Application of Graph Theory to Determine the Most Optimal F1 Calendar

Samuel Gerrard Hamonangan Girsang - 13523064[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*gerrardgrs@gmail.com*, *13523064@std.stei.itb.ac.id*

***Abstract*—Formula 1 atau biasa disingkat F1 merupakan aalah satu ajang olahraga balapan mobil. Kegiatan ini dilaakukan sepanjang tahun degan total 24 balapan dalam satu tahun. Sirkuit balapan yang dipakai berasal dari beberapa negara, sehingga tiap tim Formula 1 harus berpindah-pindah negara sebanyak 24 kali dalam satu tahun. Pada makalah ini, penulis memodelkan teori graf dengan mensimulasikan *Travelling Salesman Problem* atau TSP untuk mencari rute paling optimal yang bisa dipakai oleh Formula 1 untuk mengelilingi 24 negara tersebut dalam satu tahun.**

***Keywords*— Formula 1, *Travelling Salesman Problem (TSP)*, Teori Graf, Rute Optimal.**

I. PENDAHULUAN

Formula 1 adalah ajang olahraga balapan mobil paling prestisius di dunia, yang melibatkan sepuluh tim besar dari berbagai negara. Setiap musimnya, balapan diadakan di sirkuit-sirkuit yang tersebar di berbagai negara, dengan jumlah total sekitar 24 balapan. Pada tahun 2024, negara-negara yang menjadi tuan rumah balapan meliputi Bahrain, Saudi Arabia, Australia, Jepang, China, Amerika Serikat (Miami, Las Vegas, Austin), hingga Abu Dhabi sebagai balapan penutup.

Sebagai olahraga global, Formula 1 membawa tantangan beberapa tantangan logistik. Setiap tim diwajibkan untuk berpindah-pindah negara dengan frekuensi yang sangat tinggi selama satu musim. Untuk memenuhi kebutuhan logistik ini, tim-tim Formula 1 mengandalkan berbagai moda transportasi, terutama pesawat terbang. Namun, rute yang digunakan saat ini dinilai kurang efisien, dengan jarak total perjalanan yang mencapai sekitar 130.000 kilometer sedangkan keliling bumi hanya sekitar 40.000 kilometer. Rute tersebut memakan jarak tempuh lebih dari tiga kali keliling bumi.

Inefisiensi ini memiliki dampak signifikan, baik secara finansial maupun lingkungan. Dari sisi finansial, jarak tempuh yang panjang menyebabkan biaya operasional yang sangat tinggi, mencakup bahan bakar, logistik barang, dan tenaga kerja. Dari sisi lingkungan, tingginya penggunaan pesawat terbang mengakibatkan emisi karbon yang besar, berkontribusi pada perubahan iklim global.

Untuk menjawab tantangan ini, diperlukan pendekatan inovatif dalam menyusun jadwal dan rute balapan Formula 1 yang lebih optimal. Salah satu solusi yang dapat diterapkan adalah memanfaatkan teori graf melalui pemodelan *Travelling Salesman Problem* (TSP). TSP adalah salah satu permasalahan optimasi yang bertujuan menemukan rute paling pendek untuk mengunjungi setiap titik hanya satu kali, sebelum kembali ke titik awal. Dalam konteks Formula 1, TSP dapat digunakan untuk menentukan urutan negara tuan rumah balapan yang meminimalkan total jarak perjalanan.

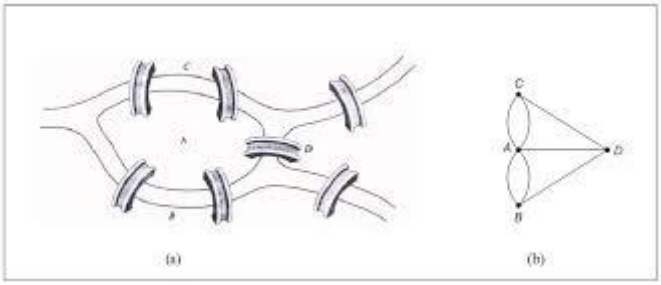
Pemodelan TSP tidak hanya bisa meminimalisir jarak tempuh, tetapi juga emisi karbon yang dihasilkan dari transportasi udara. Solusi ini diharapkan dapat mendukung tujuan Formula 1 dalam mencapai keberlanjutan lingkungan, sekaligus mengurangi beban biaya logistik bagi tim-tim yang berkompetisi.

