TUGAS BESAR
STRATEGI ALGORITMA DENGAN PENDEKATAN *TRAVELING SALESMAN PROBLEM BRUTE FORCE* PADA DESTINASI WISATA BALI

Oleh:
Muhammad Daffa Regenta S
Dewa Made Aditya
Abigael Mark

NIM: 1301184291, 1301180266, 1301180134

Email: daffaregenta@student.telkomuniversity.ac.id
@student.telkomuniversity.ac.id
markstevan@student.telkomuniversity.ac.id

ABSTRAK

Pariwisata merupakan sektor potensial untuk dikembangkan sebagai salah satu sumber pendapatan daerah. Pulau Bali merupakan salah satu destinasi wisata yang banyak digunakan warga negara asing. Disamping itu masyarakatnya dikenal memiliki karakter yang terbuka, budaya adat istiadat yang unik, kesenian yang khas, hingga pada cita rasa kuliner daerahnya yang khas. Realita ini secara tidak langsung ikut berkontribusi pada terbentuknya image baru bagi Pulau Bali yang mulai dikenal sebagai sebuah pulau multinations. Kota ini memiliki berbagai tempat wisata, karena banyaknya pilihan tempat wisata, dapat menyebabkan wisatawan sulit untuk membagi waktu berwisata yang sesuai dengan harapan.
Permasalahan yang sering dialami oleh wisatawan adalah ketika ingin mengunjungi suatu tempat wisata, namun tidak sempat berkunjung ke tempat wisata yang lainnya. Oleh sebab itu, wisatawan perlu memiliki gambaran rute perjalanan wisata agar terbentuk suatu jadwal wisata secara efektif. Pencarian rute wisata terbaik dengan waktu dan biaya seminimal mungkin, namun dapat berkunjung ke berbagai tempat wisata secara optimal. Pencarian rute wisata tersebut dapat dikenal dengan sebutan Travelling Salesman Problem. Pencarian Optimasi dari pencarian wisata dengan pertimbangan biaya dan waktu dapat menggunakan Knapsack Problem. Untuk menghitung kedua metode tersebut digunakan Metode Brute Force untuk memaksimalkan nilai optimal untuk penyusunan jadwal wisata otomatis dengan mempertimbangkan waktu dan biaya di Bali.

1. PENDAHULUAN

Perkembangan Penelitian dalam permasalah transportasi masih terus dilakukan untuk mencari algoritma baru untuk menyelesaikan sebuah permasalahan yang menghasilkan solusi yang lebih optimal dari penelitian sebelumnya. Salah satu permasalahan transportasi yang paling sering dibahas adalah Traveling salesman problem (TSP). TSP merupakan

permasalahan dimana seorang salesman harus mengunjungi setiap kota, salesman tersebut berangkat dari kota awal dan berakhir pada kota yang sama, serta setiap kota harus dikunjungi tidak lebih dari satu kali. Fungsi tujuan dari TSP adalah untuk mencari rute terpendek yang bisa dilalui oleh salesman sehingga dihasilkan total jarak terpendek.

Masalah traveling salesman problem masih sering kali ditemukan dalam kehidupan sehari-hari. Masalah nyata yang sering terjadi misalnya pada logistik perusahaan, dimana sebuah perusahaan harus dapat menentukan berapa banyak jumlah yang dibutuhkan untuk mengantarkan suatu barang atau jasa dari supplier kepada pelanggan sehingga biaya yang dibutuhkan seminimum mungkin. Selain itu ada juga masalah transportasi untuk menentukan jalur angkutan umum yang efisien dan pemasangan jaringan telekomunikasi. Manfaat menyelesaikan kasus TSP juga akan menghasilkan prospek yang bagus kedepannya, karena penyelesaian kasus TSP ini juga dapat digunakan untuk navigasi seperti GPS yang sangat populer saat ini.

