Optimasi Rute Gym Pokemon: Implementasi Algoritma Travelling Salesman Problem untuk Efisiensi Pengumpulan PokeCoins

Ferdinand Gabe Tua Sinaga - [1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]ferdinandgts5@gmail.com, 13523051@std.stei.itb.ac.id

***Abstract*— Pokemon GO adalah salah satu permainan yang menerapkan konsep augmented reality yang memerlukan seorang pemain untuk mengunjungi lokasi fisik di map seperti Gym untuk mendapat keuntungan seperti *resources* atau *Poke Coins.* Permasalahan optimasi rute dalam mengunjungi multiple Gym ini memiliki karakteristik serupa dengan Traveling Salesman Problem (TSP), sebuah permasalahan optimasi klasik dalam matematika diskrit. Dalam makalah ini kita akan menerapkan dan mengkombinasikan beberapa algoritma TSP untuk menyelesaikan masalah tersebut. Untuk menyelesaikan permasalahan ini peta lokasi *Gym* dalam permainan tersebut akan dimodelkan sebagai graf berbobot dimana lokasi Gym akan dianggap sebagai simpul dan jarak akan dianggap sebagai bobot di sisi selanjutnya algoritma TSP seperti Genetic Algorithm (GA) dan Tabu Search (TS) akan diterapkan untuk menentukan jalur paling optimalnya. Hasil simulasi menunjukkan bahwa pendekatan hybrid GA dan TS menghasilkan rute yang lebih optimal dibandingkan penggunaan algoritma Genetic Algorithm saja. Berdasarkan hasil percobaan penggunaan kombinasi algoritma tersebut meningkatkan solusi paling optimal hingga 19%-26%.**

***Keywords*—Traveling Salesman Problem(TSP), Pokemon Go, Poke Coins, Optimasi jalur**

I. PENDAHULUAN

Pokemon Go adalah salah satu permainan yang memanfaatkan teknologi *augmented reality* (*AR)* dan Global Positioning System (GPS) dalam memainkannya. Teknologi tersebut dipakai oleh Pokemon Go untuk menampilkan Pokemon di dunia nyata dan melacak posisi asli dari seorang player. Tidak seperti kebanyakan permainan, Pokemon GO dirancang untuk memberikan hiburan yang unik dengan mendorong pemain untuk aktif secara fisik, menjelajahi lingkungan sekitar, dan berinteraksi dengan sesama pemain dalam komunitas Pokemon GO.

Gambar 1. Penerapan AR pada Pokemon Go

Sebagai seorang pemain, kita diharuskan menjelajahi area disekitar kita untuk menangkap Pokemon dan mengumpulan *resource* seperti Potion dan Candy, yang penting untuk membuat pokemon kita selalu sehat dan kuat. Dalam permainan ini, pokemon pokemon tersebut menyebar secara merata di seluruh tempat di dunia. Interaksi dengan pokemon pokemon tersebut hanya dimungkinkan ketika sebuah pokemon berada dalam radius 40 meter dari posisi karakter player di peta. Selain menangkap pokemon disekitar, seorang player juga bisa berpartisipasi aktif dalam pertempuran bernama *Raid Battles*, dimana pada pertempuran tersebut seorang player bisa bekerja sama untuk menaklukan 1 pokemon besar dan mendapatkan berbagai jenis *resource.*

Salah satu resource paling penting dalam permainan ini adalah Poke Coins. Poke Coins adalah mata uang utama yang digunakan dalam permainan ini. Mata uang tersebut memiliki peran yang significant terhadap perkembangan pemain dan pokemonnya karena dengan menggunakan mata uang tersebut seorang pemain dapat membeli berbagai item dalam took seperti, Poke Balls untuk menangkap Pokemon, Potion dan Revive untuk memulihkan kesehatan Pokemon setelah pertempuran, Incubators untuk menetaskan telur, serta Raid Passes yang memberikan akses ke pertempuran Raid. Selain itu, Poke Coins memungkinkan seorang pemain untuk meningkatkan kapasitas Tas penampung Resource dan Tas penampung Pokemon. Fitur tersebut merupakan fitur yang sangat berguna bagi pemain yang sering menjelajahi area sekitar mereka untuk mengumpulkan resource dan pokemon. Dengan banyaknya manfaat dari mata uang tersebut dalam perkembangan pemain, menjadikan Poke Coin menjadi salah satu *resource* yang menjadi incaran banyak pemain.

Poke Coin dapat diperoleh dengan menempatkan Pokemon di Gym dan mempertahankan Gym tersebut selama 1 hari penuh. Jika sebuah pokemon berhasil mempertahankan Gym tersebut dalam 1 hari penuh maka ia akan kembali ke player dan memberikan Poke Coin sebanyak 50 Poke Coin. Disisi lain jika ia gagal mempertahankannya selama 1 hari penuh maka ia tidak akan mendapatkan coin secara full ia mungkin hanya mendapat ½ dari jumlah maksimum yaitu 50 Poke Coin per 1

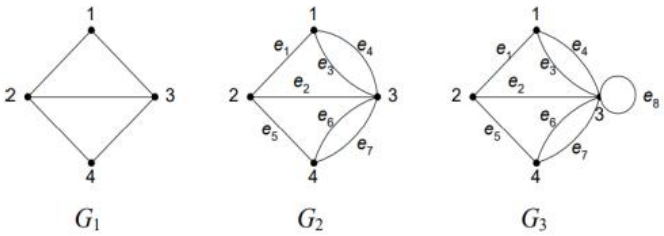
gymnya. Untuk memaksimalkan jumlah Poke Coin yang didapat seorang pemain, ia harus mengatur rute perjalanannya agar dapat mengunjungi seluruh gym disekitarnya dan menaruh pokemonnya disana untuk mendapatkan Poke Coins.

