23	80	59	60 | +-----+-----+-----+-----+-----+ ```

```
Slice 3:
    +-----+-----+-----+-----+-----+
    |  43 |  40 |  82 |  67 |   8 |
    +-----+-----+-----+-----+-----+
    |  31 |  17 |  16 |   9 | 121 |
    +-----+-----+-----+-----+-----+
    |  15 |  27 | 116 |  78 |  89 |
    +-----+-----+-----+-----+-----+
    |  42 |  72 |  94 |  57 |  68 |
    +-----+-----+-----+-----+-----+
    |  62 | 103 |  32 | 102 |   1 |
    +-----+-----+-----+-----+-----+

Slice 4:
    +-----+-----+-----+-----+-----+
    | 118 | 120 |  84 | 100 |  25 |
    +-----+-----+-----+-----+-----+
    |  90 |  20 |  95 | 105 |  85 |
    +-----+-----+-----+-----+-----+
    |   3 |  29 |  81 |  28 |  24 |
    +-----+-----+-----+-----+-----+
    | 108 |  64 |  13 |  54 |  55 |
    +-----+-----+-----+-----+-----+
    |  14 | 117 |  35 |  76 |  56 |
    +-----+-----+-----+-----+-----+

Slice 5:
    +-----+-----+-----+-----+-----+
    |  79 | 104 |  93 |   5 |  44 |
    +-----+-----+-----+-----+-----+
    |  58 |  36 | 106 |  10 |  71 |
    +-----+-----+-----+-----+-----+
    |   6 |  52 |  26 |  21 |  74 |
    +-----+-----+-----+-----+-----+
    |  91 |  18 |  96 | 124 |  86 |
    +-----+-----+-----+-----+-----+
    | 115 |  45 | 123 | 112 |  77 |
    +-----+-----+-----+-----+-----+
```

Final State	Iterations : 5172 Final Error : 524 Sideways Move : 5000 Elapsed time : 40 Seconds

```
Slice 1:
    +-----+-----+-----+-----+-----+
    |  18 |  48 |  68 | 107 |  74 |
    +-----+-----+-----+-----+-----+
    |  54 |  94 | 124 |  28 |  13 |
    +-----+-----+-----+-----+-----+
    | 125 |  17 |  79 |  50 |  29 |
    +-----+-----+-----+-----+-----+
    |  62 |  90 |  20 |  35 | 109 |
    +-----+-----+-----+-----+-----+
    |  44 |  66 |  21 |  95 |  89 |
    +-----+-----+-----+-----+-----+

Slice 2:
    +-----+-----+-----+-----+-----+
    |  37 | 116 |   4 |  46 | 112 |
    +-----+-----+-----+-----+-----+
    |  36 |  82 | 106 |  69 |  22 |
    +-----+-----+-----+-----+-----+
    | 121 |  24 |  38 |  77 |  55 |
    +-----+-----+-----+-----+-----+
    |  96 |  70 |  99 |  41 |   9 |
    +-----+-----+-----+-----+-----+
    |  25 |  23 |  91 |  59 | 117 |
    +-----+-----+-----+-----+-----+

Slice 3:
    +-----+-----+-----+-----+-----+
    |  61 |  15 |  81 | 102 |  53 |
    +-----+-----+-----+-----+-----+
    |  52 | 108 |  27 |   8 | 120 |
    +-----+-----+-----+-----+-----+
    |  42 |  31 |  85 |  83 |  76 |
    +-----+-----+-----+-----+-----+
    |  47 |  56 |  75 |  64 |  73 |
    +-----+-----+-----+-----+-----+
    | 114 | 105 |  40 |  58 |   1 |
    +-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td>
<pre>
Slice 4:
    +------+------+------+------+------+
    | 113 | 33 | 80 | 57 | 32 |
    +------+------+------+------+------+
    | 71 | 30 | 19 | 98 | 97 |
    +------+------+------+------+------+
    | 10 | 119 | 87 | 12 | 88 |
    +------+------+------+------+------+
    | 104 | 84 | 11 | 51 | 65 |
    +------+------+------+------+------+
    | 14 | 49 | 118 | 100 | 34 |
    +------+------+------+------+------+

Slice 5:
    +------+------+------+------+------+
    | 86 | 103 | 92 | 3 | 43 |
    +------+------+------+------+------+
    | 101 | 2 | 39 | 111 | 63 |
    +------+------+------+------+------+
    | 7 | 122 | 26 | 93 | 67 |
    +------+------+------+------+------+
    | 6 | 16 | 110 | 123 | 60 |
    +------+------+------+------+------+
    | 115 | 72 | 45 | 5 | 78 |
    +------+------+------+------+------+
</pre>
</td></tr>
<tr><td>Hasil</td><td>


</td></tr>
<tr><td colspan="2">Percobaan 2</td></tr>
<tr><td>Inisial</td><td>Initial Error : 6799</td></tr>
</table>

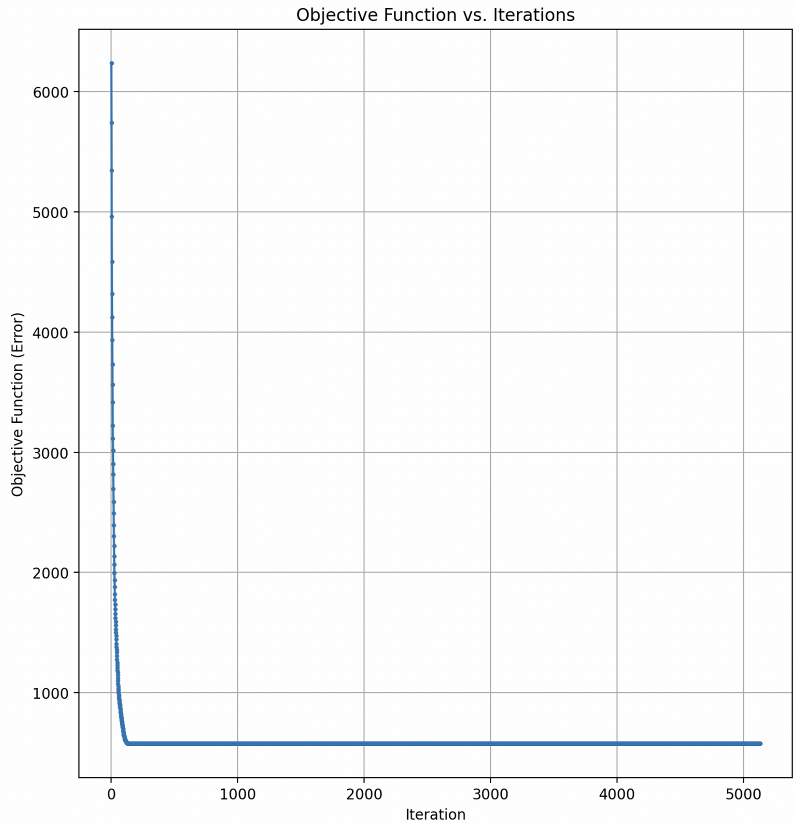
```
Initial Cube:
Slice 1:
    +------+------+------+------+------+
    |   12 |   93 |   87 |   48 |   78 |
    +------+------+------+------+------+
    |   38 |   10 |   68 |   40 |   49 |
    +------+------+------+------+------+
    |   92 |    4 |   79 |   65 |   15 |
    +------+------+------+------+------+
    |   11 |   17 |   75 |   27 |  113 |
    +------+------+------+------+------+
    |   91 |  115 |  103 |  124 |   30 |
    +------+------+------+------+------+

Slice 2:
    +------+------+------+------+------+
    |  116 |  114 |   26 |   13 |   21 |
    +------+------+------+------+------+
    |  100 |   28 |   90 |   95 |    3 |
    +------+------+------+------+------+
    |   63 |   45 |   44 |   52 |  101 |
    +------+------+------+------+------+
    |   66 |    5 |   74 |    7 |   41 |
    +------+------+------+------+------+
    |  125 |   14 |   84 |  106 |   46 |
    +------+------+------+------+------+

Slice 3:
    +------+------+------+------+------+
    |   22 |  104 |  122 |   81 |   67 |
    +------+------+------+------+------+
    |   36 |   72 |  110 |   86 |   25 |
    +------+------+------+------+------+
    |  105 |   89 |    9 |   56 |   51 |
    +------+------+------+------+------+
    |   88 |   97 |   37 |    1 |   55 |
    +------+------+------+------+------+
    |  118 |  119 |   77 |   47 |   73 |
    +------+------+------+------+------+
```

<table>
<tr>
<td></td>
<td>
<pre>
Slice 4:
    +-----+-----+-----+-----+-----+
    | 39 | 16 | 94 | 60 | 34 |
    +-----+-----+-----+-----+-----+
    | 121 | 42 | 2 | 59 | 62 |
    +-----+-----+-----+-----+-----+
    | 29 | 31 | 76 | 98 | 19 |
    +-----+-----+-----+-----+-----+
    | 6 | 117 | 35 | 53 | 18 |
    +-----+-----+-----+-----+-----+
    | 54 | 99 | 102 | 33 | 107 |
    +-----+-----+-----+-----+-----+

Slice 5:
    +-----+-----+-----+-----+-----+
    | 83 | 112 | 8 | 109 | 82 |
    +-----+-----+-----+-----+-----+
    | 80 | 85 | 61 | 111 | 43 |
    +-----+-----+-----+-----+-----+
    | 64 | 23 | 58 | 50 | 57 |
    +-----+-----+-----+-----+-----+
    | 123 | 71 | 70 | 120 | 24 |
    +-----+-----+-----+-----+-----+
    | 69 | 32 | 20 | 96 | 108 |
    +-----+-----+-----+-----+-----+
</pre>
</td>
</tr>
<tr>
<td>Final</td>
<td>Final Error : 577
Iterations : 5130
Execution time : 38.02 Detik
Sideways : 5000</td>
</tr>
</table>

```
Slice 1:
    +------+------+------+------+------+
    |    8 |   82 |   87 |   64 |   72 |
    +------+------+------+------+------+
    |   38 |  122 |   45 |   66 |   41 |
    +------+------+------+------+------+
    |   97 |    1 |  102 |   30 |  103 |
    +------+------+------+------+------+
    |  121 |    3 |   74 |   47 |   68 |
    +------+------+------+------+------+
    |   56 |  108 |   11 |  109 |   31 |
    +------+------+------+------+------+

Slice 2:
    +------+------+------+------+------+
    |  116 |  104 |    2 |   69 |   26 |
    +------+------+------+------+------+
    |   51 |   78 |   90 |   79 |   17 |
    +------+------+------+------+------+
    |    7 |   54 |   76 |   52 |  124 |
    +------+------+------+------+------+
    |   21 |   44 |  125 |    9 |  111 |
    +------+------+------+------+------+
    |  120 |   35 |   20 |  106 |   36 |
    +------+------+------+------+------+

Slice 3:
    +------+------+------+------+------+
    |   88 |    6 |   91 |   15 |  115 |
    +------+------+------+------+------+
    |   27 |   16 |  101 |   84 |   98 |
    +------+------+------+------+------+
    |  123 |   86 |   18 |   50 |   14 |
    +------+------+------+------+------+
    |   65 |   92 |   28 |  118 |   12 |
    +------+------+------+------+------+
    |    4 |  113 |   77 |   46 |   75 |
    +------+------+------+------+------+
```

	`` Slice 4: +-----+-----+-----+-----+-----+ 	81	57	96	49	32 | +-----+-----+-----+-----+-----+ 	119	10	37	40	107 | +-----+-----+-----+-----+-----+ 	29	110	63	94	19 | +-----+-----+-----+-----+-----+ 	13	105	5	99	100 | +-----+-----+-----+-----+-----+ 	73	34	114	33	61 | +-----+-----+-----+-----+-----+ Slice 5: +-----+-----+-----+-----+-----+ 	22	67	39	117	70 | +-----+-----+-----+-----+-----+ 	80	89	43	48	55 | +-----+-----+-----+-----+-----+ 	59	60	58	85	53 | +-----+-----+-----+-----+-----+ 	95	71	83	42	24 | +-----+-----+-----+-----+-----+ 	62	25	93	23	112 | +-----+-----+-----+-----+-----+ ```

Visualisasi	
Percobaan 3	
Inisial	**Initial Error : 6797**

	Slice 1: (see below)

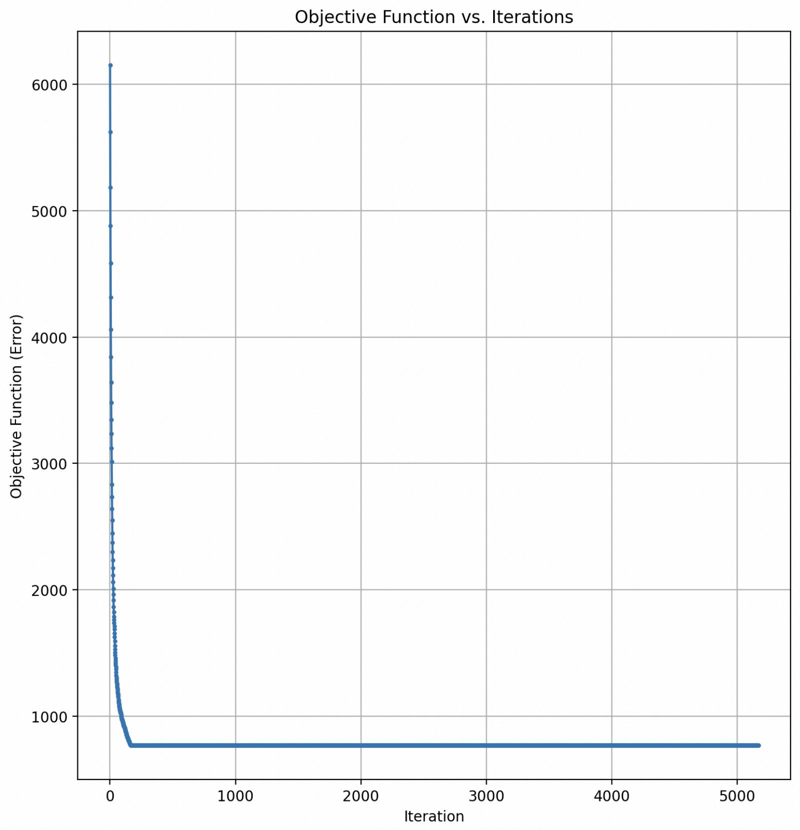
```
Slice 1:
    +-----+-----+-----+-----+-----+
    |  79 |  53 |  24 |  28 | 123 |
    +-----+-----+-----+-----+-----+
    |  31 |  99 | 118 |  64 |  95 |
    +-----+-----+-----+-----+-----+
    |  65 |   7 |  76 | 121 |  46 |
    +-----+-----+-----+-----+-----+
    |  14 |  61 |  59 |  44 |  57 |
    +-----+-----+-----+-----+-----+
    |  22 |  97 |  90 |  88 |  20 |
    +-----+-----+-----+-----+-----+

Slice 2:
    +-----+-----+-----+-----+-----+
    |  72 |  55 |  87 |  96 |  56 |
    +-----+-----+-----+-----+-----+
    |  77 |  40 |  83 | 100 |   1 |
    +-----+-----+-----+-----+-----+
    |  74 |  91 |  66 |   4 |  50 |
    +-----+-----+-----+-----+-----+
    | 110 |  19 |  47 |  75 | 119 |
    +-----+-----+-----+-----+-----+
    |  30 |  86 |  58 | 122 |  52 |
    +-----+-----+-----+-----+-----+

Slice 3:
    +-----+-----+-----+-----+-----+
    |  78 |   6 |  43 |  26 |  39 |
    +-----+-----+-----+-----+-----+
    | 117 |  82 | 102 | 105 |   2 |
    +-----+-----+-----+-----+-----+
    |  93 | 114 |  84 |  62 | 111 |
    +-----+-----+-----+-----+-----+
    | 124 |  60 |  67 |  13 | 115 |
    +-----+-----+-----+-----+-----+
    |   9 |  71 |  45 |  42 |  33 |
    +-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td><pre>
Slice 4:
    +------+------+------+------+------+
    | 63 | 32 | 25 | 41 | 29 |
    +------+------+------+------+------+
    | 70 | 109 | 106 | 101 | 21 |
    +------+------+------+------+------+
    | 15 | 37 | 69 | 34 | 3 |
    +------+------+------+------+------+
    | 51 | 27 | 85 | 23 | 81 |
    +------+------+------+------+------+
    | 68 | 103 | 104 | 36 | 10 |
    +------+------+------+------+------+

Slice 5:
    +------+------+------+------+------+
    | 92 | 5 | 108 | 112 | 73 |
    +------+------+------+------+------+
    | 125 | 48 | 89 | 107 | 54 |
    +------+------+------+------+------+
    | 35 | 8 | 120 | 113 | 38 |
    +------+------+------+------+------+
    | 16 | 98 | 116 | 18 | 11 |
    +------+------+------+------+------+
    | 17 | 94 | 12 | 80 | 49 |
    +------+------+------+------+------+
</pre></td></tr>
<tr><td>Final</td><td>Final Error : 768
Iterations : 5168
Sideways Move : 5000
Execution time : 40.45</td></tr>
</table>

```
Slice 1:
    +-----+-----+-----+-----+-----+
    |  79 |  45 |  60 |  35 |  96 |
    +-----+-----+-----+-----+-----+
    |  47 |  70 |  56 |  32 | 110 |
    +-----+-----+-----+-----+-----+
    |  62 |   6 |  72 | 123 |  52 |
    +-----+-----+-----+-----+-----+
    | 105 |  93 |   2 |  82 |  38 |
    +-----+-----+-----+-----+-----+
    |  22 | 101 | 125 |  43 |  19 |
    +-----+-----+-----+-----+-----+

Slice 2:
    +-----+-----+-----+-----+-----+
    |  63 |  94 |  86 |  17 |  55 |
    +-----+-----+-----+-----+-----+
    |  59 |  37 |  87 |  88 |  44 |
    +-----+-----+-----+-----+-----+
    |  98 |  80 |  78 |   9 |  50 |
    +-----+-----+-----+-----+-----+
    |  28 |  21 |  61 |  89 | 115 |
    +-----+-----+-----+-----+-----+
    |  67 |  85 |   5 | 112 |  49 |
    +-----+-----+-----+-----+-----+

Slice 3:
    +-----+-----+-----+-----+-----+
    |   8 | 117 |  36 | 121 |  33 |
    +-----+-----+-----+-----+-----+
    |  15 |  92 |  81 | 107 |  20 |
    +-----+-----+-----+-----+-----+
    |  95 |  97 |  75 |   7 |  40 |
    +-----+-----+-----+-----+-----+
    | 113 |  18 |  54 |  27 | 103 |
    +-----+-----+-----+-----+-----+
    |  84 |   1 |  69 |  53 | 118 |
    +-----+-----+-----+-----+-----+
```

	<pre>Slice 4: +------+------+------+------+------+ | 66 | 34 | 108 | 41 | 68 | +------+------+------+------+------+ | 71 | 74 | 3 | 90 | 77 | +------+------+------+------+------+ | 14 | 120 | 16 | 46 | 119 | +------+------+------+------+------+ | 51 | 30 | 83 | 109 | 39 | +------+------+------+------+------+ | 111 | 57 | 106 | 29 | 12 | +------+------+------+------+------+ Slice 5: +------+------+------+------+------+ | 99 | 24 | 25 | 102 | 65 | +------+------+------+------+------+ | 114 | 42 | 91 | 4 | 64 | +------+------+------+------+------+ | 48 | 13 | 73 | 122 | 58 | +------+------+------+------+------+ | 23 | 124 | 116 | 11 | 26 | +------+------+------+------+------+ | 31 | 104 | 10 | 76 | 100 | +------+------+------+------+------+</pre>

Visualisasi	

C.1.2 Hasil Analisis

1. Seberapa dekat tiap-tiap algoritma bisa mendekati global optima dan mengapa hasilnya demikian?

Algoritma tersebut hanya berhasil mencapai fitness error dalam range 300-600an. Hal ini disebabkan oleh deep plateau. Hal ini menyebabkan algoritma “nyangkut” dan tidak bisa kembali hill climb, meskipun telah bergeser dengan maksimal 5000 kali.