II. LANDASAN TEORI

1. Graf

Graf merupakan suatu struktur data yang terdiri atas beberapa simpul atau node yang terhubung oleh beberapa garis yang disebut dengan sisi. Graf umumnya digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut.

Graf bermula dari permasalahan jembatan Koenigsberg (tahun 1736) yang mempermasalahkan apakah orang bisa melalui tiap jembatan sekali dan kembali ke posisi semula.

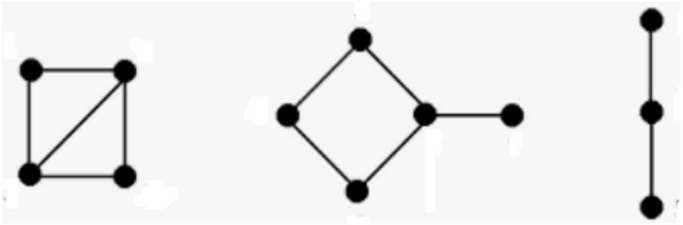


Gambar 2.1.1 Jembatan Koenigsberg
Sumber:
http://digilib.unila.ac.id/14336/6/I.%20PENDAHULUAN.pdf

Graf didefinisikan sebagai G = (V,E) yang dirincikan sebagai, V adalah himpunan tidak kosong dari simpul- simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul.

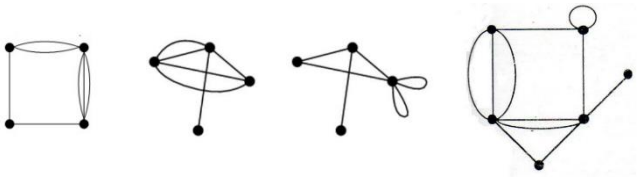
Graf terdiri dari berbagai jenis, yaitu:

- Graf Sederhana (*Simple Graph*)
  Merupakan graf yang tidak mengandung gelang maupun sisi ganda.



Gambar 2.1.2 Graf Sederhana
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

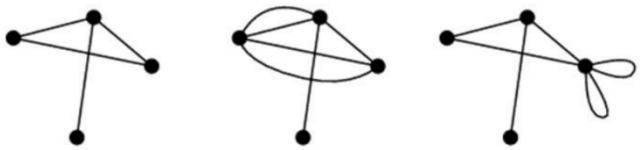
- Graf Tak-Sederhana (*Unsimple Graph*)
  Merupakan graf yang mengandung sisi ganda ataupun gelang.



Gambar 2.1.3 Graf Tak-Sederhana
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

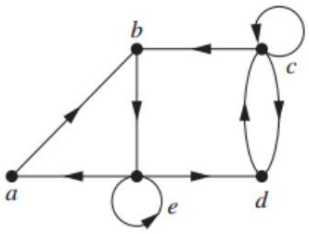
Graf tak-sederhana dibagi menjadi dua lagi yaitu graf ganda (multi-graph) yaitu graf yang menggandung sisi ganda dan graf semu (pseudo-graph) yaitu graf yang mengandung sisi gelang

- Graf Tak-Berarah (*Undirected Graph*)
  Merupakan graf yang tidak memiliki orientasi arah.



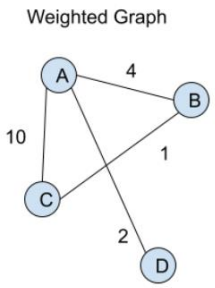
Gambar 2.1.4 Graf Tak-Berarah
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Graf Berarah (*Diagraph*)
  Merupakan graf yang setiap sisinya diberi orientasi arah.



Gambar 2.1.5 Graf Berarah
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

- Graf Berbobot (*Weighted Graf)*
  Graf berbobot adalah graf yang tiap sisinya memiliki nilai.



Gambar 2.1.6 Graf Berbobot
Sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

Ada beberapa terminologi di dalam aplikasi Graf yaitu:

1. Ketetanggaan (*Adjacent*)
   Ketetangaan adalah istilah untuk dua buah simpul yang terhubung langsung.

2. Bersisian (*Incidency*)
   Bersisian adalah suatu istilah untuk suatu sisi e (v_j,v_k) dapat dikatakan bahwa e bersisian dengan simpul v_j atau bersisian dengan simpul v_k.

3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengan simpul lain.