Permasalahan tersebut dapat dimodelkan sebagai Traveling Salesman Problem (TSP), sebuah permasalahan optimasi yang bertujuan menemukan rute terpendek untuk mengunjungi sejumlah lokasi sekaligus. Dalam konteks ini, setiap Gym dapat dianggap sebagai simpul dalam graf, sementara jarak antar Gym menjadi bobot sisi graf tersebut. Dengan menerapkan algoritma TSP, pemain dapat menentukan rute optimal yang memungkinkan mereka mengunjungi beberapa Gym dalam waktu sesingkat mungkin, sehingga meningkatkan peluang untuk mendapatkan Poké Coins. Pendekatan ini tidak hanya menghemat waktu tetapi juga memaksimalkan efisiensi pemain dalam mengelola sumber daya mereka, menjadikan algoritma TSP solusi praktis untuk meningkatkan kenyamanan bermain tanpa membuang waktu secara cuma-cuma.

II. LANDASAN TEORI

A. Teori Graf

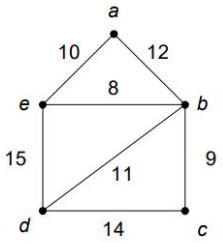
Graf adalah salah satu struktur diskrit yang digunakan untuk melihat hubungan antar simpul dan sisi. Jika diliat dari hubungan antar simpulnya, graf dapat dibagi menjadi beberapa jenis seperti graf sederhana, graf ganda dan semu. Graf sederhana adalah Graf yang tidak mengandung gelang maupun sisi ganda. Sementara graf ganda dan graf semu dapat dikelompokkan menjadi 1 jenis graf yaitu graf tak-sederhana. Dimana graf tak-sederhana adalah Graf yang mengandung sisi ganda atau gelang.

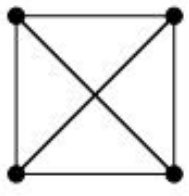


Gambar 2. Jenis-jenis graf
(G1) Graf Sederhana (G2) Graf Ganda (G3) Graf Semu
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Graf masih dapat dibagi lagi berdasarkan orientasi arahsisinya yaitu Graf tak-berarah dan Graf Berarah. Seperti Namanya graf berarah adalah graf yang sisinya memiliki arah. Sementara graf tidak-berarah sisinya tidak mempunyai arah

Selain jenis jenis graf diatas terdapat beberapa lagi jenis graf yang penting untuk diketahui. Jenis graf tersebut adalah graf berbobot dan graf lengkap. Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Sementara graf lengkap adalah graf yang setiap simpulnya saling terhubung satu sama lain tanpa terkecuali



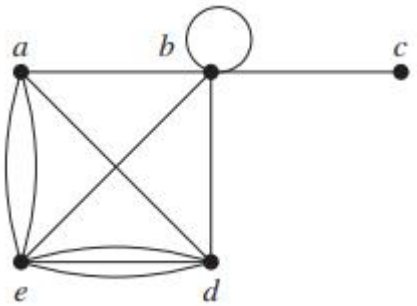


Gambar 3 Graf Berbobot Gambar 4 Graf Lengkap
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Berbagai jenis graf tersebut dapat direpresentasikan dalam notasi G = (V,E). Dimana V melambangkan himpunan tidak kosong dari simpul-simpul {v1,v2,…,vn}, sedangkan E adalah himpunan sisi yang menghubungkan sepasang simpul {e1,e2,…,en}.

B. Terminologi Graf

Dalam pembahasan mengenai graf tentu kita harus mengetahui terminology apa saja yang digunakan. Salah satu terminlogi tersebut adalah *adjacent.* Terminologi tersebut digunakan untuk menggambarkan hubungan antara dua titik atau dua simpul dalam graf tak berarah G. Hubungan antar simpul tersebut tidak terbatas hanya pada 1 simpul lain bisa saja 1 simpul memiliki adjacent yang lebih dari 1. Banyaknya titik-titik atau simpul yang berhubungan dengan suatu simpul asal disebut sebagai derajat. Dalam notasi matematikanya derajat dari sebuah simpul dapat ditulis dengan d(v) dimana v mewakili banyaknya simpul lain yang terhubung dengannya. Selain kedua terminology tersebut masih ada satu terminology lagi yang penting untuk dibahas. Terminologi tersebut adalah *Incidendy*, terminologi tersebut dapat ditulis dalam matematika e = (vj , vk) yang artinya sebuah sisi(e) dikatakan bersisian dengan simpul vj , atau e bersisian dengan simpul vk.