2. Bagaimana perbandingan hasil pencarian tiap-tiap algoritma dengan algoritma local search yang lain?

Algoritma ini memiliki final fitness score yang masih kurang baik, hal ini disebabkan oleh ketidakmampuan mereka untuk keluar dari plateau. Meski mereka berhasil dalam keluar dari shallow plateau, algoritma ini tidak cukup fleksibel layaknya algoritma lainya seperti SA dan GA.

3. Bagaimana perbandingan durasi proses pencarian tiap algoritma relatif terhadap algoritma lainnya?

Durasi pencarian sideways move sangat lama jika dibandingkan dengan algoritma lainnya. Hal ini disebabkan oleh kewajiban sideways untuk men generate all possible neighbors dari suatu titik sebelum menentukan ke yang lain. Rata rata waktu yang dibutuhkan sebesar 40 detik.

4. **Seberapa konsisten hasil akhir yang didapatkan dari tiap-tiap eksperimen yang dilakukan?**

Konsistensi hasil akhir cukup fluktuatif mengingat performa final error sangat dependen dari starting cube algoritma tersebut. Oleh karena itu, nilai dari final error berkisar di 300-1200 fitness score.

5. **Bagaimana *scalability* dari algoritma ketika diterapkan pada data yang jauh lebih besar?**

Waktu dan sumber daya yang dibutuhkan cenderung meningkat secara eksponensial atau polinomial tergantung pada kompleksitas masalah dan jumlah langkah yang diperlukan untuk mencapai solusi optimal atau mendekati optimal. Karena algoritma ini mengeksplorasi seluruh ruang tetangga untuk setiap langkah dan sering kali harus mengulang jika menemukan local optimum, biaya komputasinya meningkat tajam seiring dengan bertambahnya jumlah variabel atau ukuran ruang pencarian.

6. **Bagaimana efisiensi memori algoritma?**

Efisiensi memori dalam algoritma sideways move hill climbing tergantung pada representasi solusi dan jumlah tetangga yang dievaluasi. Karena algoritma ini hanya mempertahankan satu solusi pada satu waktu serta sejumlah terbatas tetangga, penggunaan memori umumnya rendah, cukup untuk menyimpan solusi dan tetangga terbaik sementara. Namun, untuk masalah skala besar, memori bisa menjadi faktor pembatas terutama jika jumlah tetangga sangat besar atau jika penyimpanan solusi kompleks membutuhkan lebih banyak ruang.

C.2 Simulated Annealing

C.2.1 Visualisasi

Percobaan 1	
Initial State	Initial Error: 6509

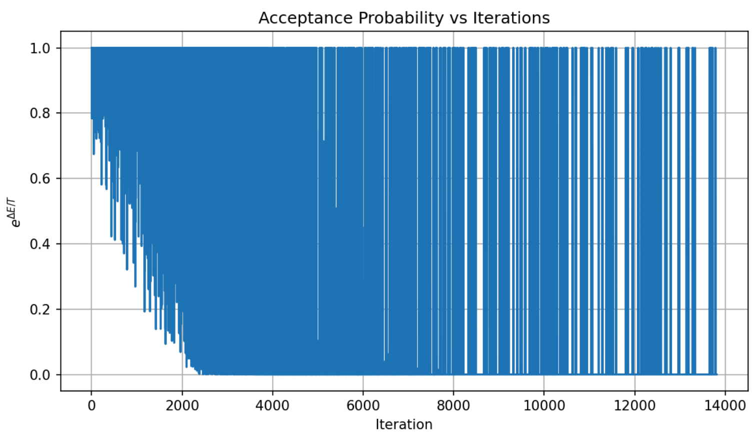
```
Initial Cube:
Slice 1:
    +-----+-----+-----+-----+-----+
    |  73 |  18 | 114 |  39 |  11 |
    +-----+-----+-----+-----+-----+
    |  69 |  63 | 110 |  23 |  56 |
    +-----+-----+-----+-----+-----+
    |  44 |   4 |  93 |  17 |  98 |
    +-----+-----+-----+-----+-----+
    |  14 |  96 |  12 |  36 | 109 |
    +-----+-----+-----+-----+-----+
    |  34 |   8 |  88 | 120 | 121 |
    +-----+-----+-----+-----+-----+

Slice 2:
    +-----+-----+-----+-----+-----+
    |  53 |  72 | 104 |   7 |  46 |
    +-----+-----+-----+-----+-----+
    | 123 |  55 | 115 | 101 |  97 |
    +-----+-----+-----+-----+-----+
    |  22 |  91 | 125 | 112 |  43 |
    +-----+-----+-----+-----+-----+
    |  61 |  77 | 102 |  10 | 105 |
    +-----+-----+-----+-----+-----+
    |  67 |  65 |  95 |  51 |  42 |
    +-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td><pre>
Slice 4:
  +-----+-----+-----+-----+-----+
  | 76 | 124 | 79 | 123 | 81 |
  +-----+-----+-----+-----+-----+
  | 117 | 62 | 68 | 16 | 111 |
  +-----+-----+-----+-----+-----+
  | 38 | 33 | 102 | 64 | 44 |
  +-----+-----+-----+-----+-----+
  | 25 | 73 | 66 | 107 | 14 |
  +-----+-----+-----+-----+-----+
  | 29 | 53 | 115 | 92 | 34 |
  +-----+-----+-----+-----+-----+

Slice 5:
  +-----+-----+-----+-----+-----+
  | 24 | 125 | 103 | 60 | 63 |
  +-----+-----+-----+-----+-----+
  | 9 | 7 | 120 | 71 | 1 |
  +-----+-----+-----+-----+-----+
  | 6 | 87 | 22 | 110 | 59 |
  +-----+-----+-----+-----+-----+
  | 106 | 104 | 78 | 80 | 99 |
  +-----+-----+-----+-----+-----+
  | 94 | 17 | 54 | 57 | 35 |
  +-----+-----+-----+-----+-----+
</pre></td></tr>
<tr><td>Final State</td><td>Iteration: 13809
Final Error: 576
Total Stuck Occurrence: 13
Execution time: 0.06 seconds</td></tr>
</table>

```
Slice 1:
   +-----+-----+-----+-----+-----+
   |  93 | 114 |  16 |  24 |  69 |
   +-----+-----+-----+-----+-----+
   | 102 |  44 |  38 |  73 |  58 |
   +-----+-----+-----+-----+-----+
   |   9 | 121 |  71 |  15 |  96 |
   +-----+-----+-----+-----+-----+
   |  31 |  23 |  77 |  95 |  84 |
   +-----+-----+-----+-----+-----+
   |  81 |   4 | 110 | 108 |  13 |
   +-----+-----+-----+-----+-----+

Slice 2:
   +-----+-----+-----+-----+-----+
   |  66 |   2 | 115 |  89 |  45 |
   +-----+-----+-----+-----+-----+
   |  22 |  74 | 106 | 105 |  10 |
   +-----+-----+-----+-----+-----+
   | 100 |  91 |  26 |   1 | 104 |
   +-----+-----+-----+-----+-----+
   |  99 | 116 |  47 |  18 |  35 |
   +-----+-----+-----+-----+-----+
   |  28 |  36 |  21 | 113 | 120 |
   +-----+-----+-----+-----+-----+

Slice 3:
   +-----+-----+-----+-----+-----+
   |  65 |  75 |  53 |  85 |  37 |
   +-----+-----+-----+-----+-----+
   | 109 |  83 |   5 |  50 |  61 |
   +-----+-----+-----+-----+-----+
   |  41 |   3 |  68 | 124 |  79 |
   +-----+-----+-----+-----+-----+
   |   6 |  62 | 123 |  43 |  82 |
   +-----+-----+-----+-----+-----+
   |  94 |  92 |  63 |  19 |  55 |
   +-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td><pre>
Slice 4:
    +-----+-----+-----+-----+-----+
    | 20 | 40 | 118 | 78 | 60 |
    +-----+-----+-----+-----+-----+
    | 30 | 101 | 80 | 27 | 76 |
    +-----+-----+-----+-----+-----+
    | 49 | 33 | 98 | 117 | 7 |
    +-----+-----+-----+-----+-----+
    | 122 | 42 | 14 | 34 | 103 |
    +-----+-----+-----+-----+-----+
    | 88 | 97 | 8 | 54 | 70 |
    +-----+-----+-----+-----+-----+

Slice 5:
    +-----+-----+-----+-----+-----+
    | 64 | 90 | 12 | 39 | 107 |
    +-----+-----+-----+-----+-----+
    | 46 | 17 | 86 | 59 | 111 |
    +-----+-----+-----+-----+-----+
    | 119 | 52 | 51 | 67 | 29 |
    +-----+-----+-----+-----+-----+
    | 48 | 72 | 56 | 125 | 11 |
    +-----+-----+-----+-----+-----+
    | 32 | 87 | 112 | 25 | 57 |
    +-----+-----+-----+-----+-----+
</pre></td></tr>
<tr><td>Plot</td><td>
</td></tr>
<tr><td colspan="2">Percobaan 2</td></tr>
<tr><td>Initial State</td><td>Initial Error : 7117</td></tr>
</table>

	(console output below)

```
Initial Cube:
Slice 1:
   +-----+-----+-----+-----+-----+
   |  11 |  21 |  71 |  23 | 110 |
   +-----+-----+-----+-----+-----+
   |  99 | 103 |  39 |  56 |   6 |
   +-----+-----+-----+-----+-----+
   |   7 |  98 |  82 |  57 |  27 |
   +-----+-----+-----+-----+-----+
   |  86 |  41 |  95 |  26 |   3 |
   +-----+-----+-----+-----+-----+
   |  28 | 124 |  59 |   4 |  14 |
   +-----+-----+-----+-----+-----+

Slice 2:
   +-----+-----+-----+-----+-----+
   |  54 |  47 |  62 |  18 | 108 |
   +-----+-----+-----+-----+-----+
   |  42 | 119 |  34 |  52 |  75 |
   +-----+-----+-----+-----+-----+
   |  85 |   2 |  88 |  58 |  72 |
   +-----+-----+-----+-----+-----+
   |  96 |  45 |  20 |  73 |  29 |
   +-----+-----+-----+-----+-----+
   | 123 |  90 | 109 |  50 |  77 |
   +-----+-----+-----+-----+-----+

Slice 3:
   +-----+-----+-----+-----+-----+
   |  16 | 121 |  83 |  76 |  91 |
   +-----+-----+-----+-----+-----+
   |   5 | 125 | 105 |  32 |  87 |
   +-----+-----+-----+-----+-----+
   |  31 | 117 |  51 |  80 |  63 |
   +-----+-----+-----+-----+-----+
   |  60 |  19 |  13 |  92 |  79 |
   +-----+-----+-----+-----+-----+
   |  97 | 116 |  24 |  61 | 120 |
   +-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td>
<pre>
Slice 4:
   +-----+-----+-----+-----+-----+
   | 55 | 49 | 64 | 122 | 38 |
   +-----+-----+-----+-----+-----+
   | 44 | 118 | 43 | 68 | 30 |
   +-----+-----+-----+-----+-----+
   | 22 | 37 | 67 | 93 | 100 |
   +-----+-----+-----+-----+-----+
   | 46 | 104 | 114 | 53 | 33 |
   +-----+-----+-----+-----+-----+
   | 12 | 66 | 65 | 81 | 15 |
   +-----+-----+-----+-----+-----+

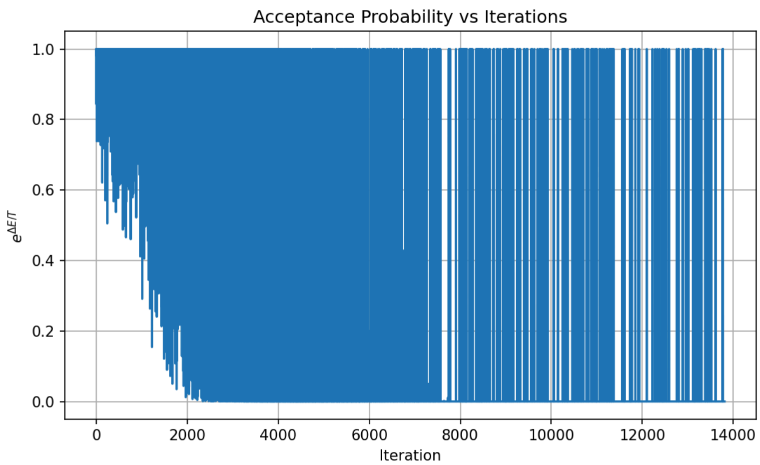
Slice 5:
   +-----+-----+-----+-----+-----+
   | 48 | 74 | 1 | 35 | 36 |
   +-----+-----+-----+-----+-----+
   | 17 | 69 | 94 | 8 | 84 |
   +-----+-----+-----+-----+-----+
   | 101 | 107 | 113 | 25 | 40 |
   +-----+-----+-----+-----+-----+
   | 10 | 78 | 112 | 115 | 102 |
   +-----+-----+-----+-----+-----+
   | 111 | 89 | 106 | 9 | 70 |
   +-----+-----+-----+-----+-----+
</pre>
</td>
</tr>
<tr>
<td>Final State</td>
<td>Final Error: 515
Total Stuck Occurrences: 11
Iteration: 13809
Execution Time: 0.07 seconds</td>
</tr>
</table>

	`` Slice 1: ```

```
Slice 1:
    +-----+-----+-----+-----+-----+
    |  14 | 117 |  46 |  73 |  63 |
    +-----+-----+-----+-----+-----+
    | 124 |  59 |  21 | 105 |   8 |
    +-----+-----+-----+-----+-----+
    |  78 |  85 |  55 |   2 |  97 |
    +-----+-----+-----+-----+-----+
    |  83 |  47 |  92 |  67 |  28 |
    +-----+-----+-----+-----+-----+
    |  16 |   9 | 102 |  71 | 121 |
    +-----+-----+-----+-----+-----+

Slice 2:
    +-----+-----+-----+-----+-----+
    |  48 |  95 |  74 |  30 |  64 |
    +-----+-----+-----+-----+-----+
    |  61 |  49 |  23 | 104 |  77 |
    +-----+-----+-----+-----+-----+
    |  41 |  42 |  75 |  54 | 106 |
    +-----+-----+-----+-----+-----+
    | 103 |  17 |  53 | 116 |  29 |
    +-----+-----+-----+-----+-----+
    |  70 | 114 |  86 |   6 |  33 |
    +-----+-----+-----+-----+-----+

Slice 3:
    +-----+-----+-----+-----+-----+
    |  66 |   7 | 123 |  90 |  19 |
    +-----+-----+-----+-----+-----+
    |  72 |   4 |  58 |  82 | 100 |
    +-----+-----+-----+-----+-----+
    | 110 | 101 |  44 |  15 |  45 |
    +-----+-----+-----+-----+-----+
    |  12 | 119 |  60 |  87 |  38 |
    +-----+-----+-----+-----+-----+
    |  52 |  88 |  27 |  40 | 113 |
    +-----+-----+-----+-----+-----+
```

```
Slice 4:
    +-----+-----+-----+-----+-----+
    | 107 |  20 |  39 |  35 | 115 |
    +-----+-----+-----+-----+-----+
    |  25 | 112 |  93 |  18 |  68 |
    +-----+-----+-----+-----+-----+
    |  31 |  69 |  57 | 120 |  37 |
    +-----+-----+-----+-----+-----+
    |  51 |  26 | 111 |  34 |  98 |
    +-----+-----+-----+-----+-----+
    |  99 |  91 |  11 | 108 |   5 |
    +-----+-----+-----+-----+-----+

Slice 5:
    +-----+-----+-----+-----+-----+
    |  80 |  76 |  24 |  84 |  50 |
    +-----+-----+-----+-----+-----+
    |  36 |  96 | 118 |   3 |  65 |
    +-----+-----+-----+-----+-----+
    |  56 |  22 |  81 | 122 |  32 |
    +-----+-----+-----+-----+-----+
    |  62 | 109 |   1 |  10 | 125 |
    +-----+-----+-----+-----+-----+
    |  79 |  13 |  89 |  94 |  43 |
    +-----+-----+-----+-----+-----+
```

Plot



Percobaan 3

Initial State

Initial Error: 6335

```
Slice 1:
   +-----+-----+-----+-----+-----+
   | 108 |  53 |  68 |  99 | 122 |
   +-----+-----+-----+-----+-----+
   | 111 |  13 | 120 | 100 |   7 |
   +-----+-----+-----+-----+-----+
   | 107 |  56 | 102 | 109 |   6 |
   +-----+-----+-----+-----+-----+
   |  15 |  94 |  93 |  69 |  90 |
   +-----+-----+-----+-----+-----+
   |  27 | 125 |   1 |  44 |  30 |
   +-----+-----+-----+-----+-----+