Tugas besar ini membahas penerapan metode Algoritma TSP Brute Force pada destinasi wisata Bali dalam pemrograman

2. Dasar Teori

2.1 Traveling Salesman Problem

Traveling salesman problem merupakan salah satu masalah yang paling populer dalam ilmu komputer. Dimana traveling salesman problem merupakan suatu masalah dalam dunia optimasi, dimana ada sebuah kota awal dan sejumlah n kota untuk dikunjungi dan seorang salesman dituntut untuk memulai perjalanan dari kota awal ke seluruh kota dengan mengunjungi setiap kota tepat satu kali. Tujuan dari permasalahan ini adalah untuk menemukan rute yang paling singkat. Permasalahan dapat dideskripsikan dengan menyatakan sebuah kota menjadi sebuah simpul graf, dan jalan yang menghubungkan kota sebagai sisi. Bobot pada sisi menyatakan jumlah antara dua buah kota.

Masalah ini akan sulit diselesaikan apabila terdapat jumlah kota yang banyak. Tantangan dalam penyelesaian masalah ini bukan hanya tentang rute yang baik, tetapi harus menjamin bahwa hasil yang didapat merupakan rute terpendek Penyelesaian masalah TSP mengharuskan untuk melakukan perhitungan terhadap semua rute yang diperoleh, baru kemudian memilih salah satu rute terpendek. Untuk itu jika terdapat n kota yang harus dikunjungi, maka diperlukan proses pencarian sebanyak n/2 rute dengan cara ini waktu komputasi yang diperlukan akan jauh meningkat seiring dengan bertambahnya jumlah kota yang harus dikunjungi.

2.2 Metode Brute Force

Metode Brute force merupakan pendekatan yang langsung dalam memecahkan suatu masalah, yang didasarkan pada pernyataan masalah serta mendefinisikan terhadap konsep yang dilibatkan secara langsung. Pemecahan masalah menggunakan Metode Brute force sangat sederhana, langsung dan jelas. Karakteristik pada Metode Brute force :

1. Jumlah langkah yang diperlukan besar.

2. Dipakai sebagai dasar dalam menemukan suatu solusi yang lebih efisien atau kreatif.
3. Hampir semua masalah dapat diselesaikan dengan metode ini.
4. Digunakan sebagai dasar dalam perbandingan kualitas suatu algoritma.

2.3 Algoritma Brute Force Traveling Salesman Problem (Exhaustive Search)

Exhaustive search adalah teknik pencarian solusi secara solusi brute force untuk masalah yang melibatkan pencarian elemen dengan sifat khusus. Biasanya di antara objek-objek kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari sebuah himpunan. Penggunaan Brute force pada permasalahan TSP dapat diilustrasikan sebagai berikut:

Sebuah biro wisata di Bali ingin meningkatkan kualitas layanannya kepada para turis dengan memberikan rekomendasi rute perjalanan mengunjungi berbagai destinasi wisata di Bali. Sehingga dalam 1 hari, seorang turis dapat mengunjungi semua destinasi wisata yang diinginkan lalu kembali lagi ke hotel tempat turis tersebut menginap.

Langkah-langkah metode exhaustive search:

1. Buat list (Enumerasi) setiap solusi yang mungkin dengan cara yang sistematis.
2. Evaluasi setiap kemungkinan solusi satu per satu, simpan solusi terbaik yang ditemukan sampai sejauh mungkin.
3. bila pencarian berakhir, umumkan solusi terbaik.

L1 = (A,B,C,D,A) = 10 + 12 + 8 + 15 = 45
L2 = (A,B,D,C,A) = 12 + 5 + 9 + 15 = 41
L3 = (A,C,B,D,A) = 10 + 5 + 9+ 8 = 32
L4 = (A,C,D,B,A) = 12 + 5 + 9+ 15 = 41
L5 = (A,D,B,C,A) = 10 + 5 + 9+ 8 = 32
L6 = (A,D,C,B,A) = 10 + 12 + 8+ 15 = 45

Jika contoh diselesaikan dengan metode exhaustive search, maka kita harus mengenumerasi sebanyak (n – 1)! buah sirkuit Hamilton, menghitung setiap bobotnya, dan memilih sirkuit Hamilton dengan bobot terkecil. Kompleksitas waktu nya adalah O(n!).