Gambar 5
Sumber:

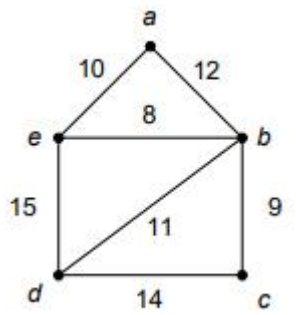
https://elearn.daffodilvarsity.edu.bd/pluginfile.php/783865/mod_resource/intro/Discrete%20Mathematics%20and%20Its%20Applications%2C%207%20edition%20-%20Rosen.pdf

Berdasarkan gambar diatas kita dapat mengetahui bahwa derajat masing masing simpul sebagai berikut d(a) = 4, d(b) = d(e) = 6, d(c) = 1, and d(d) = 5. Selain itu kita juga dapat melihat hubungan tiap simpulnya seperti simpul a bertetangga dengan {b, d, e}, simpul b bertetangga dengan {a, b, c, d, e}, simpul c bertetangga dengan {b}, simpul d bertetangga dengan = {a, b, e}, dan simpul e bertetangga dengan e {a, b, d}.

C. Representasi Graf

Dalam teori Graf terdapat beberapa cara untuk merepresentasikan sebuah graf seperti adjency list, adjency matriks dan incidency list. Pemilihan jenis representasi graf ini bisa disesuaikan dengan kebutuhannya. Misal saja kita memiliki graf sederhana yang bersifat jarang (sparse), maka adjency list biasanya menjadi pilihan yang lebih efisien. Hal itu disebabkan oleh karakteristik adjency list yang hanya menyimpan simpul simpul yang bertetangga sehingga meminimalkan penggunaan memori. Namun jika graf yang dimiliki padat yaitu tiap simpul di graf memiliki derajat mendekati jumlah simpul yang ada maka adjency matrix akan digunakan. Pemilihan adjency matrix sebagai representasi dari graf didasarkan karena adjency matriks memiliki efesian yang lebih tinggi untuk mengecek tiap sisi {vi,vj}.

Selain itu, adjacency matrix juga sangat berguna untuk memodelkan weighted graph, di mana setiap elemen matriks dapat merepresentasikan bobot dari sisi yang menghubungkan dua simpul. Hal ini memberikan visualisasi yang lebih jelas tentang hubungan antar simpul sekaligus memungkinkan kita untuk langsung melihat bobot hubungan tersebut. Dengan demikian, adjacency matrix tidak hanya efisien untuk graf padat, tetapi juga sangat efektif dalam menganalisis graf berbobot.



	a	b	c	d	e
a	∞	12	∞	∞	10
b	12	∞	9	11	8
c	∞	9	∞	14	∞
d	∞	11	14	∞	15
e	10	8	∞	15	∞

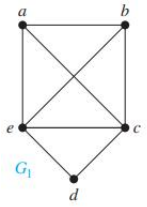
Gambar 6 Representasi Graf Berbobot dengan Adjency Matrix

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

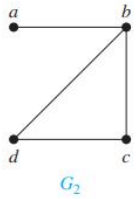
D. Lintasan dan Sirkuit Hamilton

Dalam sebuah graf jika terdapat sebuah lintasan yang bisa melewati seluruh simpul tepat sekali pada maka lintasan tersebut disebut lintasan Hamilton. Sementara itu, jika ada sebuah lintasan yang mampu melewati seluruh simpul tepat satu kali dan kembali ke titik awal maka hal tersebut disebut sebagai sirkuit Hamilton. Sebuah graf yang memiliki sirkuit Hamilton disebut sebagai graf Hamilton. Namun jika hanya mempunyai lintasan Hamilton maka disebut sebagai graf semi Hamilton

Untuk menentukan apakah sebuah graf merupakan graf Hamilton, semi-hamilton atau bukan keduanya. Kita dapat memanfaatkan salah satu teorema yang ada teorema yaitu teorema Dirac. Teorema dirac menyatakan bahwa jika G adalah graf sederhana dengan simpul n ≥3 dan setiap derajat tiap simpulnya ≥ n/2 maka ia adalah graf Hamilton.



Gambar 7 Graf Hamilton



Gambar 8 Graf Semi Hamilton

Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Jika kita melihat dari Gambar 7 kita bisa mengetahui bahwa gambar tersebut merupakan graf Hamilton karena ia mempunyai sirkuit yang melewati tiap simpul dan kembali ke titik awal. Meskipun secara visual kita mengetahui bahwa graf tersebut adalah graf Hamilton, namun pembuktian menggunakan teorema Dirac bisa menghasilkan hasil yang akan bertentangan dengan pengamatan kita. Hal ini disebabkan Teorema Dirac merupakan syarat cukup bukan syarat perlu untuk suatu graf dikatakan sebagai graf Hamilton.

E. Travel Salesman Problem

Traveling Salesman Problem atau TSP adalah salah satu masalah optimasi yang menggunakan konsep dari Sirkuit Hamilton. TSP ini bertujuan untuk menentukan rute dengan harga (bobot) paling rendah untuk mengunjungi seluruh kota yang ada masing-masing satu kali dan kembali ke kota asal. Masalah ini biasanya dimodelkan dalam bentuk graf berbobot

dimana bobotnya dapat berupa harga perjalanan atau jarak dari simpul satu ke simpul lainnya.