Slice 2:
   +-----+-----+-----+-----+-----+
   |  71 |  72 |  54 |   8 |  84 |
   +-----+-----+-----+-----+-----+
   |   5 |  31 | 119 |  67 |  86 |
   +-----+-----+-----+-----+-----+
   | 117 |  10 |  42 |  61 |  20 |
   +-----+-----+-----+-----+-----+
   |  59 |  45 |  22 |  63 |  49 |
   +-----+-----+-----+-----+-----+
   |  82 |  26 |  70 |  92 |  78 |
   +-----+-----+-----+-----+-----+

Slice 3:
   +-----+-----+-----+-----+-----+
   | 124 |  47 | 103 |  34 |  73 |
   +-----+-----+-----+-----+-----+
   | 123 |  41 |   9 |   4 |  12 |
   +-----+-----+-----+-----+-----+
   |  16 |   3 |  75 |  65 |  80 |
   +-----+-----+-----+-----+-----+
   | 106 |  77 | 116 |  24 |  50 |
   +-----+-----+-----+-----+-----+
   |  48 |  14 |  58 | 115 |  79 |
   +-----+-----+-----+-----+-----+
```

```
Slice 4:
   +-----+-----+-----+-----+-----+
   | 105 |  62 |  55 |  91 |  57 |
   +-----+-----+-----+-----+-----+
   |  46 | 118 |  25 |  43 |  35 |
   +-----+-----+-----+-----+-----+
   |  29 | 110 |  66 |  17 |  96 |
   +-----+-----+-----+-----+-----+
   |  36 |  52 |  76 |  40 |  19 |
   +-----+-----+-----+-----+-----+
   |  85 | 101 |  60 |  11 |  97 |
   +-----+-----+-----+-----+-----+

Slice 5:
   +-----+-----+-----+-----+-----+
   | 114 |  74 | 113 |  88 |  33 |
   +-----+-----+-----+-----+-----+
   |  37 |  64 |  89 |   2 |  28 |
   +-----+-----+-----+-----+-----+
   |  95 | 121 | 112 |  51 | 104 |
   +-----+-----+-----+-----+-----+
   |  87 |  39 |  98 |  21 |  81 |
   +-----+-----+-----+-----+-----+
   |  38 |  83 |  23 |  18 |  32 |
   +-----+-----+-----+-----+-----+
```

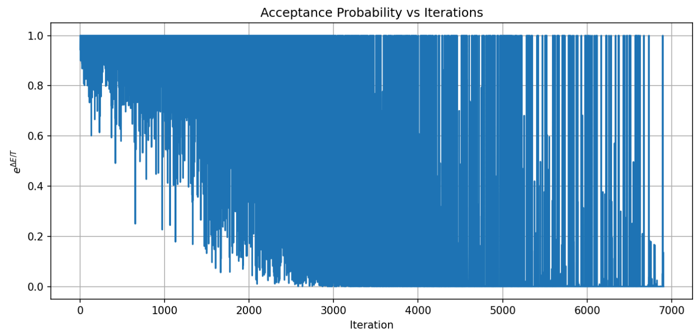
Final State	Iteration: 13809 Final Error: 690 Total Stuck Occurrences: 12 Execution time: 0.07

```
Slice 1:
   +-----+-----+-----+-----+-----+
   | 116 |   8 |  36 |  93 |  57 |
   +-----+-----+-----+-----+-----+
   |  48 |  32 |  98 |  15 | 120 |
   +-----+-----+-----+-----+-----+
   |  51 |  90 |  58 | 106 |   6 |
   +-----+-----+-----+-----+-----+
   |  28 | 121 |  54 |  37 |  66 |
   +-----+-----+-----+-----+-----+
   |  39 |  56 |  84 |  64 |  71 |
   +-----+-----+-----+-----+-----+

Slice 2:
   +-----+-----+-----+-----+-----+
   | 113 |   2 | 119 |  70 |  18 |
   +-----+-----+-----+-----+-----+
   |  26 |  79 |  65 |  95 |  52 |
   +-----+-----+-----+-----+-----+
   |  30 |  75 |  21 |  92 |  96 |
   +-----+-----+-----+-----+-----+
   |  72 | 100 |   1 |  46 | 101 |
   +-----+-----+-----+-----+-----+
   |  82 |  63 | 112 |   9 |  47 |
   +-----+-----+-----+-----+-----+

Slice 3:
   +-----+-----+-----+-----+-----+
   |  10 | 108 |  41 |  42 | 114 |
   +-----+-----+-----+-----+-----+
   | 104 |  67 |  73 |   4 |  80 |
   +-----+-----+-----+-----+-----+
   |  50 |  69 |  68 |  40 |  89 |
   +-----+-----+-----+-----+-----+
   |  88 |  44 |  49 | 125 |   7 |
   +-----+-----+-----+-----+-----+
   |  86 |  20 |  83 | 103 |  33 |
   +-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td>

```
Slice 4:
    +-----+-----+-----+-----+-----+
    |  43 | 118 |  16 |  38 |  99 |
    +-----+-----+-----+-----+-----+
    | 123 |  14 |  60 |  87 |  29 |
    +-----+-----+-----+-----+-----+
    | 111 |  78 |  91 |  17 |  19 |
    +-----+-----+-----+-----+-----+
    |  23 |  45 | 124 |  53 |  62 |
    +-----+-----+-----+-----+-----+
    |  11 |  61 |  24 | 110 | 109 |
    +-----+-----+-----+-----+-----+

Slice 5:
    +-----+-----+-----+-----+-----+
    |  31 |  76 | 102 |  81 |  27 |
    +-----+-----+-----+-----+-----+
    |  12 | 122 |  22 | 117 |  35 |
    +-----+-----+-----+-----+-----+
    |  74 |   5 |  77 |  59 | 107 |
    +-----+-----+-----+-----+-----+
    | 105 |   3 |  97 |  25 |  85 |
    +-----+-----+-----+-----+-----+
    |  94 | 115 |  13 |  34 |  55 |
    +-----+-----+-----+-----+-----+
```

</td>
</tr>
<tr>
<td>Plot</td>
<td>



</td>
</tr>
</table>

C.2.2 Hasil Analisis

1. **Analisis seberapa dekat *simulated annealing algorithm* dapat mendekati global optima dan faktor yang mempengaruhinya**

   Berdasarkan percobaan, Simulated annealing memiliki range gap error dengan global optima sebesar 515-690. Hal ini disebabkan oleh faktor pembatasan pada temperatur awal dan temperatur akhir pada algoritma, serta laju penurunan suhu.

2. **Bagaimana perbandingan hasil pencarian tiap-tiap algoritma dengan algoritma local search yang lain?**

   Berdasarkan hasil objective function yang didapatkan, simulated annealing memiliki hasil kedua terbaik dibandingkan dengan sideway moves dan juga genetic. Hal ini dapat disebabkan oleh beberapa variabel pendukung, perbedaan iterasi, dan juga fungsi dari algoritma masing-masing local search yang berbeda.

3. **Perbandingan durasi proses pencarian *simulated annealing* terhadap algoritma lainnya**

   Berdasarkan durasi pencarian, simulated annealing memiliki waktu eksekusi yang sangat cepat, yaitu direntang 0.06-0.07. Dengan skor tersebut, simulated annealing mendominasi kedua algoritma lainnya yang memiliki durasi proses hingga 50 detik.

4. **Seberapa konsisten hasil akhir yang didapatkan dari tiap-tiap eksperimen yang dilakukan?**

   Konsistensi yang didapatkan dari 3 percobaan simulated annealing menunjukan konsistensi yang cukup baik sebesar 576, 515, dan 690 melalui iterasi yang sama yaitu pada 13890.

5. **Bagaimana efisiensi memori algoritma?**

   Efisiensi algoritma yang digunakan hanya menyangkut variabel tambahan seperti temperature, iteration, dll. Secara keseluruhan memori yang digunakan berskala O(N^3).

6. **Analisis batasan memori pada *simulated annealing* yang untuk masalah skala besar**

Memori yang dibutuhkan oleh algoritma tersebut cukup kecil. Namun, mempertimbangkan ukuran *magic cube* yang besar, seperti ribuan, hal ini dapat menyebabkan keterbatasan pada komputer yang mengeksekusi algoritma ini

7. **Bagaimana tingkat *scalability* dari algoritma ketika diterapkan pada data yang jauh lebih besar?**

   Apabila dihadapkan dengan data yang lebih besar, simulated annealing membutuhkan jumlah iterasi yang lebih besar dan luas untuk mencapai hasil yang lebih optimal. Lalu, Cooling schedule yang dibutuhkan akan lebih lambat agar proses menjadi lebih efektif.

8. **Pengaruh peningkatan waktu dan sumber daya secara linear, eksponensial, dan polinomial terhadap peningkatan kompleksitas masalah**

   Algoritma ini membutuhkan waktu dan sumber daya yang cenderung meningkat secara polinomial atau bahkan eksponensial dengan pertimbangan evaluasi fungsi tujuan dengan kompleksitas yang ada serta jumlah iterasi yang meningkat seiring peningkatan skala yang dilakukan.

C.3 Genetic Algorithm

C.3.1 Visualisasi

Variasi 1 (Population = 1000)		
Iterations	**Percobaan**	**Hasil**
5000	1	Initial State Initial Fitness: 3920

```
Slice 1:
+-----+-----+-----+-----+-----+
|  81 |  43 | 121 |   2 |  37 |
+-----+-----+-----+-----+-----+
|   8 |  44 |  35 |  46 |  17 |
+-----+-----+-----+-----+-----+
|  58 |  23 |   3 |  31 |  65 |
+-----+-----+-----+-----+-----+
|  13 |   4 | 115 |  41 |  48 |
+-----+-----+-----+-----+-----+
|  25 |  29 |  22 |  15 |  84 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  39 |  40 |  14 |  80 |  62 |
+-----+-----+-----+-----+-----+
|  47 |  50 |  96 |  52 |   1 |
+-----+-----+-----+-----+-----+
| 104 |   9 |  32 |  11 |   6 |
+-----+-----+-----+-----+-----+
|  49 |  53 |   7 |  75 |  38 |
+-----+-----+-----+-----+-----+
| 101 |  86 |  76 |  42 |  82 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  67 |  69 |  19 |  97 |  60 |
+-----+-----+-----+-----+-----+
| 110 | 107 |  36 |  51 |  21 |
+-----+-----+-----+-----+-----+
|  57 | 103 | 116 | 119 |  71 |
+-----+-----+-----+-----+-----+
|  70 |  85 |  16 |  24 |  83 |
+-----+-----+-----+-----+-----+
|  98 | 125 |  27 |  79 | 105 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
| 109 | 118 |  77 |  56 |  26 |
+-----+-----+-----+-----+-----+
|  18 | 106 |  78 |  68 | 113 |
+-----+-----+-----+-----+-----+
|  95 | 102 |  63 |  89 |  73 |
+-----+-----+-----+-----+-----+
|  10 |  54 | 108 |  45 | 122 |
+-----+-----+-----+-----+-----+
|  28 |  30 |  12 | 114 |  87 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  20 |  64 |  92 |  88 | 123 |
+-----+-----+-----+-----+-----+
|  61 | 124 | 111 |  33 |  55 |
+-----+-----+-----+-----+-----+
|  72 | 112 |  59 | 120 |  34 |
+-----+-----+-----+-----+-----+
| 100 |   5 |  91 |  99 |  74 |
+-----+-----+-----+-----+-----+
|  66 |  93 |  90 | 117 |  94 |
+-----+-----+-----+-----+-----+
```
Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   5 |   4 |   1 |   2 |   8 |
+-----+-----+-----+-----+-----+
|  10 |  22 |   7 |  12 |  14 |
+-----+-----+-----+-----+-----+
|  21 |   3 |  13 |  15 |  17 |
+-----+-----+-----+-----+-----+
|  23 |   6 |  25 |  30 |  20 |
+-----+-----+-----+-----+-----+
|  16 |  36 |  49 |  11 |  18 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  19 |  27 |  26 |  78 |   9 |
+-----+-----+-----+-----+-----+
|  32 |  24 |  54 |  72 |  33 |
+-----+-----+-----+-----+-----+
|  41 |  39 |  40 |  57 |  31 |
+-----+-----+-----+-----+-----+
|  37 |  29 |  35 |  51 |  48 |
+-----+-----+-----+-----+-----+
|  42 |  44 |  38 |  34 |  63 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  69 |  46 |  50 |  28 |  58 |
+-----+-----+-----+-----+-----+
|  53 |  43 |  73 |  59 |  45 |
+-----+-----+-----+-----+-----+
|  55 |  56 |  66 |  61 |  67 |
+-----+-----+-----+-----+-----+
|  64 |  60 |  62 |  74 |  65 |
+-----+-----+-----+-----+-----+
|  68 |  84 |  47 |  91 |  75 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td></td>
<td>

```
Slice 4:
+-----+-----+-----+-----+-----+
|  71 |  70 |  77 | 100 |  93 |
+-----+-----+-----+-----+-----+
|  79 | 109 |  97 |  76 |  83 |
+-----+-----+-----+-----+-----+
|  80 |  92 |  94 |  90 |  52 |
+-----+-----+-----+-----+-----+
|  87 | 101 | 118 |  95 |  82 |
+-----+-----+-----+-----+-----+
|  96 |  86 |  81 |  89 | 103 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 122 |  98 |  85 | 123 | 102 |
+-----+-----+-----+-----+-----+
| 105 | 104 | 112 | 111 |  99 |
+-----+-----+-----+-----+-----+
| 113 | 120 | 114 | 108 | 125 |
+-----+-----+-----+-----+-----+
| 107 | 116 |  88 | 119 | 110 |
+-----+-----+-----+-----+-----+
| 106 | 117 | 121 | 124 | 115 |
+-----+-----+-----+-----+-----+
```

Time: 29.13
Best Cost: 1078</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Initial State
Initial Fitness: 4290</td>
</tr>
</table>

		`` Slice 1: +-----+-----+-----+-----+-----+ | 27 | 62 | 46 | 44 | 123 | +-----+-----+-----+-----+-----+ | 94 | 74 | 21 | 11 | 45 | +-----+-----+-----+-----+-----+ | 6 | 73 | 108 | 8 | 34 | +-----+-----+-----+-----+-----+ | 42 | 1 | 38 | 59 | 47 | +-----+-----+-----+-----+-----+ | 14 | 90 | 23 | 86 | 35 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 2 | 18 | 55 | 49 | 88 | +-----+-----+-----+-----+-----+ | 58 | 113 | 104 | 10 | 67 | +-----+-----+-----+-----+-----+ | 16 | 4 | 41 | 43 | 30 | +-----+-----+-----+-----+-----+ | 17 | 26 | 92 | 24 | 28 | +-----+-----+-----+-----+-----+ | 39 | 117 | 95 | 81 | 15 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 12 | 32 | 98 | 22 | 115 | +-----+-----+-----+-----+-----+ | 97 | 9 | 93 | 65 | 69 | +-----+-----+-----+-----+-----+ | 110 | 100 | 83 | 76 | 75 | +-----+-----+-----+-----+-----+ | 66 | 5 | 56 | 33 | 118 | +-----+-----+-----+-----+-----+ | 84 | 107 | 50 | 7 | 102 | +-----+-----+-----+-----+-----+ ```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  64 |  71 |  87 |  99 |  25 |
+-----+-----+-----+-----+-----+
|  91 |  13 | 109 | 122 |  63 |
+-----+-----+-----+-----+-----+
|  77 | 101 | 116 |  68 | 114 |
+-----+-----+-----+-----+-----+
| 119 |  51 |  48 | 125 |  36 |
+-----+-----+-----+-----+-----+
|  19 |   3 |  57 |  96 |  80 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  85 |  79 |  70 |  61 |  52 |
+-----+-----+-----+-----+-----+
|  31 |  72 | 121 | 112 | 111 |
+-----+-----+-----+-----+-----+
| 120 | 105 |  82 |  40 |  53 |
+-----+-----+-----+-----+-----+
|  54 |  78 | 103 |  89 |  29 |
+-----+-----+-----+-----+-----+
|  20 | 124 |  37 | 106 |  60 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|  16 |  10 |   3 |   1 |  14 |
+-----+-----+-----+-----+-----+
|  12 |   9 |   5 |  11 |  28 |
+-----+-----+-----+-----+-----+
|  23 |  20 |  21 |  30 |  13 |
+-----+-----+-----+-----+-----+
|  15 |   2 |   8 |  22 |   7 |
+-----+-----+-----+-----+-----+
|  18 |  19 |  27 |  26 |  24 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|   6 |  33 |  47 |  48 |  40 |
+-----+-----+-----+-----+-----+
|  35 |  78 |  38 |  53 |  31 |
+-----+-----+-----+-----+-----+
|  32 |  29 |  44 |  17 |  34 |
+-----+-----+-----+-----+-----+
|  39 |  36 |  37 |   4 |  59 |
+-----+-----+-----+-----+-----+
|  55 |  42 |  45 |  54 |  49 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  56 |  50 |  57 |  46 |  73 |
+-----+-----+-----+-----+-----+
|  25 |  71 |  68 |  60 |  43 |
+-----+-----+-----+-----+-----+
| 102 |  67 |  61 |  58 |  63 |
+-----+-----+-----+-----+-----+
|  62 |  65 |  66 |  52 |  77 |
+-----+-----+-----+-----+-----+
|  64 |  79 |  72 |  41 |  85 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td></td><td><pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 82 | 69 | 76 | 75 | 74 |
+-----+-----+-----+-----+-----+
| 70 | 51 | 84 | 80 | 103 |
+-----+-----+-----+-----+-----+
| 88 | 87 | 81 | 105 | 95 |
+-----+-----+-----+-----+-----+
| 89 | 101 | 96 | 83 | 94 |
+-----+-----+-----+-----+-----+
| 92 | 117 | 114 | 106 | 93 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 100 | 104 | 98 | 99 | 115 |
+-----+-----+-----+-----+-----+
| 125 | 108 | 90 | 91 | 116 |
+-----+-----+-----+-----+-----+
| 86 | 123 | 120 | 110 | 107 |
+-----+-----+-----+-----+-----+
| 124 | 121 | 97 | 122 | 109 |
+-----+-----+-----+-----+-----+
| 113 | 112 | 119 | 111 | 118 |
+-----+-----+-----+-----+-----+
</pre>
Time: 19.22
Best Cost: 1142</td></tr>
<tr><td></td><td>3</td><td>Initial State
Initial Fitness: 4268</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  56 |  30 |  81 |  57 |  67 |
+-----+-----+-----+-----+-----+
| 110 |  31 |   8 |  14 |  20 |
+-----+-----+-----+-----+-----+
|   2 |  48 |  17 |  21 |  40 |
+-----+-----+-----+-----+-----+
|  45 |  18 |  78 | 122 |  46 |
+-----+-----+-----+-----+-----+
|  69 |  68 |  39 |  24 |  43 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
| 106 |  61 |  29 |  47 |  12 |
+-----+-----+-----+-----+-----+
| 123 |  15 |   5 |  27 |  73 |
+-----+-----+-----+-----+-----+
| 111 |  23 | 118 |  34 |  98 |
+-----+-----+-----+-----+-----+
| 103 |  13 |  92 | 119 |   1 |
+-----+-----+-----+-----+-----+
|  84 |  25 | 120 |  41 |  55 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  72 |  36 | 101 | 113 |  71 |
+-----+-----+-----+-----+-----+
|  51 |  38 |  28 |  49 |   7 |
+-----+-----+-----+-----+-----+
| 116 | 100 |  50 |  83 |  22 |
+-----+-----+-----+-----+-----+
|  63 |  76 | 105 |  52 |  95 |
+-----+-----+-----+-----+-----+
|  33 |  80 |  75 |  96 |  66 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  26 |  94 |  32 |  11 |  35 |
+-----+-----+-----+-----+-----+
| 117 |  16 |   3 |   6 | 107 |
+-----+-----+-----+-----+-----+
| 125 |  42 |  82 |  91 |  19 |
+-----+-----+-----+-----+-----+
|  60 |  44 |  65 |   4 | 124 |
+-----+-----+-----+-----+-----+
|  74 |  77 |  86 |  90 |  93 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  10 |  99 |  70 |  59 | 109 |
+-----+-----+-----+-----+-----+
|  97 | 114 | 112 | 102 |  58 |
+-----+-----+-----+-----+-----+
|  37 |   9 | 104 | 108 |  79 |
+-----+-----+-----+-----+-----+
|  89 |  87 |  85 |  54 |  62 |
+-----+-----+-----+-----+-----+
| 121 |  53 |  64 | 115 |  88 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   3 |   4 |  10 |   1 |  32 |
+-----+-----+-----+-----+-----+
|  17 |   7 |   2 |   5 |  20 |
+-----+-----+-----+-----+-----+
|  23 |  11 |   9 |  12 |  31 |
+-----+-----+-----+-----+-----+
|  13 |  15 |  30 |  24 |  21 |
+-----+-----+-----+-----+-----+
|  14 |  22 |   8 |  16 |  33 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  29 |  28 |  25 |  39 |  36 |
+-----+-----+-----+-----+-----+
|  18 |  38 |  45 |   6 |  35 |
+-----+-----+-----+-----+-----+
|  34 |  43 |  27 |  67 |  72 |
+-----+-----+-----+-----+-----+
|  85 |  44 |  26 |  41 |  46 |
+-----+-----+-----+-----+-----+
|  57 |  50 |  77 |  19 |  37 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  47 |  53 |  42 |  63 |  61 |
+-----+-----+-----+-----+-----+
|  64 |  55 |  58 |  62 |  56 |
+-----+-----+-----+-----+-----+
|  89 |  60 | 100 |  54 |  59 |
+-----+-----+-----+-----+-----+
|  66 |  52 |  76 |  40 |  49 |
+-----+-----+-----+-----+-----+
|  70 |  65 |  51 |  48 |  87 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td></td><td>Slice 4:
+-----+-----+-----+-----+-----+
| 80 | 71 | 101 | 75 | 68 |
+-----+-----+-----+-----+-----+
| 74 | 99 | 78 | 81 | 83 |
+-----+-----+-----+-----+-----+
| 79 | 73 | 105 | 94 | 108 |
+-----+-----+-----+-----+-----+
| 84 | 69 | 88 | 82 | 92 |
+-----+-----+-----+-----+-----+
| 97 | 96 | 98 | 95 | 109 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 93 | 118 | 90 | 91 | 112 |
+-----+-----+-----+-----+-----+
| 107 | 104 | 86 | 114 | 119 |
+-----+-----+-----+-----+-----+
| 113 | 116 | 103 | 117 | 111 |
+-----+-----+-----+-----+-----+
| 120 | 121 | 106 | 125 | 123 |
+-----+-----+-----+-----+-----+
| 102 | 122 | 110 | 115 | 124 |
+-----+-----+-----+-----+-----+