4. Graf Kosong (*Null Graph*)
   Graf yang himpunan sisinya merupakan himpunan kosong.

5. Derajat (*Degree*)
   Derajat adalah jumlah sisi yang terhubung dengan suatu simpul.

6. Lintasan (*Path)*
   Lintasan adalah konsep dasar yang merepresentasikan rangkaian simpul yang terhubung oleh sisi. Biasanya lintasan ini digunakan untuk menunjukkan jalur dari suatu simpul ke simpul lainnya.

7. Sirkuit (*Circuit*)
   Sirkuit merupakan suatu lintasan yang berawal dan berakhir di simpul yang sama.

8. Keterhubungan (*Connected)*
   Dua buah simpul dikatakan berhubungan jika ada suatu lintasan dari simpul pertama ke simpul kedua.

B. Travelling Salesman Problem

Travelling Salesman Problem atau yang biasa disingkat dengan TSP adalah salah satu masalah optimisasi yang paling terkenal di bidang teori graf dan riset operasi. Inti dari permasalahan ini adalah seorang pedagang keliling (*Salesman)* ingin mengunjungi beberapa kota , tiap kota harus dikunjungi tepat satu kali, dan kembali ke kota asal dengan jarak total seminimal mungkin. TSP memiliki satu tujuan utama yaitu:

$$Minimalkan: \sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_i)$$

Di mana:
- x_i adalah urutan kota yang dikunjungi;
- $d(x_i, x_j)$ adalah jarak antara kota i dan j.

TSP sendiri memiliki beberapa komponen yang membangun permasalahan ini yaitu:

1. Graf (*Graph*)
   TSP biasanya direpresentasikan dalam bentuk graf, umumnya graf berbobot, dengan tiap simpul mewakili suatu kota dan sisi yang menghubungkan kedua simpul adalah jarak dari kedua kota tersebut.
2. Bobot (*Weight*)
   TSP pada umumnya memakai graf berbobot. Bobot pada tiap sisi mewakili jarak dari kedua kota. Bobot tersebut dapat dihitung dengan berbagai cara seperti jarak Euclidian ataupun jarak aktual.

Dari komponen-komponen tersebut, TSP dapat dibagi lagi menjadi berbagai jenis yaitu:

1. TSP Simetrik (*Symmetric TSP*)
   Suatu bentuk TSP dikatakan simetrik jika jarak antara kedua kota adalah sama dari kedua arah.

2. TSP Asimetrik (*Asymmetric TSP*)
   Suatu bentuk TSP dikatakan asimetrik jika jarak antara kedua kota berbeda tergantung arah perjalanannya. Hal ini disebabkan oleh perbedaan rute yang diambil dari kota pertama ke kota kedua dan sebaliknya.

Untuk menyelesaikan persoalan TSP, ada beberapa metode yang bisa dilakukan terutama dengan bahasa pemrograman. Metode-metode yang bisa dipakai adalah

1. Metode Brute Force
   Metode Brute Force merupakan salah satu metode yang bisa dipakai untuk menyelesaikan persoalan TSP. Metode ini mencoba semua kemungkinan urutan perjalanan yang ada (dalam bentuk permutasi) untuk mencari suatu solusi yang optimal. Metode ini dapat dengan tepat menemukan solusi, tetapi metode ini memiliki kekurangan karena memakan waktu yang lama yaitu dengan kompleksitas waktu O(n!) dengan n sebagai jumlah kota.

2. *Dynamic Programming*
   Metode *dynamic programming* adalah metode yang mirip dengan metode brute force, tetapi tanpa masalah waktu yang ada. Metode ini menggunakan teknik *memorization* untuk menyimpan hasil perhitungan sebelumnya sehingga dapat mengurangi pengulangan komputasi. Metode ini memiliki kompleksitas waktu sebesar O($n^2 \cdot 2^n$) yang lebih baik dibandingkan brute force.

3. *Nearest Neighbor*
   Algoritma *nearest neighbor* atau biasa disebut juga dengan *greedy algorithm* merupakan algoritma yang membuat sang *salesman* memilih kota selanjutnya dengan jarak terkecil. Algoritma ini dapat dengan cepat menentukan rute yang singkat, namun algoritma ini kurang akurat karena bisa saja melewatkan rute yang jauh lebih singkat daripada yang ditemukan.