Sebenarnya tidak ada cara atau algoritma eksak untuk menyelsaikan tersebut masalah secara efisen untuk semua kasus karena TSP termasuk dalam kategori NP-Hard. Meskipun demikian beberapa cara telah dikembangkan untuk mendapatkan hasil optimasi yang cukup baik. Beberapa diantaranya adalah algoritma heuristik seperti Nearest Neighbor atau 2-opt dan algoritma metaheuristik seperti Simulated Annealing dan Genetic Algorithm

F. ALGORITMA TSP

Untuk menyelesaikan TSP terdapat banyak algortima yang bisa dimanfaatkan dua diantaranya adalah algoritma Genetic Algorithm dan Table Search.

Genetic Algorithm adalah salah satu algoritma untuk menyelesaikan TSP dengan memanfaatkan konsep evolusi biologi. Dalam penerapannya algoritma ini memiliki beberapa Langkah utama yaitu

1. Insialisasi Kromosom
   Pada tahap ini, populasi akan dihasilkan secara acak dan setiap kromosom yang ada dalam populasi akan merepresentasikan satu solusi potensial untuk TSP kita,
2. Evaluasi Kromosom
   Pada tahap ini algoritma ini akan menghitung nilai fitness kromosom yang nantinya nilai tersebut digunakan untuk pemiliham kromosom terbaik.
3. Seleksi Kromosom
   Nilai fitness dari tahap evaluasi akan dimanfaatkan untuk mensseleksi solusi yang ada. Beberapa teknik seleksi yang sering digunakan adalah *Roulette Wheel Selection, Elitism, Rank Base Selection, dan Steady State Selection.*
4. Crossover
   Dua kromosom induk yang terpilih dari proses seleksi akan digunakan untuk menghasilkan solusi baru.
5. Mutasi
   Untuk menjaga keragaman populasi, kromosom dapat dimutasi dengan mengubah sebagian kecil elemen misalnya, menukar dua kota dalam urutan perjalanan.
6. Regenerasi
   Kromosom dari hasil proses 3-4 akan digunakan kembali untuk mencari solusi lain Hal ini akan terus berulang hingga kriteria tertentu dicapai.

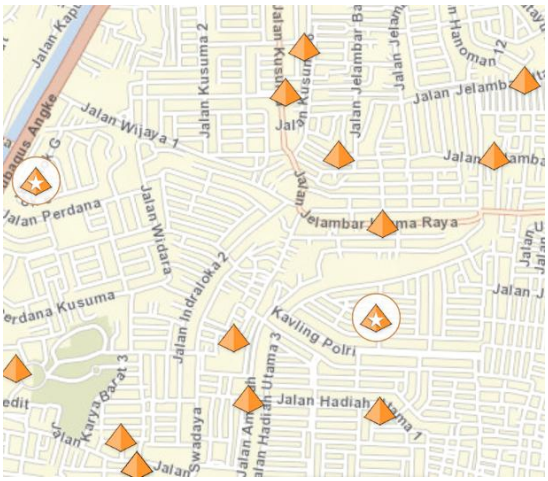
Tabu Search adalah salah satu algoritma metaheuristic yang dapat digunakan untuk menyelesaikan masalah optimasi rute(TSP). Algoritma ini memiliki kelebihan berupa ia akan menghindari Solusi yang sudah pernah dicoba sehingga dapat keluar dari jebakan local optima. Langkah Langkah implementasinya adalah sebagai berikut:

1. Inisialisasi Solusi Awal bisa dipilih secara acak
2. Mencari solusi dengan cara menukar dua kota dalam rute
3. Setelah itu solusi dengan jarak terpendek akan dipilih akan tetapi solusi tersebut tidak boleh ada dalam Tabu List.
4. Perbarui tabu List berdasarkan solusi yand didapat pada loangkah 3
5. Ulangi Langkah hingga mencapai kriteria tertentu

III. METODE

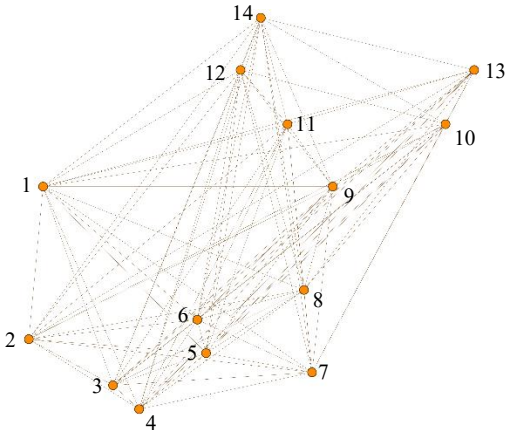
A. Pemodelan Graf dan Representasinya

Untuk melakukan percobaan dan mencari solusi paling optimal untuk masalah optimasi rute gym di Pokemon Go, langkah awal yang penting adalah harus mengetahui lokasi lokasi gym disekitar kita. Dibawah ini adalah gambar lokasi tiap gym yang akan digunakan.