Time: 20.07
Best Cost: 1154</td></tr>
<tr><td>50000</td><td>1</td><td>Initial State
Initial Fitness = 4176</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  91 |  25 | 109 |  14 |  28 |
+-----+-----+-----+-----+-----+
| 122 |  10 |   2 | 125 |  21 |
+-----+-----+-----+-----+-----+
|  26 |  85 |  15 |  67 |  60 |
+-----+-----+-----+-----+-----+
|  48 |  57 | 120 |  59 |  68 |
+-----+-----+-----+-----+-----+
|  56 |  72 |   3 |   7 |  66 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  83 |  77 |   9 |  39 |  79 |
+-----+-----+-----+-----+-----+
|  32 |  75 |  40 |  24 |  78 |
+-----+-----+-----+-----+-----+
| 100 |   1 |  87 |   4 |  16 |
+-----+-----+-----+-----+-----+
|  58 |  64 |  12 |  80 |  37 |
+-----+-----+-----+-----+-----+
|  53 |  55 |  65 |  94 |  31 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|   6 | 110 |   8 |  18 |  44 |
+-----+-----+-----+-----+-----+
|  46 |  73 | 114 |  71 |  70 |
+-----+-----+-----+-----+-----+
|  54 |  20 | 105 |  30 |  29 |
+-----+-----+-----+-----+-----+
|  81 |  19 |  63 |  97 |  33 |
+-----+-----+-----+-----+-----+
|  93 |  50 | 106 |  34 |  23 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  69 |  11 |  62 |  45 |  52 |
+-----+-----+-----+-----+-----+
|  96 |  61 | 108 | 118 |  84 |
+-----+-----+-----+-----+-----+
|  17 |  36 |   5 |  41 |  47 |
+-----+-----+-----+-----+-----+
| 104 |  90 |  27 | 124 |  92 |
+-----+-----+-----+-----+-----+
|  86 | 103 | 107 |  76 |  88 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 123 | 113 |  49 | 117 |  98 |
+-----+-----+-----+-----+-----+
|  22 |  38 |  43 | 121 | 102 |
+-----+-----+-----+-----+-----+
|  42 | 101 | 115 | 119 |  82 |
+-----+-----+-----+-----+-----+
| 116 |  35 |  13 |  51 |  89 |
+-----+-----+-----+-----+-----+
| 111 |  74 |  95 | 112 |  99 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   7 |   2 |  12 |   6 |  15 |
+-----+-----+-----+-----+-----+
|  11 |  16 |   4 |   1 |   8 |
+-----+-----+-----+-----+-----+
|  20 |  19 |  13 |  18 |  17 |
+-----+-----+-----+-----+-----+
|  21 |   5 |   9 |  25 |  14 |
+-----+-----+-----+-----+-----+
|  24 |  26 |  23 |  29 |  27 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  10 |  34 |  31 |  35 |  22 |
+-----+-----+-----+-----+-----+
|  28 |   3 |  43 |  38 |  37 |
+-----+-----+-----+-----+-----+
|  30 |  47 |  48 |  36 |  45 |
+-----+-----+-----+-----+-----+
|  39 |  33 |  46 |  41 |  55 |
+-----+-----+-----+-----+-----+
|  44 |  32 |  51 |  50 |  66 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  49 |  42 |  54 |  62 |  59 |
+-----+-----+-----+-----+-----+
|  52 |  60 |  68 |  53 |  40 |
+-----+-----+-----+-----+-----+
|  67 |  57 |  65 |  58 |  78 |
+-----+-----+-----+-----+-----+
|  63 |  56 |  61 |  84 |  70 |
+-----+-----+-----+-----+-----+
|  64 |  74 |  72 |  79 |  76 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td rowspan="2"></td>
<td></td>
<td>
<pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 71 | 81 | 92 | 75 | 83 |
+-----+-----+-----+-----+-----+
| 77 | 89 | 69 | 73 | 82 |
+-----+-----+-----+-----+-----+
| 90 | 86 | 87 | 91 | 99 |
+-----+-----+-----+-----+-----+
| 97 | 95 | 88 | 94 | 120 |
+-----+-----+-----+-----+-----+
| 98 | 80 | 104 | 101 | 116 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 111 | 100 | 102 | 103 | 96 |
+-----+-----+-----+-----+-----+
| 105 | 106 | 110 | 108 | 85 |
+-----+-----+-----+-----+-----+
| 107 | 112 | 93 | 115 | 119 |
+-----+-----+-----+-----+-----+
| 118 | 117 | 114 | 113 | 122 |
+-----+-----+-----+-----+-----+
| 109 | 123 | 124 | 121 | 125 |
+-----+-----+-----+-----+-----+
</pre>
Time: 135.49
Best Cost: 748
</td>
</tr>
<tr>
<td>2</td>
<td>Initial State
Initial Fitness: 4364</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
| 120 |  18 |  67 |   5 |  25 |
+-----+-----+-----+-----+-----+
| 104 | 115 |  29 |  58 |  71 |
+-----+-----+-----+-----+-----+
|  51 |  68 | 107 |  26 |  44 |
+-----+-----+-----+-----+-----+
|  36 |  92 |  48 |  74 |  10 |
+-----+-----+-----+-----+-----+
|  21 |  14 |   6 |  98 |  37 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
| 117 |  45 | 110 |  33 |  93 |
+-----+-----+-----+-----+-----+
|  22 |  46 |  15 |   9 |  52 |
+-----+-----+-----+-----+-----+
|  12 | 121 |  39 | 116 |  75 |
+-----+-----+-----+-----+-----+
|  79 |   4 |  16 |  19 |  28 |
+-----+-----+-----+-----+-----+
|  87 |  30 | 114 |  23 |  63 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  11 |  70 |  57 |   2 |   7 |
+-----+-----+-----+-----+-----+
|  61 | 113 | 125 |  17 |  65 |
+-----+-----+-----+-----+-----+
|  77 |  31 |  56 |  99 |  53 |
+-----+-----+-----+-----+-----+
|  54 |  27 |  60 |  43 |  13 |
+-----+-----+-----+-----+-----+
|  55 |  72 |  35 |  76 |  82 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td></td>
<td>
<pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 101 | 34 | 109 | 123 | 95 |
+-----+-----+-----+-----+-----+
| 8 | 1 | 83 | 122 | 97 |
+-----+-----+-----+-----+-----+
| 62 | 40 | 118 | 106 | 105 |
+-----+-----+-----+-----+-----+
| 94 | 20 | 102 | 42 | 80 |
+-----+-----+-----+-----+-----+
| 38 | 96 | 47 | 103 | 85 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 69 | 111 | 32 | 73 | 91 |
+-----+-----+-----+-----+-----+
| 3 | 84 | 41 | 59 | 119 |
+-----+-----+-----+-----+-----+
| 78 | 90 | 81 | 86 | 64 |
+-----+-----+-----+-----+-----+
| 124 | 108 | 112 | 100 | 66 |
+-----+-----+-----+-----+-----+
| 49 | 89 | 88 | 50 | 24 |
+-----+-----+-----+-----+-----+
</pre>
Final State
</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  15 |   4 |   1 |   3 |   8 |
+-----+-----+-----+-----+-----+
|   5 |   7 |   2 |  11 |  14 |
+-----+-----+-----+-----+-----+
|  23 |  10 |   9 |   6 |  30 |
+-----+-----+-----+-----+-----+
|  18 |  19 |  13 |  24 |  17 |
+-----+-----+-----+-----+-----+
|  12 |  42 |  27 |  22 |  20 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  25 |  29 |  21 |  33 |  26 |
+-----+-----+-----+-----+-----+
|  28 |  32 |  16 |  31 |  36 |
+-----+-----+-----+-----+-----+
|  34 |  53 |  44 |  35 |  52 |
+-----+-----+-----+-----+-----+
|  40 |  41 |  38 |  46 |  43 |
+-----+-----+-----+-----+-----+
|  39 |  45 |  62 |  54 |  49 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  59 |  58 |  55 |  51 |  64 |
+-----+-----+-----+-----+-----+
|  47 |  56 |  57 |  73 |  50 |
+-----+-----+-----+-----+-----+
|  37 |  76 |  69 |  70 |  80 |
+-----+-----+-----+-----+-----+
|  48 |  71 |  63 |  68 |  61 |
+-----+-----+-----+-----+-----+
|  74 |  60 |  65 |  79 |  66 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td></td><td><pre>Slice 4:
+-----+-----+-----+-----+-----+
| 75 | 67 | 81 | 78 | 91 |
+-----+-----+-----+-----+-----+
| 77 | 100 | 83 | 84 | 93 |
+-----+-----+-----+-----+-----+
| 99 | 72 | 86 | 109 | 96 |
+-----+-----+-----+-----+-----+
| 92 | 89 | 94 | 104 | 85 |
+-----+-----+-----+-----+-----+
| 87 | 97 | 90 | 95 | 88 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 111 | 82 | 101 | 107 | 113 |
+-----+-----+-----+-----+-----+
| 117 | 98 | 108 | 102 | 105 |
+-----+-----+-----+-----+-----+
| 112 | 115 | 114 | 106 | 110 |
+-----+-----+-----+-----+-----+
| 123 | 103 | 125 | 120 | 119 |
+-----+-----+-----+-----+-----+
| 122 | 118 | 121 | 124 | 116 |
+-----+-----+-----+-----+-----+</pre>
Time: 131.07
Best Cost: 766</td></tr>
<tr><td></td><td>3</td><td>Initial State
Initial Fitness: 4188</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  16 |  10 |  99 |  28 |  22 |
+-----+-----+-----+-----+-----+
| 125 |  81 |  43 |  42 |  68 |
+-----+-----+-----+-----+-----+
|  35 |   7 |  37 |  20 |  25 |
+-----+-----+-----+-----+-----+
|  92 |  61 |  95 |  52 |  56 |
+-----+-----+-----+-----+-----+
| 118 |  50 |  12 |  32 |  14 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|   2 |  63 |  18 |  60 |  41 |
+-----+-----+-----+-----+-----+
|  59 |  75 |  26 |  23 |  79 |
+-----+-----+-----+-----+-----+
|  38 |  86 | 116 |  78 |  40 |
+-----+-----+-----+-----+-----+
|  67 | 101 | 111 |  15 |   8 |
+-----+-----+-----+-----+-----+
|  31 |  49 |  24 |  33 |  88 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  55 |  58 |  98 |  66 |  72 |
+-----+-----+-----+-----+-----+
|  73 |  44 |  54 | 103 |   4 |
+-----+-----+-----+-----+-----+
|  64 | 110 |  27 | 100 |  39 |
+-----+-----+-----+-----+-----+
| 106 |  17 |  83 | 117 |  74 |
+-----+-----+-----+-----+-----+
|  11 |  62 |  51 |   5 | 112 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  53 |  93 |  80 | 108 |  85 |
+-----+-----+-----+-----+-----+
|  70 | 121 |   1 |  29 |  96 |
+-----+-----+-----+-----+-----+
| 105 |   3 |  69 |  48 | 115 |
+-----+-----+-----+-----+-----+
|  77 |  30 |  36 |  89 | 104 |
+-----+-----+-----+-----+-----+
| 102 |  90 | 123 |  65 |  84 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  97 |  57 |  76 | 107 |  91 |
+-----+-----+-----+-----+-----+
| 114 | 113 |  13 |  45 |  82 |
+-----+-----+-----+-----+-----+
|  46 |  87 | 120 | 122 |  21 |
+-----+-----+-----+-----+-----+
|  71 |  47 | 109 |  19 | 124 |
+-----+-----+-----+-----+-----+
|   6 |  34 |  94 | 119 |   9 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   8 |   2 |   3 |   1 |  17 |
+-----+-----+-----+-----+-----+
|   7 |   9 |  10 |  12 |  45 |
+-----+-----+-----+-----+-----+
|   4 |  29 |  16 |  15 |   6 |
+-----+-----+-----+-----+-----+
|  21 |  13 |  27 |  19 |  23 |
+-----+-----+-----+-----+-----+
|  14 |   5 |  20 |  31 |  11 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  24 |  32 |  26 |  18 |  25 |
+-----+-----+-----+-----+-----+
|  34 |  35 |  41 |  30 |  28 |
+-----+-----+-----+-----+-----+
|  37 |  22 |  39 |  36 |  53 |
+-----+-----+-----+-----+-----+
|  40 |  48 |  44 |  62 |  50 |
+-----+-----+-----+-----+-----+
|  42 |  51 |  46 |  49 |  43 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  61 |  38 |  55 |  33 |  52 |
+-----+-----+-----+-----+-----+
|  68 |  57 |  59 |  47 |  63 |
+-----+-----+-----+-----+-----+
|  54 |  56 |  73 |  75 |  71 |
+-----+-----+-----+-----+-----+
|  66 |  58 |  76 | 104 |  60 |
+-----+-----+-----+-----+-----+
|  83 |  70 |  74 |  67 |  80 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td></td><td>Slice 4:
+-----+-----+-----+-----+-----+
| 69 | 64 | 65 | 81 | 87 |
+-----+-----+-----+-----+-----+
| 72 | 89 | 78 | 77 | 93 |
+-----+-----+-----+-----+-----+
| 90 | 86 | 79 | 85 | 82 |
+-----+-----+-----+-----+-----+
| 91 | 99 | 95 | 94 | 92 |
+-----+-----+-----+-----+-----+
| 111 | 96 | 84 | 108 | 102 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 100 | 88 | 101 | 103 | 106 |
+-----+-----+-----+-----+-----+
| 113 | 98 | 105 | 120 | 109 |
+-----+-----+-----+-----+-----+
| 97 | 117 | 125 | 107 | 112 |
+-----+-----+-----+-----+-----+
| 114 | 124 | 115 | 122 | 116 |
+-----+-----+-----+-----+-----+
| 110 | 119 | 118 | 121 | 123 |
+-----+-----+-----+-----+-----+

Time: 130.16
Best Cost: 800</td></tr>
<tr><td>75000</td><td>1</td><td>Initial State
Initial Fitness:</td></tr>
</table>