4. *Approximation Algorithm*
   Algoritma aproksimasi adalah algortima umum untuk mencari solusi dari permasalahan optimisasi bertipe NP-Hard. Contoh dari algoritma ini adalah *Minimum Spanning Tree (MST)-Based Approximation*.

5. *Integer Linear Programming* (ILP)
   Algoritma ini merepresentasikan TSP dalam bentuk persamaan linear yang akan dihitung menggunakan dua pendekatan popular yaitu Miller-Tucker-Zemlin (MTZ) ataupun Dantzig-Fulkerson-Johnson (DFJ). Keduanya memiliki kekuatan dan kekurangan masing-masing dalam permasalahan ini.

III. IMPLEMENTASI

Untuk mencari rute negara paling optimal, dibuatlah sebuah program dalam bahasa python untuk mesimulasikan permasalahan tersebut. Algoritma yang digunakan adalah *dynamic programming* untuk mendapatkan hasil yang cukup

akurat dengan waktu yang relatif singkat.

Sebelum memulai kedalam perhitungan, setiap negara tempat balapan F1 dilaksanakan direpresentasikan berdasarkan latitude dan longitudenya.

```
# Location of F1 circuits (latitude, longitude)
lokasi = [
    (26.0393, 50.5371), # Bahrain
    (21.4858, 39.1925), # Jeddah
    (-37.8136, 144.9631), # Melbourne
    (35.4437, 139.6380), # Suzuka
    (31.2304, 121.4737), # Shanghai
    (25.7617, -80.1918), # Miami
    (44.3558, 11.7164), # Imola
    (43.7384, 7.4246), # Monaco
    (56.1304, -106.3468), # Canada
    (41.3851, 2.1734), # Barcelona
    (47.2071, 14.7968), # Austria
    (55.3781, -3.4359), # Silverstone
    (47.1625, 19.5033), # Hungary
    (50.5039, 4.4699), # Belgium
    (52.3788, 4.9001), # Zandvoort
    (45.5828, 9.2725), # Monza
    (40.4092, 49.8671), # Baku
    (1.3521, 103.8198), # Singapore
    (30.2672, -97.7431), # Austin
    (23.6345, -102.5528), # Mexico
    (-23.5505, -46.6333), # Interlagos
    (36.1699, -115.1398), # Las Vegas
    (25.3979, 51.5131), # Losail
    (36.6875, -121.7906) # Abu Dhabi
]
```

Gambar 3.1 Koordinat Negara Lokasi Sirkuit F1

Dikarenakan bentuk bumi yang bulat, digunakan perhitungan dengan rumus *Haversine* dengan memanfaatkan nilai radian dari latitude dan longitude setiap negara dan radius dari bumi.untuk menentukan jarak dari kedua negara.

```
def jarak(latitude1, longitude1, latitude2, longitude2):
    # earth radius in km
    R = 6371.0
    # Convert latitude and longitude from degrees to radians
    lat1 = radians(latitude1)
    lon1 = radians(longitude1)
    lat2 = radians(latitude2)
    lon2 = radians(longitude2)
    lon_range = lon2 - lon1
    lat_range = lat2 - lat1
    # Using Haversine formula to calculate distance between two points
    a = sin(lat_range / 2)**2 + cos(lat1) * cos(lat2) * sin(lon_range / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    return R * c
```

Gambar 3.2 Fungsi Jarak Antar Dua Negara

Sebelum masuk ke dalam penyelesaian TSP, kita buat terlebih dahulu matriks jarak antar dua negara untuk mempermudah perhitungan TSP. Dengan memanfaatkan fungsi jarak yang sudah dibuat, kita dapat membuat matriks dengan program berikut.

```
# Create distance matrix
n = len(lokasi)
dist = [[0] * n for _ in range(n)]
for i in range(n):
    for j in range(n):
        if i != j:
            dist[i][j] = jarak(lokasi[i][0], lokasi[i][1], lokasi[j][0], lokasi[j][1])
```