Gambar 9 Lokasi Gym
Source:https://www.pogomap.info/location/-6,147389/106,781545/15

Pada gambar 9, setiap symbol oranye merepresentasikan lokasi gym dan tiap symbol oranye tersebut akan menjadi simpul dalam graf kita. Dengan memanfaatkan tools online seperti yED live kita dapat melihat bagaimana hubungan tiap simpul gym tersebut. Berikut representasi grafnya:



Gambar 10. Representasi Graf Lengkap lokasi gym

Untuk mencari rute terbaik yang bisa dilewati oleh pemain, kita harus membuat graf berbobot, dimana bobot ini akan merepresentasikan jarak antar kedua gym. Jarak antara ke dua gym ini akan dihitung berdasarkan data koordinat tiap simpul

(gym) dengan menggunakan rumus jarak antar dua titik dan rumus Haversine distance. Dibawah ini merupakan rumus Haversine Distance:

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

r : Jari-jari bumi (6371 km untuk kilometer).
$\phi 1, \phi 2$: Lintang dari dua titik.
$\lambda 1, \lambda 2$: Bujur dari dua titik.

Data-data koordinat dari tiap simpul tidak akan ditampilkan karena jumlah simpul yang cukup banyak sebagai gantinya, hasil perhitungan bobot akan ditunjukkan dalam bentuk adjency matriks. Proses perhitungan bobot dan representasi graf tersebut akan dieksekusi menggunakan bahasa pemograman python. Di bawah ini merupakan representasi matriks adjency dari gambar 9:

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14
e1	0.0	0.67	0.97	1.08	1.09	0.91	1.49	1.33	1.25	1.64	1.1	0.96	1.79	1.07
e2	0.67	0.0	0.44	0.56	0.83	0.77	1.3	1.31	1.39	1.85	1.38	1.38	2.08	1.53
e3	0.97	0.44	0.0	0.12	0.47	0.52	0.93	1.03	1.2	1.66	1.28	1.37	1.92	1.53
e4	1.08	0.56	0.12	0.0	0.44	0.54	0.86	1.01	1.2	1.65	1.31	1.43	1.93	1.59
e5	1.09	0.83	0.47	0.44	0.0	0.22	0.47	0.57	0.78	1.22	0.94	1.11	1.51	1.27
e6	0.91	0.77	0.52	0.54	0.22	0.0	0.59	0.54	0.68	1.13	0.77	0.91	1.4	1.08
e7	1.49	1.3	0.93	0.86	0.47	0.59	0.0	0.35	0.67	0.99	0.93	1.19	1.29	1.33
e8	1.33	1.31	1.03	1.01	0.57	0.54	0.35	0.0	0.32	0.68	0.59	0.87	0.98	0.99
e9	1.25	1.39	1.2	1.2	0.78	0.68	0.67	0.32	0.0	0.46	0.29	0.59	0.72	0.69
e10	1.64	1.85	1.66	1.65	1.22	1.13	0.99	0.68	0.46	0.0	0.54	0.77	0.3	0.78
e11	1.1	1.38	1.28	1.31	0.94	0.77	0.93	0.59	0.29	0.54	0.0	0.3	0.7	0.4
e12	0.96	1.38	1.37	1.43	1.11	0.91	1.19	0.87	0.59	0.77	0.3	0.0	0.85	0.16
e13	1.79	2.08	1.92	1.93	1.51	1.4	1.29	0.98	0.72	0.3	0.7	0.85	0.0	0.8
e14	1.07	1.53	1.53	1.59	1.27	1.08	1.33	0.99	0.69	0.78	0.4	0.16	0.8	0.0

Gambar 11. Adjency Matriks
Source: Dokumen Pribadi

B. Implementasi menggunakan GA dan Tabu Search

Pada penerapannya, dua algoritma Traveling Salesman Problem (TSP), yaitu Genetic Algorithm (GA) dan Tabu Search (TS), digunakan untuk mencari rute optimal antar gym dalam permainan Pokemon Go. Kedua algoritma ini diimplementasikan dalam python dan menerima adjacency matrix sebagai input utama.

Genetic Algrotithm (GA) disini digunakan untuk mengeksplorasi ruang solusi dengan menggunakan konsep evolusi dan perkawinan silang antar parent. Secara garis besar algoritma utamanya masih sama dengan yang dijelaskan pada sesi dasar teori. Berikut implementasinya untuk algoritma GA:

Dalam penerapan GA, untuk inisialisasinya kita memilih solusi yang mungkin secara random. Setelah itu, solusi tersebut akan di evaluasi menggunakan fungsi fitness. Fungsi fitness sendiri untuk melihat seberapa optimal solusi/kromosom tersebut. Berikut implementasi evaluasi kromosom:

```python
def fitness(chromosome, distance_matrix):
    distance = sum(
        distance_matrix[chromosome[i], chromosome[i + 1]] for i in range(len(chromosome) - 1)
    )
    distance += distance_matrix[chromosome[-1], chromosome[0]]
    return 1 / distance
```

Gambar Implementasi Fitness
Source: Dokumen Pribadi

Selanjutnya populasi dari solusi awal yang sudah dicari nilai fitnessnya akan melalui 2 proses utama yaitu pemilihan, dan evolusi. Proses evolusi terdiri dari dua tahap berbasis probabilitas, yaitu perkawinan silang (crossover) dan mutasi. . Dalam percobaan kali ini proses pemilihan dan evolusi menggunakan metode probabilitas. Hal ini diperlukan agar algoritma GA mempunyai kemungkinan untuk memperluas

```python
def select_parents(population, fitness_scores):
    probabilities = [score / sum(fitness_scores) for score in fitness_scores]
    parents = random.choices(population, probabilities, k=2)
    return parents

def ordered_crossover(parent1, parent2):
    start, end = sorted(random.sample(range(len(parent1)), 2))
    child = [-1] * len(parent1)
    child[start:end] = parent1[start:end]

    pos = end
    for city in parent2:
        if city not in child:
            if pos == len(parent2):
                pos = 0
            chiid[pos] = city
            pos += 1

    return child

def swap_mutation(chromosome):
    idx1, idx2 = random.sample(range(len(chromosome)), 2)
    chromosome[idx1], chromosome[idx2] = chromosome[idx2], chromosome[idx1]
```