		`` Slice 1: +-----+-----+-----+-----+-----+ | 21 | 15 | 44 | 35 | 47 | +-----+-----+-----+-----+-----+ | 12 | 34 | 50 | 28 | 27 | +-----+-----+-----+-----+-----+ | 61 | 83 | 10 | 29 | 6 | +-----+-----+-----+-----+-----+ | 110 | 7 | 36 | 22 | 76 | +-----+-----+-----+-----+-----+ | 62 | 41 | 13 | 68 | 14 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 19 | 18 | 120 | 103 | 89 | +-----+-----+-----+-----+-----+ | 40 | 25 | 55 | 88 | 106 | +-----+-----+-----+-----+-----+ | 84 | 38 | 100 | 121 | 111 | +-----+-----+-----+-----+-----+ | 52 | 16 | 4 | 43 | 53 | +-----+-----+-----+-----+-----+ | 49 | 20 | 102 | 63 | 51 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 57 | 80 | 119 | 60 | 2 | +-----+-----+-----+-----+-----+ | 45 | 93 | 124 | 67 | 91 | +-----+-----+-----+-----+-----+ | 87 | 54 | 115 | 95 | 65 | +-----+-----+-----+-----+-----+ | 26 | 97 | 117 | 81 | 98 | +-----+-----+-----+-----+-----+ | 79 | 31 | 11 | 64 | 46 | +-----+-----+-----+-----+-----+ ```

<table>
<tr>
<td></td>
<td></td>
<td>
<pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 107 | 56 | 48 | 30 | 33 |
+-----+-----+-----+-----+-----+
| 101 | 77 | 42 | 109 | 90 |
+-----+-----+-----+-----+-----+
| 82 | 74 | 113 | 78 | 32 |
+-----+-----+-----+-----+-----+
| 8 | 75 | 94 | 116 | 17 |
+-----+-----+-----+-----+-----+
| 72 | 39 | 104 | 118 | 24 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 58 | 9 | 105 | 122 | 73 |
+-----+-----+-----+-----+-----+
| 96 | 69 | 108 | 5 | 1 |
+-----+-----+-----+-----+-----+
| 37 | 123 | 114 | 3 | 70 |
+-----+-----+-----+-----+-----+
| 85 | 112 | 59 | 23 | 66 |
+-----+-----+-----+-----+-----+
| 92 | 99 | 71 | 125 | 86 |
+-----+-----+-----+-----+-----+
</pre>
Final State
</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|   3 |   4 |  13 |  10 |  17 |
+-----+-----+-----+-----+-----+
|   8 |  18 |  14 |   5 |   1 |
+-----+-----+-----+-----+-----+
|  16 |   7 |   9 |   2 |  24 |
+-----+-----+-----+-----+-----+
|  19 |  23 |  15 |  20 |  27 |
+-----+-----+-----+-----+-----+
|  22 |  21 |  33 |   6 |  28 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  11 |  32 |  12 |  29 |  40 |
+-----+-----+-----+-----+-----+
|  36 |  35 |  31 |  30 |  34 |
+-----+-----+-----+-----+-----+
|  87 |  39 |  25 |  45 |  37 |
+-----+-----+-----+-----+-----+
|  44 |  42 |  41 |  48 |  43 |
+-----+-----+-----+-----+-----+
|  56 |  38 |  53 |  46 |  55 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  54 |  52 |  47 |  50 |  57 |
+-----+-----+-----+-----+-----+
|  59 |  62 |  60 |  76 |  26 |
+-----+-----+-----+-----+-----+
|  69 |  58 |  66 |  65 |  51 |
+-----+-----+-----+-----+-----+
|  67 |  64 |  63 |  78 |  61 |
+-----+-----+-----+-----+-----+
|  49 |  68 |  70 |  73 |  74 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  77 |  83 |  71 |  79 |  81 |
+-----+-----+-----+-----+-----+
|  85 |  80 |  75 |  84 |  86 |
+-----+-----+-----+-----+-----+
|  91 |  88 |  82 |  94 |  97 |
+-----+-----+-----+-----+-----+
|  90 |  72 |  99 |  93 |  92 |
+-----+-----+-----+-----+-----+
| 100 |  96 |  98 | 111 | 112 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 104 |  89 | 103 | 101 | 110 |
+-----+-----+-----+-----+-----+
| 115 | 102 | 123 | 117 |  95 |
+-----+-----+-----+-----+-----+
| 109 | 107 | 114 | 113 | 108 |
+-----+-----+-----+-----+-----+
| 119 | 121 | 106 | 105 | 116 |
+-----+-----+-----+-----+-----+
| 124 | 118 | 120 | 125 | 122 |
+-----+-----+-----+-----+-----+
```

		Time: 197.10 Best Cost: 752
	2	Initial State Initial Fitness: 4322

```
Slice 1:
+-----+-----+-----+-----+-----+
| 112 |   8 |  25 |  56 |  17 |
+-----+-----+-----+-----+-----+
|  16 |  45 | 111 |  19 |  58 |
+-----+-----+-----+-----+-----+
|  57 |  63 |  14 |  44 | 114 |
+-----+-----+-----+-----+-----+
|  22 | 106 |  47 |  60 | 104 |
+-----+-----+-----+-----+-----+
| 124 |  30 |   6 |  38 |  98 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  33 |  66 |  76 |  23 |  20 |
+-----+-----+-----+-----+-----+
|   4 |  53 |  49 |  65 | 117 |
+-----+-----+-----+-----+-----+
|  78 |  21 |  15 | 102 | 119 |
+-----+-----+-----+-----+-----+
|   1 |   9 |  67 |  32 |  24 |
+-----+-----+-----+-----+-----+
|  10 |  54 |  36 |  62 |  48 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  37 | 105 |  70 |  80 |  41 |
+-----+-----+-----+-----+-----+
|  86 | 109 |  91 |  73 | 125 |
+-----+-----+-----+-----+-----+
|  26 |  35 |  95 |  72 |  55 |
+-----+-----+-----+-----+-----+
|  12 |  71 |  28 |  29 | 107 |
+-----+-----+-----+-----+-----+
|  75 |  52 |  93 |  61 |  40 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  50 |  31 | 122 |  77 |  74 |
+-----+-----+-----+-----+-----+
| 116 |   3 | 108 |  42 |  64 |
+-----+-----+-----+-----+-----+
| 121 |  34 |  39 |  84 |  92 |
+-----+-----+-----+-----+-----+
|  13 |  85 |  90 |   7 |  46 |
+-----+-----+-----+-----+-----+
|  79 | 120 | 113 |  69 |  99 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  89 | 123 |  18 |  88 | 100 |
+-----+-----+-----+-----+-----+
|  87 | 115 |  68 | 101 |   5 |
+-----+-----+-----+-----+-----+
| 110 |   2 |  43 |  59 |  81 |
+-----+-----+-----+-----+-----+
|  83 |  97 |  96 |  82 |  11 |
+-----+-----+-----+-----+-----+
|  51 | 103 |  94 |  27 | 118 |
+-----+-----+-----+-----+-----+
```

Final State

		`` Slice 1: +-----+-----+-----+-----+-----+ | 3 | 2 | 4 | 7 | 6 | +-----+-----+-----+-----+-----+ | 1 | 8 | 9 | 18 | 11 | +-----+-----+-----+-----+-----+ | 23 | 10 | 12 | 5 | 15 | +-----+-----+-----+-----+-----+ | 16 | 19 | 13 | 14 | 27 | +-----+-----+-----+-----+-----+ | 22 | 34 | 21 | 30 | 35 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 26 | 20 | 24 | 31 | 28 | +-----+-----+-----+-----+-----+ | 33 | 29 | 25 | 17 | 36 | +-----+-----+-----+-----+-----+ | 43 | 38 | 40 | 32 | 41 | +-----+-----+-----+-----+-----+ | 48 | 51 | 37 | 39 | 44 | +-----+-----+-----+-----+-----+ | 46 | 45 | 63 | 47 | 50 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 49 | 52 | 53 | 54 | 42 | +-----+-----+-----+-----+-----+ | 55 | 77 | 60 | 56 | 67 | +-----+-----+-----+-----+-----+ | 62 | 57 | 61 | 71 | 66 | +-----+-----+-----+-----+-----+ | 59 | 64 | 58 | 68 | 70 | +-----+-----+-----+-----+-----+ | 69 | 73 | 97 | 65 | 76 | +-----+-----+-----+-----+-----+ ```

<table>
<tr><td></td><td></td><td>Slice 4:
+-----+-----+-----+-----+-----+
| 80 | 81 | 87 | 79 | 75 |
+-----+-----+-----+-----+-----+
| 83 | 86 | 74 | 85 | 91 |
+-----+-----+-----+-----+-----+
| 72 | 98 | 88 | 103 | 89 |
+-----+-----+-----+-----+-----+
| 100 | 84 | 93 | 104 | 96 |
+-----+-----+-----+-----+-----+
| 82 | 90 | 102 | 92 | 94 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 108 | 110 | 95 | 99 | 121 |
+-----+-----+-----+-----+-----+
| 109 | 101 | 106 | 78 | 107 |
+-----+-----+-----+-----+-----+
| 124 | 120 | 116 | 112 | 113 |
+-----+-----+-----+-----+-----+
| 122 | 111 | 105 | 115 | 114 |
+-----+-----+-----+-----+-----+
| 117 | 118 | 119 | 123 | 125 |
+-----+-----+-----+-----+-----+

Time: 204.26
Best Cost: 646</td></tr>
<tr><td></td><td>3</td><td>Initial State
Initial Fitness: 4276</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  69 |  55 |  16 |  70 |   4 |
+-----+-----+-----+-----+-----+
|  49 |  10 |  26 |  12 |  29 |
+-----+-----+-----+-----+-----+
|  13 |  51 | 106 |  84 |  59 |
+-----+-----+-----+-----+-----+
| 103 |  18 | 108 |   3 |  20 |
+-----+-----+-----+-----+-----+
|  25 |   7 |  17 |  73 |  24 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  64 |  39 |  38 |  53 |  89 |
+-----+-----+-----+-----+-----+
|  21 | 105 |  19 | 110 |  57 |
+-----+-----+-----+-----+-----+
|  35 |  68 | 114 |  11 |  52 |
+-----+-----+-----+-----+-----+
|  94 |  50 |  67 |  66 |  96 |
+-----+-----+-----+-----+-----+
|  81 |  72 |  36 |   9 |  30 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  62 | 101 |   8 |  98 |  77 |
+-----+-----+-----+-----+-----+
|  61 |  31 |  22 |  65 | 124 |
+-----+-----+-----+-----+-----+
| 125 |  91 |  75 |  27 | 112 |
+-----+-----+-----+-----+-----+
|  45 | 121 |  40 |  42 |  48 |
+-----+-----+-----+-----+-----+
|  56 |   2 |  60 |  74 |   1 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  28 |  33 |  88 |   5 | 107 |
+-----+-----+-----+-----+-----+
| 119 |  63 | 113 | 102 | 123 |
+-----+-----+-----+-----+-----+
|  79 | 118 |  92 | 122 |  90 |
+-----+-----+-----+-----+-----+
|  99 | 116 |  37 | 104 |  47 |
+-----+-----+-----+-----+-----+
| 117 |  44 |  93 |  85 | 115 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  76 |  86 | 111 |  87 |  46 |
+-----+-----+-----+-----+-----+
|   6 |  32 |  43 |  23 |  34 |
+-----+-----+-----+-----+-----+
|  15 |  41 |  80 |  95 |  54 |
+-----+-----+-----+-----+-----+
|  97 |  83 |  58 |  14 | 100 |
+-----+-----+-----+-----+-----+
| 109 |  71 | 120 |  78 |  82 |
+-----+-----+-----+-----+-----+
```

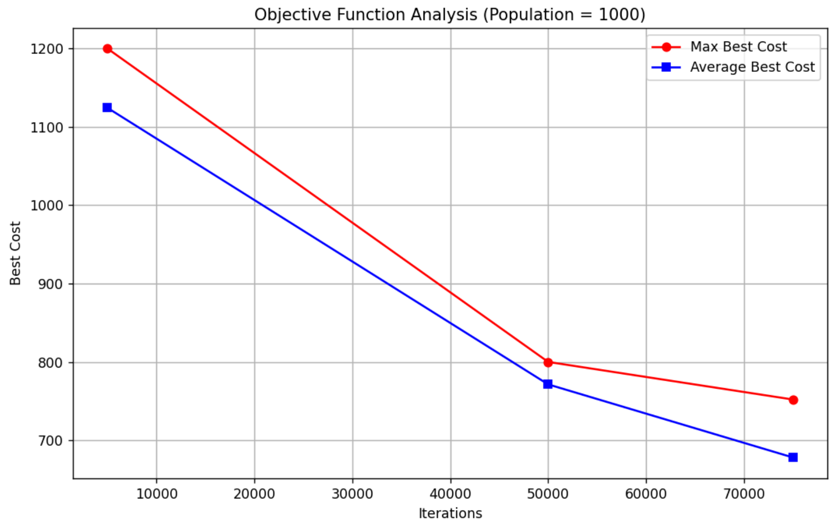
Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   1 |   5 |   2 |  17 |   3 |
+-----+-----+-----+-----+-----+
|   8 |   9 |   6 |   7 |  34 |
+-----+-----+-----+-----+-----+
|   4 |  10 |  15 |  11 |  21 |
+-----+-----+-----+-----+-----+
|  12 |  14 |  16 |  33 |  18 |
+-----+-----+-----+-----+-----+
|  13 |  20 |  19 |  22 |  24 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  23 |  31 |  26 |  25 |  38 |
+-----+-----+-----+-----+-----+
|  39 |  37 |  27 |  32 |  29 |
+-----+-----+-----+-----+-----+
|  35 |  36 |  28 |  30 |  44 |
+-----+-----+-----+-----+-----+
|  40 |  45 |  46 |  42 |  43 |
+-----+-----+-----+-----+-----+
|  48 |  60 |  47 |  49 |  53 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  41 |  50 |  52 |  78 |  58 |
+-----+-----+-----+-----+-----+
|  59 |  56 |  51 |  62 |  70 |
+-----+-----+-----+-----+-----+
|  65 |  54 |  68 |  55 |  74 |
+-----+-----+-----+-----+-----+
|  66 |  69 |  57 |  64 |  73 |
+-----+-----+-----+-----+-----+
|  67 |  80 |  72 |  61 |  95 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  71 |  75 |  63 |  79 |  87 |
+-----+-----+-----+-----+-----+
|  81 |  85 |  94 |  86 |  84 |
+-----+-----+-----+-----+-----+
|  93 |  91 |  77 |  90 |  92 |
+-----+-----+-----+-----+-----+
|  89 |  82 |  76 |  97 |  96 |
+-----+-----+-----+-----+-----+
|  83 |  98 |  99 |  88 | 102 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 117 | 106 | 108 | 105 | 112 |
+-----+-----+-----+-----+-----+
| 103 | 107 | 100 | 111 | 101 |
+-----+-----+-----+-----+-----+
| 109 | 104 | 114 | 124 | 116 |
+-----+-----+-----+-----+-----+
| 119 | 120 | 115 | 122 | 121 |
+-----+-----+-----+-----+-----+
| 118 | 125 | 123 | 113 | 110 |
+-----+-----+-----+-----+-----+
```

Time: 194.53
Best Cost: 636

Plot



Variasi 2 (Iterations = 5000)		
Population	**Percobaan**	**Hasil**

1000	1	Initial State Initial Fitness: 4300