Gambar 3.3 Perhitungan Matriks Jarak

Setelah itu, kita masuk ke program dalam menentukan rute paling optimal yang bisa didapat.

```
# TSP using dynamic programming
dp = [[sys.maxsize] * n for _ in range(1 << n)]
previous_city = [[-1] * n for _ in range(1 << n)]
dp[1][0] = 0  # Starting at city 0

# Iterate over all possible subsets of cities
for mask in range(1 << n):
    for i in range(n):
        if mask & (1 << i):  # If city i is in the mask
            for j in range(n):
                if mask & (1 << j) and i != j:  # If city j is also in the mask
                    new_cost = dp[mask ^ (1 << i)][j] + dist[j][i]
                    if new_cost < dp[mask][i]:
                        dp[mask][i] = new_cost
                        previous_city[mask][i] = j  # Store the previous city for path reconstruction

# Find the optimal route
result = sys.maxsize
last_city = -1
for i in range(1, n):
    cost = dp[(1 << n) - 1][i] + dist[i][0]
    if cost < result:
        result = cost
        last_city = i
```

Gambar 3.4, 3.5, 3.6 Program Menentukan Solusi TSP

Penyelesaian tersebut menggunakan pendekatan brute force dengan *dynamic programming* yang menyimpan perhitungan sebelumnya untuk megurangi perulangan. Program tersebut dimulai dari kota pertama yang dicari jarak minimunya. Kemudian, program akan memilih subset kota yang sudah dikunjungi, kemudian mencoba mencari kota terbaik untuk dikunjungi berikutnya kemudian menghitung jarak total. Setelah menghitung jarak untuk setiap subset kota, dicari jarak minimum yang mengunjungi semua kota dan kembali ke kota asal.

Untuk menampilkan hasil dari perhitungan, dibuatlah program untuk menampilkan hasilnya yaitu

```
# Reconstruct the optimal path
path = []
mask = (1 << n) - 1
current_city = last_city
while current_city != -1:
    path.append(current_city)
    next_city = previous_city[mask][current_city]
    mask ^= (1 << current_city)
    current_city = next_city
path.append(0)
path.reverse()
```

Gambar 3.7 Rekonstruksi Kota Untuk Menampilkan Rute

Setelah menemukan jarak minimum, direkonstruksi jalur optimal dengan mengikuti kota sebelumnya yang nilainya sudah disimpan. Kemudian kita tampilkan hasilnya di layer dengan program

```
# Print results
print("The minimum distance to visit all locations is:", result, "km")
print("The optimal route is:")
for city in path:
    print(f"City {city + 1}: {lokasi[city]}")
```

Gambar 3.8 Program Display Hasil Perhitungan TSP

IV. HASIL DAN PEMBAHASAN

Sebelum kita masuk ke hasil rute optimal dari pengaplikasian TSP, kita perlu mencaritahu terlebih dahulu keadaan rute saat ini sebagai pembanding. Diambil dari website resmi Formula 1 berikut adalah kalender balapan pada tahun 2024.

Gambar 4.1 Kalender Formula 1 2024
Sumber: https://corp.formula1.com/formula-1-announces-calendar-for-2024/

Dari gambar tersebut, kita ketahui bahwa rute yang diambil adalah Bahrain - Saudi Arabia – Australia – Jepang - China – USA(Miami) - Italia (Imola) - Monaco – Canada - Barcelona - Austria – United Kingdom - Hungaria - Belgium - Belanda - Italia (Monza) – Azerbaijan – Singapur – USA(Austin) -Mexico – Brazil – USA(Las Vegas) – Qatar – Abu Dhabi. Rute yang diambil ini setelah melalui perhitungan didapat jarak tempuh sebesar 137.952 km.

```
Total distance traveled: 137952.17 km
```

Gambar 4.2 Jarak Tempuh Rute F1 2024

Setelah diterapkan perhitungan TSP dengan pendekatan *dynamic programming* ditemukan sebuah rute paling optimal sebagai berikut.

```
The optimal route is:
City 1: (26.0393, 50.5371)
City 1: (26.0393, 50.5371)
City 17: (40.4092, 49.8671)
City 4: (35.4437, 139.638)
City 5: (31.2304, 121.4737)
City 18: (1.3521, 103.8198)
City 3: (-37.8136, 144.9631)
City 21: (-23.5505, -46.6333)
City 6: (25.7617, -80.1918)
City 19: (30.2672, -97.7431)
City 20: (23.6345, -102.5528)
City 22: (36.1699, -115.1398)
City 24: (36.6875, -121.7906)
City 9: (56.1304, -106.3468)
City 12: (55.3781, -3.4359)
City 15: (52.3788, 4.9001)
City 14: (50.5039, 4.4699)
City 10: (41.3851, 2.1734)
City 8: (43.7384, 7.4246)
City 16: (45.5828, 9.2725)
City 7: (44.3558, 11.7164)
City 11: (47.2071, 14.7968)
City 13: (47.1625, 19.5033)
City 2: (21.4858, 39.1925)
City 23: (25.3979, 51.5131)
```