Gambar Implementasi

Kedua proses pemilihan orang tua dan evolusi akan dimanfaatkan oleh fungsi utama yaitu GA, dalam nama fungsi evolve_population, untuk melakukan eksplorasi dan eksploitasi terhadap solusi yang ada

```python
def genetic_algorithm(coordinates):
    num_nodes = len(coordinates)
    adj_matrix = np.zeros((num_nodes, num_nodes))

    for i in range(num_nodes):
        for j in range(num_nodes):
            if i != j:
                adj_matrix[i, j] = haversine_distance(coordinates[i], coordinates[j])
    population = initialize_population(POP_SIZE, num_nodes)
    best_solution = None
    best_fitness = -1

    for generation in range(MAX_GEN):
        population, fitness_scores = evolve_population(population, adj_matrix)

        current_best = max(fitness_scores)
        current_best_index = fitness_scores.index(current_best)

        if current_best > best_fitness:
            best_fitness = current_best
            best_solution = population[current_best_index]

        print(f"Generation {generation}: Best Fitness = {best_fitness:.6f}")

    return best_solution, 1 / best_fitness
```

Gambar Implementasi
Source: Dokumen Pribadi

Solusi yang dihasilkan dari implementasi GA diatas akan dimanfaatkan oleh Tabu Search untuk mengoptimalkan solusinya dengan melakukan pencarian local lebih mendalam. Berikut implementasinya dalam python:

```python
def tabu_search(best_route, distance_matrix):
    current_solution = best_route[:]
    best_solution = best_route[:]
    best_distance = 1 / fitness(best_solution, distance_matrix)

    tabu_list = []
    for iteration in range(MAX_TABU_ITER):
        neighbors = []
        for i in range(len(current_solution)):
            for j in range(i + 1, len(current_solution)):
                neighbor = current_solution[:]
                neighbor[i], neighbor[j] = neighbor[j], neighbor[i]  # Swap cities
                neighbors.append((neighbor, i, j))

        neighbors = [(neighbor, i, j) for neighbor, i, j in neighbors if (i, j) not in tabu_list]
        neighbors = sorted(neighbors, key=lambda x: 1 / fitness(x[0], distance_matrix))

        if not neighbors:
            break

        best_neighbor, i, j = neighbors[0]
        tabu_list.append((i, j))
        if len(tabu_list) > TABU_TENURE:
            tabu_list.pop(0)
```

```
        current_solution = best_neighbor
        current_distance = 1 / fitness(current_solution, distance_matrix)

        if current_distance < best_distance:
            best_solution = current_solution
            best_distance = current_distance

    return best_solution, best_distance
```

Gambar Implementasi
Source: Dokumen Pribadi

Setelah seluruh algoritma yang dibutuhkan telah diimplementasi kami akan melakukan beberapa percobaan pada algoritma GA untuk melihat seberapa baik performanya dalam mencari solusi paling optimalnya dengan beberapa variasi pada probabilitas evolusinya. Selain itu, kami juga merancang percobaan untuk menganalisis dampak penggunaan Tabu Search (TS) dalam meningkatkan solusi yang dihasilkan oleh GA. Berikut table yang menunjukkan variasi parameter untuk GA:

Tabel Variasi

POP_SIZE	CO_PROB	MUTATE_PROB	GEN
50	0.6	0.1	50
100	0.7	0.2	60
200	0.8	0.3	75

IV. HASIL PERCOBAAN DAN DISKUSI

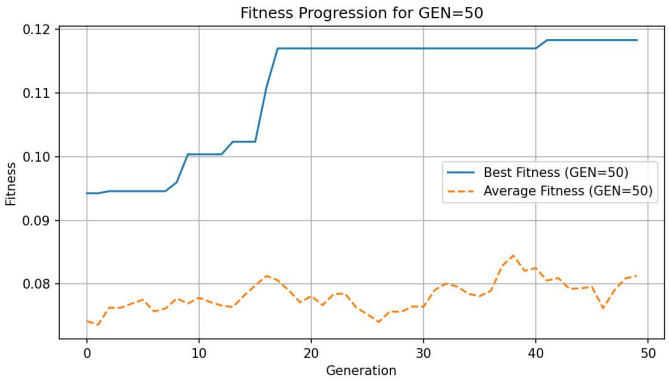
POP_SIZE	CO_PROB	MUTATE_PROB	GEN	Best Distance (km)
50.0	0.6	0.2	50.0	8.45
100.0	0.7	0.25	60.0	7.62
200.0	0.8	0.3	75.0	7.98