```
Slice 1:
+-----+-----+-----+-----+-----+
|  13 | 117 |  48 |  14 |  54 |
+-----+-----+-----+-----+-----+
|  51 |   6 | 102 |  16 |   3 |
+-----+-----+-----+-----+-----+
|  18 |  59 |   5 |   8 |  20 |
+-----+-----+-----+-----+-----+
|  56 | 120 | 116 |  12 |  91 |
+-----+-----+-----+-----+-----+
|  72 |  98 |   4 |  23 |  19 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  78 |  47 |  22 | 121 |  77 |
+-----+-----+-----+-----+-----+
|  75 | 105 |   2 | 110 |  95 |
+-----+-----+-----+-----+-----+
|  15 |  92 |  30 |  33 |   7 |
+-----+-----+-----+-----+-----+
|  69 |  45 |  26 |  66 |  27 |
+-----+-----+-----+-----+-----+
|  44 |  36 |  57 | 107 | 100 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  10 | 109 |  64 |  38 |  62 |
+-----+-----+-----+-----+-----+
| 125 |  39 | 122 |  52 |  41 |
+-----+-----+-----+-----+-----+
|  85 |  97 |  70 |  96 |  40 |
+-----+-----+-----+-----+-----+
|  49 |  74 |  58 |  50 | 118 |
+-----+-----+-----+-----+-----+
|  28 |  21 |  88 |  84 |  89 |
+-----+-----+-----+-----+-----+
```

		<pre>Slice 4: +-----+-----+-----+-----+-----+ | 112 | 87 | 55 | 86 | 11 | +-----+-----+-----+-----+-----+ | 25 | 46 | 65 | 119 | 43 | +-----+-----+-----+-----+-----+ | 67 | 114 | 71 | 103 | 94 | +-----+-----+-----+-----+-----+ | 37 | 93 | 124 | 61 | 24 | +-----+-----+-----+-----+-----+ | 35 | 73 | 68 | 53 | 111 | +-----+-----+-----+-----+-----+ Slice 5: +-----+-----+-----+-----+-----+ | 90 | 60 | 17 | 32 | 113 | +-----+-----+-----+-----+-----+ | 9 | 79 | 99 | 101 | 83 | +-----+-----+-----+-----+-----+ | 106 | 82 | 104 | 29 | 108 | +-----+-----+-----+-----+-----+ | 115 | 42 | 31 | 1 | 76 | +-----+-----+-----+-----+-----+ | 123 | 80 | 81 | 34 | 63 | +-----+-----+-----+-----+-----+</pre>Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   2 |   8 |  33 |   3 |   4 |
+-----+-----+-----+-----+-----+
|  13 |   1 |  12 |  16 |  11 |
+-----+-----+-----+-----+-----+
|  14 |  35 |  20 |   7 |  15 |
+-----+-----+-----+-----+-----+
|  24 |   5 |   6 |  21 |  18 |
+-----+-----+-----+-----+-----+
|  17 |  19 |  10 |  28 |   9 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  26 |  22 |  38 |  30 |  34 |
+-----+-----+-----+-----+-----+
|  29 |  27 |  49 |  36 |  32 |
+-----+-----+-----+-----+-----+
|  46 |  50 |  43 |  44 |  59 |
+-----+-----+-----+-----+-----+
|  41 |  52 |  47 |  40 |  42 |
+-----+-----+-----+-----+-----+
|  37 |  48 |  92 |  23 |  54 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  55 |  25 |  70 |  56 |  57 |
+-----+-----+-----+-----+-----+
|  51 |  65 |  53 |  45 |  61 |
+-----+-----+-----+-----+-----+
|  71 |  39 |  62 |  81 |  31 |
+-----+-----+-----+-----+-----+
|  64 |  76 |  58 |  74 |  72 |
+-----+-----+-----+-----+-----+
|  60 |  84 |  75 |  73 | 110 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td></td><td></td><td><pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 67 | 63 | 68 | 87 | 83 |
+-----+-----+-----+-----+-----+
| 88 | 91 | 79 | 101 | 82 |
+-----+-----+-----+-----+-----+
| 85 | 104 | 66 | 89 | 86 |
+-----+-----+-----+-----+-----+
| 93 | 77 | 80 | 96 | 97 |
+-----+-----+-----+-----+-----+
| 95 | 90 | 102 | 106 | 107 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 105 | 98 | 78 | 100 | 111 |
+-----+-----+-----+-----+-----+
| 69 | 125 | 117 | 108 | 115 |
+-----+-----+-----+-----+-----+
| 124 | 116 | 94 | 112 | 109 |
+-----+-----+-----+-----+-----+
| 114 | 118 | 122 | 99 | 120 |
+-----+-----+-----+-----+-----+
| 123 | 119 | 103 | 121 | 113 |
+-----+-----+-----+-----+-----+
</pre>
Time: 11.48
Best Cost: 1056</td></tr>
<tr><td></td><td>2</td><td>Initial State
Initial Fitness: 4206</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  46 |  95 |  33 |  12 |  21 |
+-----+-----+-----+-----+-----+
|   5 |  13 |  65 | 107 |  20 |
+-----+-----+-----+-----+-----+
|   3 |  42 |  93 |  39 | 117 |
+-----+-----+-----+-----+-----+
|  79 |   7 |  49 |  70 |  83 |
+-----+-----+-----+-----+-----+
|  41 |  92 |  18 |  84 |  35 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  67 | 118 |  99 |  16 |  22 |
+-----+-----+-----+-----+-----+
|  27 |  66 |  78 |  14 |  29 |
+-----+-----+-----+-----+-----+
|  62 |  44 |  55 |   2 |  43 |
+-----+-----+-----+-----+-----+
|  11 |   8 |  77 |   9 |  40 |
+-----+-----+-----+-----+-----+
| 105 |  88 |  37 |   4 |  59 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  52 | 122 |  94 |  74 |  54 |
+-----+-----+-----+-----+-----+
|  61 | 125 | 123 |  47 |  71 |
+-----+-----+-----+-----+-----+
|  72 |  31 |  17 |  30 |  24 |
+-----+-----+-----+-----+-----+
|  90 |  80 |  57 |  60 |  86 |
+-----+-----+-----+-----+-----+
|  75 | 108 |  53 | 110 |  58 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  10 | 113 | 119 |  34 |  76 |
+-----+-----+-----+-----+-----+
| 120 | 116 |  28 |  68 | 121 |
+-----+-----+-----+-----+-----+
|  64 |  36 |  91 |   1 |  15 |
+-----+-----+-----+-----+-----+
|  87 |  45 | 104 |  81 |  73 |
+-----+-----+-----+-----+-----+
| 112 |  56 |  25 |  32 | 102 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 103 |  26 |  23 |  82 |  51 |
+-----+-----+-----+-----+-----+
|  38 | 115 | 111 |  89 |  19 |
+-----+-----+-----+-----+-----+
| 106 |  50 |  69 |  98 | 101 |
+-----+-----+-----+-----+-----+
|  96 |  63 | 100 | 124 |   6 |
+-----+-----+-----+-----+-----+
| 114 |  48 |  97 |  85 | 109 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   7 |   3 |   8 |   1 |  31 |
+-----+-----+-----+-----+-----+
|  12 |   5 |  20 |  15 |   4 |
+-----+-----+-----+-----+-----+
|   9 |  10 |  13 |   2 |  25 |
+-----+-----+-----+-----+-----+
|  41 |  36 |  18 |  21 |  47 |
+-----+-----+-----+-----+-----+
|   6 |  19 |  48 |  24 |  14 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  11 |  29 |  16 |  28 |  32 |
+-----+-----+-----+-----+-----+
|  22 |  39 |  26 |  40 |  60 |
+-----+-----+-----+-----+-----+
|  35 |  49 |  27 |  33 |  30 |
+-----+-----+-----+-----+-----+
|  23 |  17 |  43 |  70 |  66 |
+-----+-----+-----+-----+-----+
|  46 |  54 |  37 |  50 |  53 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  52 |  62 |  45 |  59 |  67 |
+-----+-----+-----+-----+-----+
| 103 |  58 |  34 |  64 |  42 |
+-----+-----+-----+-----+-----+
|  55 |  57 |  65 |  63 |  56 |
+-----+-----+-----+-----+-----+
|  76 |  51 |  88 |  61 |  79 |
+-----+-----+-----+-----+-----+
|  73 |  38 |  78 |  74 |  80 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td></td>
<td>Slice 4:
+-----+-----+-----+-----+-----+
| 44 | 68 | 81 | 72 | 90 |
+-----+-----+-----+-----+-----+
| 69 | 87 | 71 | 92 | 82 |
+-----+-----+-----+-----+-----+
| 83 | 85 | 100 | 75 | 91 |
+-----+-----+-----+-----+-----+
| 106 | 89 | 84 | 86 | 93 |
+-----+-----+-----+-----+-----+
| 98 | 109 | 77 | 97 | 110 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 104 | 94 | 105 | 112 | 95 |
+-----+-----+-----+-----+-----+
| 117 | 101 | 108 | 96 | 99 |
+-----+-----+-----+-----+-----+
| 125 | 113 | 111 | 118 | 116 |
+-----+-----+-----+-----+-----+
| 115 | 102 | 120 | 114 | 119 |
+-----+-----+-----+-----+-----+
| 123 | 121 | 124 | 122 | 107 |
+-----+-----+-----+-----+-----+

Time: 11.42
Best Cost: 1086</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Initial State
Initial Fitness: 3936</td>
</tr>
</table>

		Slice 1: +-----+-----+-----+-----+-----+ | 28 | 13 | 121 | 70 | 3 | +-----+-----+-----+-----+-----+ | 37 | 96 | 24 | 15 | 7 | +-----+-----+-----+-----+-----+ | 8 | 31 | 33 | 60 | 4 | +-----+-----+-----+-----+-----+ | 67 | 119 | 54 | 27 | 51 | +-----+-----+-----+-----+-----+ | 26 | 5 | 93 | 74 | 50 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 107 | 83 | 6 | 80 | 47 | +-----+-----+-----+-----+-----+ | 68 | 2 | 43 | 56 | 36 | +-----+-----+-----+-----+-----+ | 20 | 94 | 125 | 29 | 77 | +-----+-----+-----+-----+-----+ | 35 | 49 | 88 | 65 | 55 | +-----+-----+-----+-----+-----+ | 44 | 98 | 66 | 16 | 113 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 22 | 100 | 21 | 25 | 52 | +-----+-----+-----+-----+-----+ | 41 | 14 | 45 | 1 | 64 | +-----+-----+-----+-----+-----+ | 91 | 86 | 38 | 11 | 34 | +-----+-----+-----+-----+-----+ | 102 | 72 | 106 | 30 | 109 | +-----+-----+-----+-----+-----+ | 57 | 10 | 110 | 105 | 63 | +-----+-----+-----+-----+-----+

```
Slice 4:
+-----+-----+-----+-----+-----+
|  32 |  62 | 103 |  84 |  85 |
+-----+-----+-----+-----+-----+
|  59 |  81 |  48 | 118 |  46 |
+-----+-----+-----+-----+-----+
|  39 |  82 |   9 |  90 |  19 |
+-----+-----+-----+-----+-----+
|  99 | 123 |  17 |  42 |  23 |
+-----+-----+-----+-----+-----+
| 104 |  69 | 120 | 108 |  92 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  79 | 117 |  76 |  97 |  87 |
+-----+-----+-----+-----+-----+
|  53 |  40 | 112 |  58 |  12 |
+-----+-----+-----+-----+-----+
|  73 | 115 |  95 | 111 |  71 |
+-----+-----+-----+-----+-----+
| 124 |  89 |  75 |  61 | 114 |
+-----+-----+-----+-----+-----+
|  18 | 122 | 116 | 101 |  78 |
+-----+-----+-----+-----+-----+
```

Final State

		<pre>Slice 1: +-----+-----+-----+-----+-----+ | 2 | 18 | 7 | 1 | 11 | +-----+-----+-----+-----+-----+ | 3 | 10 | 8 | 5 | 12 | +-----+-----+-----+-----+-----+ | 6 | 13 | 14 | 19 | 72 | +-----+-----+-----+-----+-----+ | 16 | 28 | 20 | 21 | 15 | +-----+-----+-----+-----+-----+ | 33 | 35 | 48 | 22 | 27 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 23 | 32 | 31 | 42 | 9 | +-----+-----+-----+-----+-----+ | 24 | 30 | 26 | 17 | 29 | +-----+-----+-----+-----+-----+ | 51 | 25 | 39 | 41 | 55 | +-----+-----+-----+-----+-----+ | 61 | 38 | 40 | 34 | 46 | +-----+-----+-----+-----+-----+ | 43 | 52 | 54 | 49 | 53 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 44 | 58 | 60 | 76 | 93 | +-----+-----+-----+-----+-----+ | 80 | 37 | 56 | 47 | 50 | +-----+-----+-----+-----+-----+ | 82 | 64 | 57 | 83 | 63 | +-----+-----+-----+-----+-----+ | 68 | 66 | 67 | 74 | 36 | +-----+-----+-----+-----+-----+ | 78 | 59 | 62 | 65 | 75 | +-----+-----+-----+-----+-----+</pre>

<table>
<tr><td></td><td></td><td>Slice 4:
+-----+-----+-----+-----+-----+
| 84 | 79 | 4 | 77 | 73 |
+-----+-----+-----+-----+-----+
| 70 | 106 | 89 | 96 | 71 |
+-----+-----+-----+-----+-----+
| 81 | 88 | 86 | 87 | 102 |
+-----+-----+-----+-----+-----+
| 69 | 124 | 92 | 103 | 91 |
+-----+-----+-----+-----+-----+
| 97 | 95 | 85 | 90 | 105 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 45 | 100 | 98 | 104 | 111 |
+-----+-----+-----+-----+-----+
| 108 | 121 | 109 | 94 | 110 |
+-----+-----+-----+-----+-----+
| 114 | 99 | 113 | 119 | 112 |
+-----+-----+-----+-----+-----+
| 107 | 117 | 125 | 120 | 123 |
+-----+-----+-----+-----+-----+
| 115 | 122 | 116 | 101 | 118 |
+-----+-----+-----+-----+-----+

Time: 11.40
Best Cost: 1118</td></tr>
<tr><td>1500</td><td>1</td><td>Initial State
Initial Fitness: 4086</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
| 119 |  27 |   6 |  62 |  45 |
+-----+-----+-----+-----+-----+
|  24 |  22 |  59 |   8 |  97 |
+-----+-----+-----+-----+-----+
|   9 |  94 |  57 | 111 |  14 |
+-----+-----+-----+-----+-----+
|  56 |  63 |  29 |  92 | 115 |
+-----+-----+-----+-----+-----+
|  44 |  25 |   4 |  48 |  17 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  15 |   5 |  13 |  30 |  41 |
+-----+-----+-----+-----+-----+
|  95 |  74 | 100 |  10 |  38 |
+-----+-----+-----+-----+-----+
|  64 | 101 |  33 |  35 |  11 |
+-----+-----+-----+-----+-----+
|  36 | 107 |  54 |  93 |  20 |
+-----+-----+-----+-----+-----+
|  23 | 110 |  42 |  58 | 124 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  21 |  34 |  67 |  68 |  61 |
+-----+-----+-----+-----+-----+
|  66 |  80 |  65 | 117 |  53 |
+-----+-----+-----+-----+-----+
|   3 |  16 |   1 |  39 |  96 |
+-----+-----+-----+-----+-----+
| 103 |  51 |  47 |  89 |  43 |
+-----+-----+-----+-----+-----+
| 104 |  69 | 106 |  32 |  71 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  79 |   2 |  78 | 113 |  72 |
+-----+-----+-----+-----+-----+
|  46 |  81 | 116 | 123 |  55 |
+-----+-----+-----+-----+-----+
| 125 |  82 |  19 |  70 | 108 |
+-----+-----+-----+-----+-----+
|  87 | 118 | 102 |  83 | 114 |
+-----+-----+-----+-----+-----+
|   7 |  91 |  26 |  76 |  98 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 122 | 109 |  60 |  37 |  77 |
+-----+-----+-----+-----+-----+
|  28 |  12 |  49 |  18 |  86 |
+-----+-----+-----+-----+-----+
|  85 | 105 |  50 |  40 | 120 |
+-----+-----+-----+-----+-----+
| 112 |  84 |  99 |  31 |  88 |
+-----+-----+-----+-----+-----+
| 121 |  90 |  73 |  75 |  52 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   1 |  70 |   5 |   9 |   2 |
+-----+-----+-----+-----+-----+
|  16 |  18 |  17 |  19 |   3 |
+-----+-----+-----+-----+-----+
|   4 |  12 |  14 |  29 |   6 |
+-----+-----+-----+-----+-----+
|  25 |   8 |   7 |  42 |  15 |
+-----+-----+-----+-----+-----+
|  32 |  24 |  23 |  22 |  27 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  26 |  21 |  31 |  20 |  44 |
+-----+-----+-----+-----+-----+
|  30 |  10 |  11 |  33 |  28 |
+-----+-----+-----+-----+-----+
|  35 |  34 |  49 |  40 |  38 |
+-----+-----+-----+-----+-----+
|  43 |  37 |  36 |  39 |  52 |
+-----+-----+-----+-----+-----+
|  74 | 107 |  46 |  50 |  48 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  41 |  61 |  51 |  68 |  54 |
+-----+-----+-----+-----+-----+
| 123 |  67 |  13 |  60 |  47 |
+-----+-----+-----+-----+-----+
|  53 |  45 |  56 |  63 |  64 |
+-----+-----+-----+-----+-----+
|  66 |  57 |  59 |  65 |  71 |
+-----+-----+-----+-----+-----+
|  93 |  55 |  75 |  62 |  77 |
+-----+-----+-----+-----+-----+
```

<table>
<tr><td rowspan="2"></td><td></td><td>Slice 4:
+-----+-----+-----+-----+-----+
| 83 | 90 | 76 | 78 | 72 |
+-----+-----+-----+-----+-----+
| 84 | 94 | 82 | 79 | 85 |
+-----+-----+-----+-----+-----+
| 87 | 86 | 106 | 91 | 58 |
+-----+-----+-----+-----+-----+
| 73 | 69 | 92 | 102 | 88 |
+-----+-----+-----+-----+-----+
| 95 | 98 | 81 | 100 | 101 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 89 | 96 | 116 | 80 | 125 |
+-----+-----+-----+-----+-----+
| 105 | 109 | 111 | 113 | 114 |
+-----+-----+-----+-----+-----+
| 115 | 103 | 108 | 120 | 99 |
+-----+-----+-----+-----+-----+
| 122 | 124 | 97 | 104 | 121 |
+-----+-----+-----+-----+-----+
| 118 | 119 | 117 | 110 | 112 |
+-----+-----+-----+-----+-----+

Time: 19.78
Best Cost: 1140</td></tr>
<tr><td>2</td><td>Initial State
Initial Fitness: 4204</td></tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  86 |  26 |  58 |  40 |  17 |
+-----+-----+-----+-----+-----+
|  25 |  13 | 122 |  54 |  77 |
+-----+-----+-----+-----+-----+
|   4 |  16 |  11 | 114 |  12 |
+-----+-----+-----+-----+-----+
|  66 |  97 |  14 | 107 |  21 |
+-----+-----+-----+-----+-----+
|  31 |   5 |  82 |  22 | 106 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  78 |   6 |  80 | 102 |  83 |
+-----+-----+-----+-----+-----+
|  23 |  67 |  88 | 111 |  20 |
+-----+-----+-----+-----+-----+
|   7 |   9 |  70 |  47 | 101 |
+-----+-----+-----+-----+-----+
| 123 |  87 | 115 |  37 |  34 |
+-----+-----+-----+-----+-----+
|  49 |  38 |  42 |  35 |  30 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  19 |  95 |  24 |   1 | 124 |
+-----+-----+-----+-----+-----+
|  32 |  53 | 110 |  33 |  65 |
+-----+-----+-----+-----+-----+
|  18 |  46 |  29 |  45 | 118 |
+-----+-----+-----+-----+-----+
| 103 |  48 |  81 |  43 |  68 |
+-----+-----+-----+-----+-----+
| 109 |  10 |  79 |  52 |  73 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  15 |  57 |  36 |  90 |  64 |
+-----+-----+-----+-----+-----+
|  71 |  89 |  41 | 108 |  62 |
+-----+-----+-----+-----+-----+
|  94 |   3 |  69 |  75 |  63 |
+-----+-----+-----+-----+-----+
| 119 |  96 |  61 |  76 |  93 |
+-----+-----+-----+-----+-----+
| 117 |  74 |  85 | 113 |  44 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|   2 | 104 |  51 |  50 | 120 |
+-----+-----+-----+-----+-----+
|  59 |  72 |  27 | 125 |  55 |
+-----+-----+-----+-----+-----+
|  39 |  84 |  91 | 105 |  60 |
+-----+-----+-----+-----+-----+
|  28 |  99 |  92 | 112 | 100 |
+-----+-----+-----+-----+-----+
|   8 | 121 |  98 | 116 |  56 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|  14 |   1 |   3 |  46 |  12 |
+-----+-----+-----+-----+-----+
|  18 |   4 |   2 |  13 |   8 |
+-----+-----+-----+-----+-----+
|  20 |   9 |  21 |  11 |  15 |
+-----+-----+-----+-----+-----+
|  31 |  17 |  29 |   7 |  22 |
+-----+-----+-----+-----+-----+
|  24 |   6 |   5 |  33 |  35 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  28 |  27 |  16 |  66 |  42 |
+-----+-----+-----+-----+-----+
|  19 |  36 |  25 |  23 |  38 |
+-----+-----+-----+-----+-----+
|  34 |  61 |  30 |  10 |  47 |
+-----+-----+-----+-----+-----+
|  43 |  41 |  37 |  26 |  44 |
+-----+-----+-----+-----+-----+
|  48 |  50 |  45 |  32 | 105 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  51 |  39 |  59 |  70 |  53 |
+-----+-----+-----+-----+-----+
|  52 |  65 |  68 |  57 |  56 |
+-----+-----+-----+-----+-----+
|  54 |  58 |  55 |  40 |  64 |
+-----+-----+-----+-----+-----+
|  62 |  75 |  60 |  87 |  67 |
+-----+-----+-----+-----+-----+
|  78 |  49 |  71 |  76 |  82 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td rowspan="2"></td>
<td></td>
<td>
Slice 4:
<table>
<tr><td>90</td><td>79</td><td>94</td><td>80</td><td>63</td></tr>
<tr><td>83</td><td>72</td><td>100</td><td>98</td><td>69</td></tr>
<tr><td>88</td><td>118</td><td>81</td><td>101</td><td>109</td></tr>
<tr><td>77</td><td>86</td><td>95</td><td>73</td><td>84</td></tr>
<tr><td>120</td><td>92</td><td>91</td><td>97</td><td>93</td></tr>
</table>
Slice 5:
<table>
<tr><td>102</td><td>85</td><td>74</td><td>106</td><td>104</td></tr>
<tr><td>112</td><td>107</td><td>103</td><td>113</td><td>108</td></tr>
<tr><td>110</td><td>99</td><td>114</td><td>96</td><td>117</td></tr>
<tr><td>123</td><td>115</td><td>119</td><td>125</td><td>89</td></tr>
<tr><td>116</td><td>121</td><td>122</td><td>111</td><td>124</td></tr>
</table>