Gambar 4.3 Hasil Rute Optimal dari Perhitungan TSP

Dari hasil tersebut didapat rute optimal yaitu Bahrain – Azerbaijan – Jepang – China – Singapur – Australia – Brazil – USA(Miami) – USA(Austin) – Mexico – USA (Las Vegas) – Abu Dabhi – Canada – United Kingdom – Belanda – Belgium – Barcelona – Monaco – Italia (Monza) – Italia (Imola) – Austria – Hungaria – Saudi Arabia – Qatar . Jarak tempuh total yang didapat dari rute optimal ini adalah 62.205 km.

```
The minimum distance to visit all locations is: 62205.32197457123 km
```

Gambar 4.4 Jarak Tempuh dari Rute Optimal

Dari kedua hasil tersebut didapat perbedaan jarak sebesar 75.747 km. Jarak tempuh dari rute yang optimal adalah sekitar 45% dari rute yang dipakai sepanjang 2024. Pengurangan jarak tempuh ini bisa mendakan pengurangan emisi karbon yang dikeluarkan moda transportasi untuk perjalanan sebesar 45%. Selain itu, hal ini juga mengurangi biaya operasional dan logistic dari setiap tim maupun dari Formula 1 itu sendiri.

Walaupun begitu, hasil dari perhitungan ini mengabaikan beberapa faktor lainnya dalam Formula 1 untuk memilih rute balapannya. Faktor tersebut seperti cuaca, musim, dan faktor lainnya seperti mereka yang menghindari melakukan balapan di area Timur Tengah pada masa lebaran.

V. KESIMPULAN

Dari percobaan yang telah dilakukan di atas, dapat disimpulkan bahwa pendekatan *Travelling Salesman Problem* mampu mencari sebuah rute dengan jarak optimal yang melewati beberapa titik tepat satu kali. Selain dari permasalahan kalender Formula 1 ini, pendekatan TSP memang bisa digunakan untuk memecah masalah lainnya yang berkaitan dengan berkeliling dari negara satu ke negara lainnya.

Walaupun begitu, perlu diingat juga bahwa pendekatan TSP ini merupakan masalah optimisasi bertipe NP-Hard yang memerlukan algoritma dengan kompleksitas tinggi untuk mendapat hasil yang akurat. Sehingga untuk memecahkan masalah dalam skala besar tidak bisa dilakukan dalam waktu yang singkat, tetapi jika dalam skala kecil, pendekatan TSP ini sangat layak untuk digunakan.

VII. UCAPAN TERIMA KASIH

Dengan ini, sang penulis menyampaikan ucapan terimakasih sebesar-besarnya kepada berbagai pihak yang telah membantu sang penulis dalam proses pembuatan makalah ini. Pertama, sang penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas berkatnya yang membimbing dan menguatkan selama proses belajar dan menulis makalah ini. Sang penulis juga mengucapkan terimakasih yang tulus kepada pengajar kelas K-02 mata kuliah Matematika Diskrit IF1220, Pak Rila Mandala, atas bimbingan dan pengajarannya selama satu semester. Tak lupa juga sang penulis mengucapkan terimakasih kepada keluarga serta teman-teman yang senantiasa mendukung dan membantu proses belajar sang penulis selama satu semester.

REFERENCES

[1] Zhang, P. and Chartrand, G., 2006. Introduction to graph theory. *Tata McGraw-Hill*, *2*, pp.2-1.
[2] Gavish, B. and Graves, S.C., 1978. The travelling salesman problem and related problems.
[3] Goyal, S., 2010, April. A survey on travelling salesman problem. In *Midwest instruction and computing symposium* (pp. 1-9).
[4] Antropow De la Hoz, T., 2024. The Race Towards Sustainability: A Study of Formula 1's Race Calendar Distribution and Net Zero Carbon Goals for 2030-Antropow de la Hoz, Teresa.
[5] Chauhan, C., Gupta, R. and Pathak, K., 2012. Survey of methods of solving tsp along with its implementation using dynamic programming approach. *International journal of computer applications*, *52*(4)..

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, 26 Desember 2024

Samuel Gerrard Hamonangan Girsang
13523064