POP_SIZE	CO_PROB	MUTATE_PROB	GEN	Best Route
50	0.6	0.2	50	[12, 0, 10, 11, 1, 9, 4, 6, 7, 8, 3, 2, 5, 13, 12]
100	0.7	0.25	60	[1, 7, 0, 2, 6, 5, 12, 4, 9, 8, 13, 11, 10, 3, 1]
200	0.8	0.3	75	[12, 5, 2, 11, 7, 8, 9, 0, 6, 1, 10, 13, 4, 3, 12]

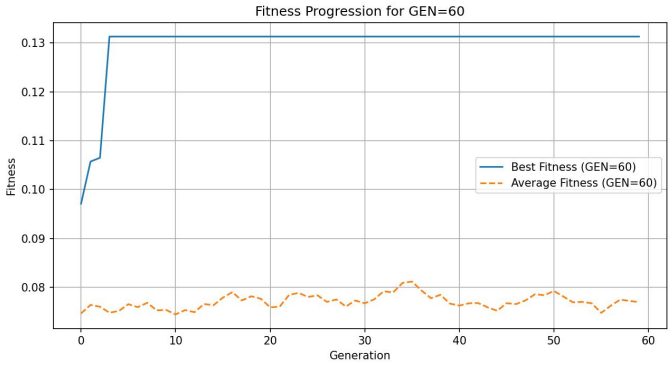
POP_SIZE	CO_PROB	MUTATE_PROB	GEN	Execution Time (s)
50.0	0.6	0.2	50.0	0.34
100.0	0.7	0.25	60.0	2.03
200.0	0.8	0.3	75.0	20.77

Gambar Tabel Hasil Percobaan GA dengan berbagai variasi
Source: Dokumen Pribadi

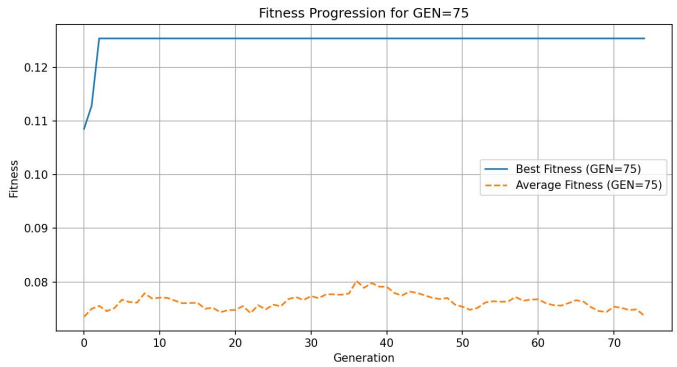
Berdasarkan tabel di atas, dapat disimpulkan bahwa perubahan parameter seperti penambahan populasi (POP_SIZE), probabilitas crossover (CO_PROB), probabilitas mutasi (MUTATE_PROB), dan jumlah generasi (GEN) tidak secara signifikan memengaruhi hasil akhir. Dalam percobaan ini, terlihat bahwa semakin besar ukuran populasi dan jumlah generasi, semakin tinggi juga waktu yang diperlukan untuk mendapatkan solusi. Selain itu, probabilitas yang lebih tinggi, ukuran populasi yang lebih besar, dan jumlah generasi yang lebih banyak tidak selalu menjamin hasil yang lebih baik dibandingkan dengan parameter yang lebih rendah. Hal ini dapat dapat terjadi karena beberapa faktor seperti terjebaknya algoritma GA dalam local optimum.



Grafik Fitness 1
Source: Dokumen Pribadi



Grafik Fitness 2
Source: Dokumen Pribadi



Grafik Fitness 3
Source: Dokumen Pribadi

Jika kita mengamati dengan saksama ketiga grafik di atas, terlihat bahwa nilai *best fitness* hanya mengalami peningkatan signifikan pada generasi-generasi awal. Setelah itu, peningkatan tersebut melambat atau bahkan stagnan, yang mengindikasikan kemungkinan bahwa algoritma Genetic Algorithm (GA) terjebak pada solusi optimum lokal pada tahap awal.

Fenomena ini dapat disebabkan oleh solusi optimum awal yang memiliki probabilitas tinggi untuk dipilih dalam proses seleksi. Akibatnya, hasil dari proses *crossover* (kawin silang) sering kali menghasilkan individu baru (*offspring*) yang tidak jauh berbeda dari solusi awal dalam hal nilai fitness. Meskipun demikian, algoritma GA tetap menunjukkan kemampuan untuk mengeksplorasi solusi yang lebih luas, tidak hanya terfokus

pada solusi awal. Hal ini dapat dilihat dari peningkatan nilai fitness pada generasi ke-40 di grafik *Fitness 1*.

Selain itu, ketiga grafik juga menunjukkan bahwa hasil dari proses *crossover* tidak selalu lebih baik dibandingkan dengan *parent*-nya. Hal ini tercermin dari fluktuasi nilai fitness yang terlihat sejak generasi awal hingga akhir. Meskipun ada fluktuasi, peningkatan nilai fitness pada beberapa generasi tertentu menunjukkan bahwa proses kawin silang dan mutasi mampu memperluas ruang solusi dan pada akhirnya menghasilkan solusi yang lebih optimal.