Time: 16.72
Best Cost: 1128
</td>
</tr>
<tr>
<td>3</td>
<td>Initial State
Initial Fitness: 4322</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
| 102 |  60 | 116 | 124 |  39 |
+-----+-----+-----+-----+-----+
|  11 |  20 |  12 |  19 |  21 |
+-----+-----+-----+-----+-----+
|  61 |  71 |  22 | 106 | 113 |
+-----+-----+-----+-----+-----+
|  27 |  26 |  18 |  16 |  70 |
+-----+-----+-----+-----+-----+
|  41 |  40 |  48 |  81 |   2 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
| 103 |  17 |  52 | 104 |   4 |
+-----+-----+-----+-----+-----+
|  31 |  77 |  15 |  87 |  14 |
+-----+-----+-----+-----+-----+
|  10 |  58 |  46 |  63 |   1 |
+-----+-----+-----+-----+-----+
|  38 |  78 |  34 |  33 |  84 |
+-----+-----+-----+-----+-----+
|  67 | 117 |  57 |  59 |  83 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  43 |  74 | 115 |  25 |  76 |
+-----+-----+-----+-----+-----+
| 120 |   8 | 109 |   5 |   6 |
+-----+-----+-----+-----+-----+
|  32 |  36 |  53 |  72 | 125 |
+-----+-----+-----+-----+-----+
| 122 |  80 |  86 |  79 |  92 |
+-----+-----+-----+-----+-----+
|  23 |  75 |  56 |  28 | 100 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
| 112 | 110 |  97 |  68 |  47 |
+-----+-----+-----+-----+-----+
|  88 |  90 |  73 |  13 |  99 |
+-----+-----+-----+-----+-----+
| 105 |  54 |  42 | 123 |  49 |
+-----+-----+-----+-----+-----+
|  55 |  94 |  45 |  98 |  91 |
+-----+-----+-----+-----+-----+
|  69 |  30 |  50 |  64 | 114 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  95 | 121 |  35 |  66 | 108 |
+-----+-----+-----+-----+-----+
|  96 |  89 |  44 |  24 |  29 |
+-----+-----+-----+-----+-----+
|   3 |   9 | 111 | 101 |  37 |
+-----+-----+-----+-----+-----+
|  82 | 119 |  65 | 107 | 118 |
+-----+-----+-----+-----+-----+
|  62 |  93 |   7 |  85 |  51 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   1 |   5 |   7 |  13 |  16 |
+-----+-----+-----+-----+-----+
|   4 |   3 |   8 |   6 |   2 |
+-----+-----+-----+-----+-----+
|  11 |  20 |  15 |  67 |   9 |
+-----+-----+-----+-----+-----+
|  12 |  18 |  32 |  10 |  25 |
+-----+-----+-----+-----+-----+
|  27 |  17 |  23 |  64 |  14 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  26 |  39 |  21 |  28 |  31 |
+-----+-----+-----+-----+-----+
|  22 |  19 |  42 |  34 |  51 |
+-----+-----+-----+-----+-----+
|  29 |  45 |  37 |  70 |  40 |
+-----+-----+-----+-----+-----+
|  38 |  30 |  44 |  52 |  77 |
+-----+-----+-----+-----+-----+
|  36 |  46 |  48 |  57 |  24 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  49 |  58 |  47 |  35 |  65 |
+-----+-----+-----+-----+-----+
|  73 |  43 |  63 |  33 |  74 |
+-----+-----+-----+-----+-----+
|  66 |  53 |  54 |  56 |  55 |
+-----+-----+-----+-----+-----+
|  41 |  68 |  61 |  69 |  71 |
+-----+-----+-----+-----+-----+
|  90 |  50 | 116 |  75 |  80 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td></td>
<td>
<pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 79 | 59 | 88 | 83 | 62 |
+-----+-----+-----+-----+-----+
| 82 | 96 | 78 | 72 | 99 |
+-----+-----+-----+-----+-----+
| 84 | 89 | 87 | 76 | 94 |
+-----+-----+-----+-----+-----+
| 81 | 107 | 60 | 86 | 93 |
+-----+-----+-----+-----+-----+
| 105 | 98 | 109 | 91 | 111 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 101 | 102 | 123 | 92 | 100 |
+-----+-----+-----+-----+-----+
| 119 | 110 | 125 | 85 | 97 |
+-----+-----+-----+-----+-----+
| 114 | 106 | 115 | 122 | 108 |
+-----+-----+-----+-----+-----+
| 104 | 120 | 113 | 95 | 103 |
+-----+-----+-----+-----+-----+
| 118 | 124 | 112 | 121 | 117 |
+-----+-----+-----+-----+-----+
</pre>
Time: 16.97
Best Cost: 1158
</td>
</tr>
<tr>
<td>2000</td>
<td>1</td>
<td>Initial State
Initial Fitness: 4154</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
| 102 |  39 |   7 |  49 | 117 |
+-----+-----+-----+-----+-----+
|   3 |  10 |  58 | 118 |   9 |
+-----+-----+-----+-----+-----+
|  16 |  43 |  91 |  57 | 109 |
+-----+-----+-----+-----+-----+
|  46 |  54 |  15 |  64 |  61 |
+-----+-----+-----+-----+-----+
|  53 |  18 |  60 | 101 |  34 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  28 |  98 |  35 |  76 | 111 |
+-----+-----+-----+-----+-----+
|  69 |  56 |  37 |  83 |  67 |
+-----+-----+-----+-----+-----+
|  30 |  62 | 112 |  75 |  29 |
+-----+-----+-----+-----+-----+
|  11 |  41 |  40 |  12 |  31 |
+-----+-----+-----+-----+-----+
|  68 |  13 |  42 |   1 |  36 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  14 |  27 |  25 |  99 |   2 |
+-----+-----+-----+-----+-----+
|  90 |  65 |  50 |  66 |  19 |
+-----+-----+-----+-----+-----+
|  71 | 108 |  47 |  70 |  80 |
+-----+-----+-----+-----+-----+
|  21 |  26 |  87 |  72 |  23 |
+-----+-----+-----+-----+-----+
| 110 |  86 |  22 |  24 | 121 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
| 120 |  51 |  59 | 106 |  17 |
+-----+-----+-----+-----+-----+
|   5 |  88 |  44 |  95 |  33 |
+-----+-----+-----+-----+-----+
| 123 |  92 |   4 |  38 | 104 |
+-----+-----+-----+-----+-----+
| 125 |   6 |  78 |   8 | 103 |
+-----+-----+-----+-----+-----+
| 116 |  52 |  20 |  32 | 105 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 124 |  74 |  81 |  85 | 115 |
+-----+-----+-----+-----+-----+
|  73 |  79 | 100 |  82 |  77 |
+-----+-----+-----+-----+-----+
|  97 |  45 | 122 |  96 | 113 |
+-----+-----+-----+-----+-----+
|  84 |  55 |  48 |  63 | 114 |
+-----+-----+-----+-----+-----+
|  94 |  93 | 119 |  89 | 107 |
+-----+-----+-----+-----+-----+
```

Final State

		<pre>Slice 1: +-----+-----+-----+-----+-----+ | 8 | 1 | 16 | 15 | 11 | +-----+-----+-----+-----+-----+ | 9 | 4 | 24 | 5 | 2 | +-----+-----+-----+-----+-----+ | 13 | 7 | 40 | 19 | 33 | +-----+-----+-----+-----+-----+ | 22 | 21 | 26 | 23 | 17 | +-----+-----+-----+-----+-----+ | 6 | 18 | 14 | 43 | 20 | +-----+-----+-----+-----+-----+ Slice 2: +-----+-----+-----+-----+-----+ | 34 | 3 | 27 | 42 | 38 | +-----+-----+-----+-----+-----+ | 30 | 32 | 28 | 31 | 53 | +-----+-----+-----+-----+-----+ | 10 | 29 | 35 | 12 | 60 | +-----+-----+-----+-----+-----+ | 41 | 45 | 36 | 59 | 61 | +-----+-----+-----+-----+-----+ | 49 | 25 | 44 | 55 | 46 | +-----+-----+-----+-----+-----+ Slice 3: +-----+-----+-----+-----+-----+ | 51 | 50 | 48 | 47 | 57 | +-----+-----+-----+-----+-----+ | 52 | 77 | 58 | 54 | 67 | +-----+-----+-----+-----+-----+ | 56 | 62 | 69 | 68 | 74 | +-----+-----+-----+-----+-----+ | 66 | 63 | 73 | 91 | 65 | +-----+-----+-----+-----+-----+ | 64 | 39 | 72 | 124 | 81 | +-----+-----+-----+-----+-----+</pre>

<table>
<tr>
<td rowspan="2"></td>
<td></td>
<td>

```
Slice 4:
+-----+-----+-----+-----+-----+
|  79 |  84 |  70 |  94 |  80 |
+-----+-----+-----+-----+-----+
|  98 |  87 |  78 |  93 |  86 |
+-----+-----+-----+-----+-----+
|  75 | 106 |  37 |  90 |  89 |
+-----+-----+-----+-----+-----+
| 109 |  71 |  85 |  96 |  92 |
+-----+-----+-----+-----+-----+
| 100 |  99 |  97 | 101 |  83 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  82 |  76 | 119 |  88 | 102 |
+-----+-----+-----+-----+-----+
| 123 |  95 | 108 | 105 | 104 |
+-----+-----+-----+-----+-----+
| 116 | 107 | 103 | 120 | 117 |
+-----+-----+-----+-----+-----+
| 110 | 118 | 114 | 125 | 115 |
+-----+-----+-----+-----+-----+
| 121 | 111 | 122 | 112 | 113 |
+-----+-----+-----+-----+-----+
```

Time: 24.84
Best Cost: 1096</td>
</tr>
<tr>
<td>2</td>
<td>Initial State
Initial Fitness: 4216</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  25 |  40 |  72 |  33 |  23 |
+-----+-----+-----+-----+-----+
|  79 |  93 |  32 |   8 |  13 |
+-----+-----+-----+-----+-----+
|  84 |  14 |  41 |  21 | 125 |
+-----+-----+-----+-----+-----+
|  16 | 109 |  56 | 111 |  51 |
+-----+-----+-----+-----+-----+
|  61 |  36 |  52 |  62 |  98 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  24 | 103 |  96 |  35 |   2 |
+-----+-----+-----+-----+-----+
|  31 |   9 |  75 |  37 |  66 |
+-----+-----+-----+-----+-----+
|  47 | 122 |  50 |   7 |  34 |
+-----+-----+-----+-----+-----+
|  43 |  10 |   5 |  12 | 107 |
+-----+-----+-----+-----+-----+
|  83 |   1 | 106 |  95 |  42 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  17 |  59 |  78 |  54 |  85 |
+-----+-----+-----+-----+-----+
|  69 |  70 |  22 |  87 |  97 |
+-----+-----+-----+-----+-----+
|  38 |  73 |  92 |  39 |  11 |
+-----+-----+-----+-----+-----+
| 113 |  53 |  89 |  58 |  76 |
+-----+-----+-----+-----+-----+
| 119 |  30 |  90 |   6 |  74 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
| 116 |  29 |  65 |  18 |  20 |
+-----+-----+-----+-----+-----+
| 104 |  19 |  88 | 115 |  26 |
+-----+-----+-----+-----+-----+
|  45 | 105 |  86 |  64 |  67 |
+-----+-----+-----+-----+-----+
|  44 | 108 | 101 |  82 |  80 |
+-----+-----+-----+-----+-----+
| 100 | 102 |   4 |  28 | 114 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  55 |  57 |  81 |  99 |  60 |
+-----+-----+-----+-----+-----+
| 124 |  63 |  94 |  27 |  77 |
+-----+-----+-----+-----+-----+
| 121 | 112 |  71 | 120 |  15 |
+-----+-----+-----+-----+-----+
| 123 | 118 |  68 |  49 |  46 |
+-----+-----+-----+-----+-----+
|  48 |  91 | 110 |   3 | 117 |
+-----+-----+-----+-----+-----+
```

Final State

```
Slice 1:
+-----+-----+-----+-----+-----+
|   1 |  41 |   8 |  11 |   2 |
+-----+-----+-----+-----+-----+
|   5 |   4 |   6 |  27 |  16 |
+-----+-----+-----+-----+-----+
|   7 |  42 |  18 |  14 |   9 |
+-----+-----+-----+-----+-----+
|  10 |  37 |  39 |  17 |  20 |
+-----+-----+-----+-----+-----+
|  21 |  13 |  12 |   3 |  23 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  24 |  26 |  40 |  31 |  35 |
+-----+-----+-----+-----+-----+
|  15 |  19 |  32 |  34 |  22 |
+-----+-----+-----+-----+-----+
|  36 |  28 |  38 |  59 |  25 |
+-----+-----+-----+-----+-----+
|  67 |  29 |  47 |  46 |  50 |
+-----+-----+-----+-----+-----+
|  52 |  45 |  30 |  64 |  58 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  49 |  44 |  57 |  60 |  61 |
+-----+-----+-----+-----+-----+
|  53 |  51 |  82 |  79 |  62 |
+-----+-----+-----+-----+-----+
|  33 |  55 |  68 |  43 |  83 |
+-----+-----+-----+-----+-----+
|  48 |  77 |  63 |  76 |  71 |
+-----+-----+-----+-----+-----+
|  80 |  75 |  73 | 101 |  78 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td rowspan="2"></td>
<td></td>
<td>
<pre>
Slice 4:
+-----+-----+-----+-----+-----+
| 70 | 93 | 72 | 91 | 54 |
+-----+-----+-----+-----+-----+
| 56 | 74 | 69 | 65 | 108 |
+-----+-----+-----+-----+-----+
| 85 | 81 | 92 | 104 | 118 |
+-----+-----+-----+-----+-----+
| 84 | 88 | 103 | 89 | 87 |
+-----+-----+-----+-----+-----+
| 100 | 125 | 116 | 66 | 105 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
| 99 | 120 | 97 | 102 | 96 |
+-----+-----+-----+-----+-----+
| 111 | 86 | 95 | 106 | 94 |
+-----+-----+-----+-----+-----+
| 107 | 114 | 90 | 98 | 117 |
+-----+-----+-----+-----+-----+
| 109 | 122 | 113 | 115 | 124 |
+-----+-----+-----+-----+-----+
| 121 | 112 | 119 | 123 | 110 |
+-----+-----+-----+-----+-----+
</pre>
Time: 23.32
Best Cost: 1200
</td>
</tr>
<tr>
<td>3</td>
<td>Initial State
Initial Fitness: 4090</td>
</tr>
</table>

```
Slice 1:
+-----+-----+-----+-----+-----+
|  16 |  91 |  26 |  82 |  60 |
+-----+-----+-----+-----+-----+
| 116 |  61 |  53 |  78 |   7 |
+-----+-----+-----+-----+-----+
|  57 |  46 | 115 |  12 | 102 |
+-----+-----+-----+-----+-----+
|  67 |  31 |   4 |  72 |  19 |
+-----+-----+-----+-----+-----+
|  14 |  45 |  11 |   5 |   9 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  75 |  21 | 113 |   3 |  65 |
+-----+-----+-----+-----+-----+
|  23 |  47 |  68 |   1 |  36 |
+-----+-----+-----+-----+-----+
| 100 |  42 |  27 |  88 | 106 |
+-----+-----+-----+-----+-----+
|   2 | 103 |  38 |  44 |  56 |
+-----+-----+-----+-----+-----+
|  94 |  50 | 117 |  20 |  49 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  54 | 110 |  81 |  13 |  63 |
+-----+-----+-----+-----+-----+
|  71 |  30 |  17 |  24 |   8 |
+-----+-----+-----+-----+-----+
|  10 |  92 | 118 |  28 |  74 |
+-----+-----+-----+-----+-----+
|   6 |  79 | 112 |  51 |  15 |
+-----+-----+-----+-----+-----+
|  43 |  48 | 105 | 101 |  66 |
+-----+-----+-----+-----+-----+
```

<table>
<tr>
<td></td>
<td></td>
<td>

```
Slice 4:
+-----+-----+-----+-----+-----+
| 107 |  62 |  41 |  22 | 121 |
+-----+-----+-----+-----+-----+
|  52 |  90 |  80 |  32 |  35 |
+-----+-----+-----+-----+-----+
|  98 |  18 | 114 |  87 |  37 |
+-----+-----+-----+-----+-----+
|  29 | 108 |  73 |  33 |  69 |
+-----+-----+-----+-----+-----+
| 104 |  89 |  84 |  64 |  59 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  34 |  55 |  95 |  76 |  83 |
+-----+-----+-----+-----+-----+
|  25 | 111 |  96 | 123 |  93 |
+-----+-----+-----+-----+-----+
|  85 | 122 | 119 |  58 |  70 |
+-----+-----+-----+-----+-----+
|  99 |  39 |  86 | 109 |  97 |
+-----+-----+-----+-----+-----+
| 125 |  77 |  40 | 124 | 120 |
+-----+-----+-----+-----+-----+
```

Final State
</td>
</tr>
</table>

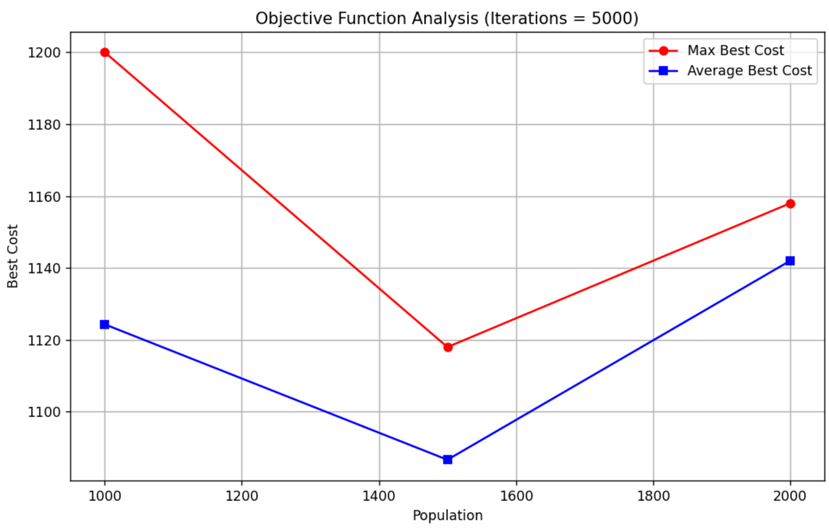
		Slice 1: (see below)