Dari algoritma exhaustive search di atas dapat dilakukan perbaikan mengenai banyaknya pencarian rute, yakni dengan diketahui bahwa setengah dari rute perjalanan adalah hasil pencerminan dari setengah rute yang lain, yaitu dengan mengubah arah rute perjalanan, rute tersebut adalah :

1. L1 dan L6
2. L2 dan L4
3. L3 dan L5

3. Eksperimen

3.1 Persiapan Data

```
34
35 wisata = [[23,70],[33,29],
36           [13,36],[65,21],
37           [64,56],[53,93],
38           [21,5],[63,93],
39           [41,79],[59,94],
40           [75,41],[75,52],
41           [61,67],[47,60],
42           [32,60]]
43 n = int(input("Masukan jumlah wisata (1 - 14): "))
44
```

Disini dipersiapkan berupa 15 destinasi yang dituju oleh wisatawan berupa koordinat.

```python
#rumus pyhtagoras untuk menghitung jarak antar dua kota
#menggunakan titik koordinat
def jarak(x1, y1, x2, y2) :
  return math.sqrt((x1-x2)**2+(y1-y2)**2)

#fungsi menghhitung jarak rute perjalanan
def totalJarak(tour) :
  d=0
  #Hitung jarak anatar kota dari kota-1 sampai kota-n
  for i in range(1,len(tour)):
        x1 = wisata[tour[i-1]][0]
        y1 = wisata[tour[i-1]][1]
        x2 = wisata[tour[i]][0]
        y2 = wisata[tour[i]][1]
        d += jarak(x1,y1,x2,y2)
  #hitung jarak antara kota pertama dan kota terakhir
  x1 = wisata[tour[len(tour)-1]][0]
  y1 = wisata[tour[len(tour)-1]][1]
  x2 = wisata[tour[0]][0]
  y2 = wisata[tour[0]][1]
  d += jarak(x1,y1,x2,y2)
  return d
```

Berikut adalah fungsi untuk menghitung jarak antar destinasi pertama dan kedua sampai ke-n.

3.2 Pengujian

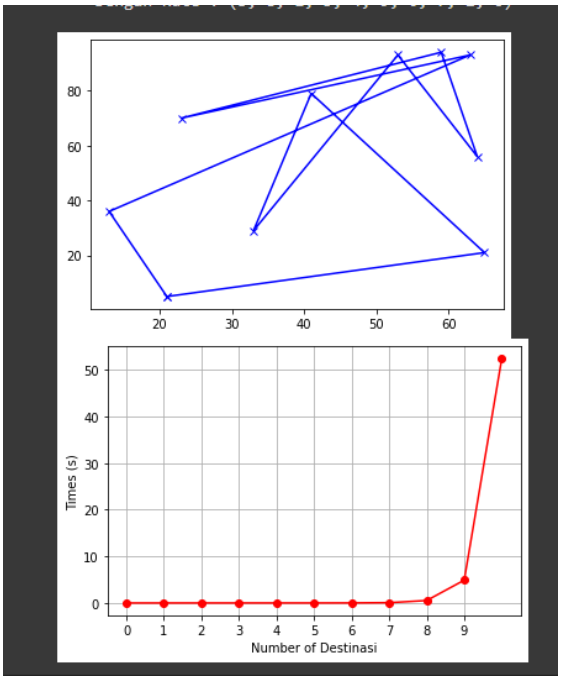
```python
for k in range(1,n+1):
    x.append(k)

    #Generate SATU Rute perjalanan
    tour = random.sample(range(k),k)

    #Membuat SEMUA Rute Perjalanan yang memungkinkan
    generateTour = list(itertools.permutations(tour))

    jarakTerpendek = 999999
    ruteTerpendek = tour
    for i in range(len(generateTour)):
        j = totalJarak(generateTour[i])
        if j<jarakTerpendek:
            jarakTerpendek = int(j)
            ruteTerpendek = i
    y.append(time.perf_counter()-t0)

    print("Jarak terpendek",k,"destinasi wisata yaitu",jarakTerpendek,"KM")
    print("    Dengan Rute :",generateTour[i])
```

di fungsi ini dicoba untuk meng evaluasi hasil yang didapat dan mencari solusi rute terpendek.