POP_SIZE	CO_PROB	MUTATE	GEN	DIST
50	0.6	0.2	50	6.18
100	0.7	0.25	60	6.13
200	0.8	0.3	75	6.12

POP_SIZE	CO_PROB	MUTATE_PROB	GEN	Best Route
50	0.6	0.2	50	[3, 5, 4, 6, 7, 8, 10, 9, 12, 13, 11, 0, 1, 2, 3]
100	0.7	0.25	60	[0, 1, 2, 3, 5, 4, 6, 7, 8, 9, 12, 10, 11, 13, 0]
200	0.8	0.3	75	[5, 4, 6, 7, 8, 9, 12, 10, 13, 11, 0, 1, 2, 3, 5]

POP_SIZE	CO_PROB	MUTATE	GEN	TIME
50	0.6	0.2	50	0.24
100	0.7	0.25	60	1.75
200	0.8	0.3	75	19.17

Tabel Hasil Percobaan GA + TS

Setelah mengkombinasikan Genetic Algorithm dan Tabu Search, kita mendapatkan hasil rute yang jauh lebih baik, yaitu meningkat sekitar 19.55% hingga 26.85%. Tabu Search dapat membantu GA dalam eksplorasi karena Tabu Search memiliki mekanisme untuk menghindari *local optima* dengan memasukkan solusi yang sudah pernah dieksplor kedalam *tabu list*.

Dengan cara ini, Tabu Search memungkinkan pencarian solusi yang lebih menyeluruh di ruang solusi. Sementara itu, Genetic Algorithm unggul dalam eksploitasi dengan mekanisme seleksi, kawin silang (*crossover*), dan mutasi untuk menghasilkan individu baru yang lebih baik.

Berdasarkan hasil-hasil percobaan diatas dengan menggunakan algortima hybrid untuk menyelesaikan permasalahan TSP. Maka seorang player pokemon Go akan membutuhkan waktu dan jarak yang lebih sedikit dalam proses mengambil seluruh Poke Coins dari Gym disekitarnya.

V. KESIMPULAN

Berdasarkan hasil percobaan dan analisis yang telah dilakukan, dapat disimpulkan beberapa hal berikut:

1. Performa Genetic Algorithm (GA):
   Algoritma GA memiliki keunggulan untuk mencari solusi-solusi yang mendekati optimal. Namun, algoritma ini memiliki kecendrungan untuk terjebak dalam solusi optimum local. Hal tersebut bisa dilihat dari adanya stagnansi dalam nilai *best fitness* pada grafik.
2. Pengaruh Parameter GA:
   Perubahan parameter seperti ukuran populasi (POP_SIZE), probabilitas crossover (CO_PROB), probabilitas mutasi (MUTATE_PROB), dan jumlah generasi (GEN) tidak selalu memberikan dampak signifikan terhadap hasil akhir.
3. Kombinasi GA dan Tabu Search (TS):
   Implementasi hybrid antara GA dan Tabu Search menunjukkan peningkatan performa yang signifikan dalam konteks optimasi rute Gym di Pokemon GO. Peningkatan tersebut berada dikisaran 19.55% hingga 26.85% dibandingkan penggunaan GA saja.

Secara keseluruhan, pendekatan hybrid GA+TS menawarkan solusi yang lebih baik untuk permasalahan optimisasi rute dibandingkan penggunaan GA secara mandiri. Algoritma GA dapat dimanfaatkan sebagai pondasi atau inisialisasi rute pada Tabu Search untuk mendapatkan hasil yang lebih optimal.

Pendekatan hybrid GA+TS tidak hanya berhasil menyelesaikan permasalahan optimasi rute Gym dengan lebih baik dibandingkan penggunaan GA secara mandiri, tetapi juga memberikan dampak positif langsung terhadap pengalaman pemain Pokémon GO. Pemodelan masalah sebagai Traveling Salesman Problem (TSP) terbukti relevan dan dapat diadaptasi untuk konteks permainan lain dengan mekanisme serupa.

Dengan memanfaatkan kombinasi algoritma ini, permainan Pokémon GO menjadi lebih strategis, efisien, dan menyenangkan bagi para pemain, menjadikan algoritma hybrid sebagai pendekatan yang praktis dan aplikatif dalam dunia *game optimization*.

REFERENCES

[1] R. Munir, “Graf Bagian 1,” IF2120 Matematika Diskrit. [Online]: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. Diakses pada 4 Januari 2024

[2] R. Munir, “Graf Bagian 2,” IF2120 Matematika Diskrit. [Online]: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf Diakses pada 4 Januari 2024

[3] R. Munir, “Graf Bagian 3,” IF2120 Matematika Diskrit. [Online]: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf Diakses pada 4 Januari 2024

[4] Rosen, K. H. (2012). Discrete Mathematics and Its Applications (7th ed., pp. 641-693). McGraw-Hill Education. [Online] https://elearn.daffodilvarsity.edu.bd/pluginfile.php/783865/mod_resource/intro/Discrete%20Mathematics%20and%20Its%20Applications%2C%207%20edition%20-%20Rosen.pdf Diakses pada 6 Januari 2024

[5] GeeksforGeeks, "Genetic Algoritm," [Online]: .https://www.geeksforgeeks.org/genetic-algorithms/C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995. Diakses pada 6 Januari 2024

[6] GeeksforGeeks, " What is TABU Search?," [Online]: https://www.geeksforgeeks.org/what-is-tabu-search/ Diakses 6 Januari 2024

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Januari 2024

Ferdinand Gabe Tua Sinaga dan 13523051