```
Slice 1:
+-----+-----+-----+-----+-----+
|  11 |   3 |   6 |  67 |   4 |
+-----+-----+-----+-----+-----+
|  14 |   1 |   2 |  29 |  15 |
+-----+-----+-----+-----+-----+
|   9 |  10 |   5 |   8 |  22 |
+-----+-----+-----+-----+-----+
|  13 |  27 |  12 |  21 |  23 |
+-----+-----+-----+-----+-----+
|  19 |  51 |  16 |  25 |  36 |
+-----+-----+-----+-----+-----+

Slice 2:
+-----+-----+-----+-----+-----+
|  34 |  24 |  41 |  28 |  31 |
+-----+-----+-----+-----+-----+
|  40 |  20 |  17 |  30 |  35 |
+-----+-----+-----+-----+-----+
|  18 |  46 |  37 |  53 |   7 |
+-----+-----+-----+-----+-----+
|  52 |  38 |  39 |  69 |  54 |
+-----+-----+-----+-----+-----+
|  62 |  45 |  32 |  48 |  42 |
+-----+-----+-----+-----+-----+

Slice 3:
+-----+-----+-----+-----+-----+
|  47 |  63 |  26 |  33 |  78 |
+-----+-----+-----+-----+-----+
|  49 |  44 |  61 |  64 |  55 |
+-----+-----+-----+-----+-----+
|  58 |  59 |  93 |  60 |  79 |
+-----+-----+-----+-----+-----+
|  56 |  75 |  57 |  66 |  68 |
+-----+-----+-----+-----+-----+
|  43 |  65 |  73 |  90 |  72 |
+-----+-----+-----+-----+-----+
```

```
Slice 4:
+-----+-----+-----+-----+-----+
|  71 |  89 |  83 |  76 | 107 |
+-----+-----+-----+-----+-----+
|  80 |  88 |  84 |  50 |  74 |
+-----+-----+-----+-----+-----+
|  87 | 104 |  82 |  70 | 103 |
+-----+-----+-----+-----+-----+
|  91 |  81 |  98 | 108 |  85 |
+-----+-----+-----+-----+-----+
|  86 |  99 | 102 |  77 |  97 |
+-----+-----+-----+-----+-----+

Slice 5:
+-----+-----+-----+-----+-----+
|  96 |  92 |  94 | 109 |  95 |
+-----+-----+-----+-----+-----+
| 101 | 113 | 105 | 106 | 100 |
+-----+-----+-----+-----+-----+
| 122 | 111 | 110 | 115 | 116 |
+-----+-----+-----+-----+-----+
| 119 | 114 | 112 | 117 | 120 |
+-----+-----+-----+-----+-----+
| 125 | 123 | 124 | 121 | 118 |
+-----+-----+-----+-----+-----+
```

Time: 23.41
Best Cost: 1076

Plot



C.3.2 Hasil Analisis

1. **Analisis seberapa dekat *genetic algorithm* dapat mendekati global optima dan faktor yang mempengaruhinya**

Secara teoritis, *genetic algorithm* memiliki kapabilitas untuk mendekati *global optima* dibandingkan algoritma lainnya. Hal ini karena ruang pencarian solusi *genetic algorithm* lebih luas dan algoritma ini juga memastikan solusi terbaik dipertahankan.

Namun, berdasarkan hasil eksperimen, jika dilihat dari segi nilai objektif yang dicapai, algoritma ini menjadi lebih buruk dibandingkan algoritma lainnya. Hal ini terjadi karena kedekatan *genetic algorithm* dengan global optima bergantung pada parameter dan desain algoritma.

2. **Bagaimana perbandingan hasil pencarian tiap-tiap algoritma dengan algoritma local search yang lain?**

Berdasarkan data hasil *genetic algorithm* (GA), algoritma ini mampu menghasilkan solusi dengan *best cost* yang **relatif optimal, terutama pada iterasi tinggi dan populasi besar.** Sebagai contoh, pada populasi 1000 dan iterasi 50.000, GA mencapai *best cost* 748 dalam waktu 135 detik. Dibandingkan dengan *hill-climbing with sideways move*, GA lebih mampu menghindari jebakan optimum lokal dan menemukan solusi global, meskipun waktu komputasinya lebih tinggi.

Simulated annealing (SA), meskipun membutuhkan waktu yang lebih singkat dibanding GA, menunjukkan performa yang kompetitif dalam mencapai solusi mendekati optimal dengan mekanisme probabilistik untuk menghindari optimum lokal. Oleh karena itu, GA lebih unggul dalam eksplorasi solusi kompleks, tetapi SA menawarkan keseimbangan yang baik antara kecepatan dan kualitas solusi.

3. **Perbandingan durasi proses pencarian *genetic algorithm* terhadap algoritma lainnya**

Durasi proses pencarian *genetic algorithm* cenderung **lebih lama** dibandingkan algoritma lainnya. Hal ini disebabkan karena algoritma ini mengevaluasi terlebih dahulu nilai *fitness* untuk setiap individu pada seluruh populasi. Lalu, algoritma ini juga melakukan operasi tambahan seperti *crossover* dan *mutation* yang menyebabkan waktu kopmutasi menjadi lebih lama.

4. **Seberapa konsisten hasil akhir yang didapatkan dari tiap-tiap eksperimen yang dilakukan?**

Hasil akhir dari tiap eksperimen menunjukkan konsistensi yang cukup tinggi **(*moderately high*)**. Hal ini terlihat dari perbedaan nilai *best cost* di setiap percobaan dalam skenario yang sama, yang cenderung berada dalam rentang nilai yang sempit. Misalnya, untuk populasi 1000 dan iterasi 5000, *best cost* berkisar antara 1078 hingga 1154 pada tiga percobaan. Variasi tersebut menunjukkan bahwa meskipun ada fluktuasi kecil, algoritma tetap memberikan solusi yang relatif stabil dan mendekati optimum pada berbagai percobaan.

5. **Analisis pengaruh banyak iterasi dan jumlah populasi terhadap hasil akhir pencarian pada *genetic algorithm***

Saat menjadikan jumlah populasi sebagai variabel kontrol dan melakukan variasi pada banyak iterasi, dapat diketahui bahwa **semakin banyak iterasi, semakin dekat hasil akhir pencarian dengan nilai objektif optimal**. Hal ini dikarenakan setiap iterasi memungkinkan proses eksploitasi dan eksplorasi populasi yang lebih banyak. Pada saat melakukan seleksi individu, semakin banyak iterasi, maka semakin banyak individu dengan dengan nilai *fitness* yang lebih baik. Anak-anak baru yang dihasilkan dari proses mutasi *crossover* juga memiliki peluang lebih besar untuk mendekati solusi optimal. Solusi-solusi yang sebelumnya terjebak di *local optimal* juga dapat ditemukan. Namun, walaupun begitu, **semakin banyak iterasi berarti semakin lama waktu komputasi yang dibutuhkan**.

Di sisi lain, saat menjadikan banyak iterasi sebagai variabel kontrol dan melakukan variasi pada jumlah populasi, dapat diketahui bahwa **semakin banyak populasi, semakin jauh hasil akhir pencarian dengan nilai objektif optimal**. Hal ini dikarenakan jumlah populasi yang terlalu besar akan menjadi lebih sulit untuk dieksploitasi secara efektif. Proses seleksi dan mutasi *crossover* pun akan membutuhkan waktu lebih lama untuk menghasilkan solusi optimal karena tingkat keberagaman yang tinggi. Dengan demikian, untuk mencapai solusi optimal, diperlukan **keseimbangan** antara jumlah populasi dan banyak iterasi. **Semakin banyak populasi, maka semakin banyak iterasi yang dibutuhkan**.

6. Pengaruh variasi *crossover rate, mutation rate*, dan *selection pressure* terhadap performa algoritma

Crossover rate, mutation rate, dan *selection pressure* mempengaruhi performa algoritma secara signifikan. *Crossover rate* yang tinggi akan meningkatkan eksploitasi gen yang baik dari individu yang lebih baik dan akan menghasilkan solusi optimal, namun individu bisa menjadi terlalu mirip (*premature convergence*) yang menyebabkan solusi terjebak di *local optima* jika jumlah populasi yang digunakan terlalu sedikit. Di lain sisi, *crossover rate* yang rendah akan mengurangi tingkat eksploitasi dan mengakibatkan waktu komputasi yang lebih lama.

Dari segi *mutation rate*, tingkat mutasi yang tinggi akan memperluas ruang pencarian solusi dan membantu untuk keluar dari *local optima*. Akan tetapi, individu yang sudah mendekati *global optima* bisa terombak yang mengakibatkan penurunan nilai *fitness*. Di sisi lain, tingkat mutasi yang rendah akan memperkecil ruang pencarian solusi, namun memiliki stabilitas yang lebih tinggi, sehingga dapat mempertahankan individu yang sudah baik.

Dari segi *selection pressure*, semakin tinggi seleksi, maka semakin cepat algoritma mencari solusi optimal. Akan tetapi, sama halnya seperti *crossover rate*, tingkat seleksi yang tinggi dapat menyebabkan individu menjadi terlalu mirip (*premature convergence*) jika jumlah populasi yang digunakan terlalu sedikit. Sedangkan, tingkat seleksi yang rendah dapat memperluas ruang pencarian solusi, namun solusi yang dipilih belum mencapai *global optimal*.

Dengan demikian, dibutuhkan **keseimbangan** untuk ketiga parameter tersebut untuk menghasilkan solusi yang optimal.

7. Bagaimana efisiensi memori algoritma?

Efisiensi memori algoritma *genetic algorithm* (GA) **bergantung pada ukuran populasi dan dimensi data yang diolah**. Setiap individu dalam populasi membutuhkan ruang untuk menyimpan representasi penuh dari kubus 3D serta nilai *fitness*-nya. Misalnya, dengan ukuran kubus $N \times N \times N$ dan populasi yang besar (seperti 1000 atau lebih), **kebutuhan memori dapat meningkat secara signifikan.** Namun, algoritma GA tetap relatif efisien karena hanya menyimpan populasi aktif dan tidak memerlukan penyimpanan riwayat solusi sebelumnya, berbeda dengan algoritma seperti *simulated annealing* yang memerlukan langkah-langkah riwayat tertentu. *Hill-climbing*, di sisi lain, umumnya lebih ringan secara memori karena hanya memproses satu solusi dalam satu waktu.

8. Analisis batasan memori pada *genetic algorithm* yang untuk masalah skala besar

Batasan memori menjadi faktor pembatas utama pada *genetic algorithm*. Algoritma ini cenderung membutuhkan ruang memori yang lebih banyak untuk menjalankan setiap fungsinya. Semakin banyak jumlah populasi, dimensi pencarian, dan semakin kompleks fungsi evaluasi *fitness*-nya, semakin banyak juga ruang memori yang dibutuhkan. Oleh karena itu, untuk memastikan efektivitas algoritma ini pada masalah skala besar, diperlukan optimalisasi representasi data, teknik, dan sumber daya komputasi, seperti menggunakan teknik paralel atau berbasis *cloud*.

9. Bagaimana tingkat *scalability* dari algoritma ketika diterapkan pada data yang jauh lebih besar?

Dari segi *scalability*, *genetic algorithm* (GA) cenderung **lebih efektif pada masalah** besar karena ia bekerja dengan populasi solusi, memungkinkan eksplorasi ruang solusi yang lebih luas. Meskipun GA memiliki biaya komputasi yang tinggi dengan meningkatnya jumlah populasi dan generasi, ia dapat menangani data besar lebih baik dibandingkan algoritma lainnya. Sebaliknya, *hill climbing with sideways move* (HC) memiliki skalabilitas terbatas karena cenderung terjebak di solusi lokal, meskipun *sideways move* sedikit membantu. Pada masalah besar, HC kurang efisien dalam menjelajahi ruang solusi. *Simulated annealing* (SA), meskipun lebih *scalable* dari HC, masih menghadapi tantangan dalam masalah besar karena prosesnya yang lebih lambat dan kebutuhan untuk mengatur suhu yang tepat. Namun, SA lebih mampu menghindari solusi lokal dibandingkan HC.

10. Pengaruh peningkatan waktu dan sumber daya secara linear, eksponensial, dan polinomial terhadap peningkatan kompleksitas masalah

Terdapat tiga faktor yang dapat meningkatkan kebutuhan waktu dan sumber daya, yaitu jumlah populasi, banyak iterasi, dan kompleksitas fungsi evaluasi nilai *fitness*. Hal ini menyebabkan waktu dan sumber daya dapat meningkatkan kompleksitas masalah secara linear, eksponensial, maupun polinomial, tergantung dengan ketiga faktor tersebut pada algoritma.

Kompleksitas masalah dapat meningkat secara linear saat *crossover rate*, *mutation rate*, jumlah populasi, dan banyak iterasi yang tetap dan menggunakan fungsi evaluasi nilai *fitness* yang optimal. Contohnya saat melakukan optimasi *one-max* atau *sphere function*.

Lalu, kompleksitas masalah dapat meningkat secara eksponensial saat ruang pencarian solusi sangat besar, algoritma harus mengevaluasi banyak individu untuk menghindari terjebak di *local optima*, dan saat fungsi evaluasi nilai *fitness* melibatkan perhitungan atau simulasi kompleks. Contohnya adalah optimasi masalah seperti *Travelling Salesman Problem* (TSP) untuk jumlah kota yang besar.

Sedangkan, kompleksitas masalah dapat meningkat secara polinomial saat algoritma menggunakan teknik optimasi yang efisien, seperti *dynamic mutation rates, adaptive crossover*, atau *parallel computing*.

Kesimpulan dan Saran

Setelah melakukan eksperimen pencarian solusi pada *diagonal magic cube* dengan tiga algoritma *local search*: *hill-climbing with sideways move*, *simulated annealing*, dan *genetic algorithm*, dapat disimpulkan bahwa algoritma yang paling efektif adalah ***simulated annealing***.

Keunggulan *simulated annealing* dibandingkan dengan *genetic algorithm* dan *hill-climbing with sideways move* terletak pada kemampuannya untuk menghindari solusi lokal dan mengeksplorasi ruang solusi lebih luas. *Simulated annealing* menggunakan probabilitas untuk menerima solusi yang lebih buruk pada awalnya, memungkinkan pencarian yang lebih dalam dan mengurangi kemungkinan terjebak di titik stagnasi. Dibandingkan *genetic algorithm*, yang lebih kompleks dan bergantung pada pengaturan parameter, *simulated annealing* lebih stabil dan sederhana, memberikan pencarian yang lebih terstruktur dan efektif dalam mencapai solusi optimal pada masalah yang kompleks seperti diagonal *magic cube*.

Hanya saja, dalam kasus tertentu, ***genetic algorithm*** bisa lebih unggul, terutama pada masalah yang membutuhkan eksplorasi ruang solusi yang sangat besar dalam waktu yang lebih singkat. *Genetic algorithm* bekerja dengan populasi solusi yang memungkinkan pencarian lebih luas dan dapat beradaptasi dengan berbagai kondisi, seperti dalam masalah yang memiliki banyak solusi global yang tersebar. Sementara *simulated annealing* sangat efektif dalam menghindari solusi lokal dan menemukan solusi optimal, *genetic algorithm* lebih cocok untuk masalah yang memerlukan eksplorasi paralel dan dapat lebih cepat menemukan solusi yang baik pada ruang solusi yang lebih besar. Dengan kata lain, *simulated annealing* lebih cocok untuk **pencarian solusi optimal jangka panjang**, sementara *genetic* *algorithm* lebih efektif pada masalah dengan ruang solusi yang sangat besar yang memerlukan eksplorasi lebih cepat.

Untuk meningkatkan efektivitas pencarian solusi optimal pada diagonal *magic cube* ke depannya, diperlukan optimalisasi pada setiap algoritma dengan metode-metode terbarukan.

Pembagian Tugas

NIM	Nama	Tugas
18222012	Syakira Fildza Nazhifan	• Membuat template laporan • Merapikan laporan • Membuat deskripsi fungsi/kelas genetic algorithm • Membuat source code genetic algorithm • Melakukan analisis eksperimen genetic algorithm • Membuat kesimpulan dan saran • Membuat deskripsi masalah
18222050	Nicolaas Heru Dreandachrista	• Membuat source code genetic algorithm • Membuat source code visualisasi eksperimen genetic algorithm • Melakukan iterasi dan plot terhadap 2 variasi genetic algorithm • Melakukan analisis eksperimen genetic algorithm • Membuat kesimpulan dan saran
18222134	Ardra Rafif Sahasika	• Membuat source code sideways move • Membuat source code simulated annealing • Membuat visualisasi eksperimen simulated annealing • Melakukan analisis eksperimen simulated annealing • Membuat fitness function
18222137	Timothy Haposan Simanjuntak	• Membuat source code sideways move

		• Membuat source code simulated annealing • Membuat visualisasi eksperimen sideways move • Melakukan analisis eksperimen sideways move • Membuat fitness function

Referensi

Half pandiagonal 5x5x5 magic cube (Shift method). (2014). Magisch Vierkant. https://www.magischvierkant.com/three-dimensional-eng/5x5x5/half-pandiagonal-shift/

Introduction to Hill Climbing | Artificial Intelligence. (2017, December 12). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

Maximilian Weichart. (2023, November 25). *MW - Simulated Annealing*. Maximilian-Weichart.de. https://maximilian-weichart.de/posts/simulated-annealing/

Rachmawati, D., Handrizal, & Batubara, R. (2020). *COMPARISON OF STEEPEST ASCENT HILL CLIMBING ALGORITHM AND BEST FIRST SEARCH ALGORITHM IN DETERMINING THE SHORTEST ROUTE FOR MEDAN TOURISM*. *98*(19).