3.3 Hasil

```
Masukan jumlah wisata (1 - 14): 10
Jarak terpendek 1 destinasi wisata yaitu 0 KM
    Dengan Rute : (0,)
Jarak terpendek 2 destinasi wisata yaitu 84 KM
    Dengan Rute : (0, 1)
Jarak terpendek 3 destinasi wisata yaitu 98 KM
    Dengan Rute : (1, 0, 2)
Jarak terpendek 4 destinasi wisata yaitu 154 KM
    Dengan Rute : (2, 0, 1, 3)
Jarak terpendek 5 destinasi wisata yaitu 167 KM
    Dengan Rute : (0, 2, 1, 4, 3)
Jarak terpendek 6 destinasi wisata yaitu 201 KM
    Dengan Rute : (4, 0, 2, 3, 1, 5)
Jarak terpendek 7 destinasi wisata yaitu 238 KM
    Dengan Rute : (3, 0, 6, 2, 1, 5, 4)
Jarak terpendek 8 destinasi wisata yaitu 247 KM
    Dengan Rute : (1, 0, 6, 4, 5, 7, 2, 3)
Jarak terpendek 9 destinasi wisata yaitu 247 KM
    Dengan Rute : (5, 1, 0, 7, 4, 3, 6, 2, 8)
Jarak terpendek 10 destinasi wisata yaitu 248 KM
    Dengan Rute : (3, 8, 1, 5, 4, 9, 0, 7, 2, 6)
```

Dapat dilihat bahwa setiap jumlah destinasi langsung mendapatkan solusi terbaik. dengan kompleksitas waktu yang berbeda-beda semakin banyak jumlah destinasi semakin lama waktu untuk meng eksekusi. disini hanya menguji dengan 10 destinasi dikarenakan metode brute force sangat memakan waktu untuk destinasi >10 dan sangat mustahil untuk dieksekusi.



4. Kesimpulan

Kesimpulan yang kami dapat dari percobaan ini adalah bahwa penggunaan algoritma Brute Force ini kurang efisien karena harus mencoba satu per satu setiap kemungkinan yang didapat dan mencari solusi yang terpendek. Dibandingkan dengan metode greedy, metode greedy lebih efisien dikarenakan walaupun tidak menjamin mendapatkan solusi paling optimal, namun metode greedy menjanjikan mendapatkan solusi. Apalagi jika dibenturkan dengan node yang banyak metode greedy adalah yang paling cepat memecahkan masalah TSP.

5. Referensi

[1] (PDF) Penyelesaian Travelling Salesman Problem dengan Algoritma Greedy (researchgate.net)

[2] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj6nauclJXxAhXEQ30KHfx0CE8QFjADegQICRAE&url=http%3A%2F%2Fjurnalti.polinema.ac.id%2Findex.php%2FSIAP%2Farticle%2Fdownload%2F853%2F302%2F&usg=AOvVaw3cIAUFg9TUqZnfVIQxnb17

[3] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwj6nauclJXxAhXEQ30KHfx0CE8QFjABegQIAxAE&url=https%3A%2F%2Frepository.telkomuniversity.ac.id%2Fpustaka%2Ffiles%2F100464%2Fjurnal_eproc%2Fanalisis-penyelesaian-traveling-salesman-problem-dengan-metode-brute-force-menggunakan-graphic-processing-unit.pdf&usg=AOvVaw22QkXwBe7wzdcJzfb-_TYZ

[4] TSP/BruteForce.py at master · X4D0/TSP (github.com)

[5] travelling-salesman/main.py at master · KevHg/travelling-salesman (github.com)

[6] Travelling-Salesman_Problem/router.py at master · rohankumar10/Travelling-Salesman_Problem (github.com)

[7] CS512-TSP/tsp_brute.py at main · cschaozhao/CS512-TSP (github.com)

[8] https://www.academia.edu/6206947/Penyelesaian_Travelling_Salesman_Problem_dengan_Algoritma_Greedy