Optimasi Tutorial Aplikasi Gojek dengan Pendekatan Travelling Salesman Problem

Andi Farhan Hidayat - 13523128[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*andifarhan1094@gmail.com*, *13523128@std.stei.itb.ac.id*

***Abstrak*—Kemajuan teknologi digital mendorong pesatnya penggunaan aplikasi layanan seperti Gojek yang menawarkan berbagai fitur mulai dari transportasi hingga pemesanan makanan. Meskipun kaya akan fitur, aplikasi ini seringkali menantang bagi pengguna baru dalam hal pemahaman dan navigasi. Oleh karena itu, diperlukan tutorial yang efisien dan mudah dipahami. Dalam makalah ini, penulis mengusulkan penggunaan pendekatan Travelling Salesman Problem (TSP) untuk mengoptimalkan urutan langkah-langkah tutorial aplikasi Gojek. Pendekatan ini bertujuan meminimalkan waktu dan usaha yang dibutuhkan pengguna untuk mempelajari aplikasi dengan menyusun urutan langkah tutorial yang optimal. Setiap fitur dalam aplikasi diwakili sebagai simpul dalam graf berbobot, dengan bobot menggambarkan jumlah klik yang diperlukan untuk berpindah antar fitur. Penulis menerapkan algoritma *brute force* untuk mencari urutan langkah optimal, namun ditemukan bahwa urutan tutorial sangat dipengaruhi oleh urutan input data *adjacency matrix* yang menyebabkan sistem gagal menghasilkan urutan optimal pada data yang tidak terurut. Untuk mengatasi masalah ini, penulis memberikan beberapa saran, termasuk penerapan algoritma yang lebih stabil seperti *Dynamic Programming* (DP) atau *Branch and Bound*, serta penggunaan algoritma heuristik untuk skala besar. Penelitian ini diharapkan dapat memberikan kontribusi pada optimasi tutorial pada aplikasi yang lebih efektif, meningkatkan pengalaman pengguna, serta memberikan dasar untuk penelitian selanjutnya dalam penerapan TSP pada optimasi tutorial aplikasi.**

***Kata Kunci*— Travelling Salesman Problem, TSP, optimasi tutorial, Gojek, algoritma heuristik, Dynamic Programming, Branch and Bound, Brute Force.**

I. PENDAHULUAN

Kemajuan teknologi di era digital telah menjadikan aplikasi layanan seperti Gojek sebagai bagian penting dari kehidupan masyarakat modern. Gojek menyediakan berbagai layanan, mulai dari transportasi, pengiriman barang, hingga pemesanan makanan, sehingga membuatnya menjadi aplikasi dengan fitur yang sangat beragam. Namun, kompleksitas tersebut seringkali menjadi tantangan bagi pengguna baru untuk memahami cara kerja aplikasi secara menyeluruh. Oleh sebab itu, pengembang aplikasi harus merancang tutorial yang mudah diikuti dan efisien, sehingga pengguna dapat dengan cepat menguasai fitur-fitur yang ada.

Keberhasilan efisiensi sebuah tutorial aplikasi sangat bergantung pada urutan langkah-langkah yang disajikan. Dalam hal ini, Travelling Salesman Problem (TSP) merupakan pendekatan yang sesuai. TSP adalah sebuah masalah optimasi yang bertujuan menentukan rute terpendek untuk mengunjungi sejumlah titik hanya sekali sebelum kembali ke titik awal [1]. Dalam sebuah tutorial aplikasi, setiap fitur yang ada harus diperkenalkan kepada pengguna dengan urutan *user journey* yang seminimum mungkin. Dengan memanfaatkan konsep ini, penyusunan tutorial dapat dioptimalkan sehingga pengguna dapat mempelajari aplikasi dengan waktu dan usaha yang lebih sedikit. Pendekatan tersebut dapat meminimalkan kebingungan pengguna, memperbaiki pengalaman pembelajaran, serta meningkatkan adopsi aplikasi.

Penelitian terdahulu menunjukkan bahwa TSP telah berhasil diterapkan di berbagai bidang, seperti optimasi rute logistik dan perencanaan jalur wisata [3][4]. Selain itu, dalam teknologi informasi, algoritma berbasis TSP telah digunakan untuk mengelola proses kompleks seperti e-learning [2]. Studi-studi tersebut menunjukkan bahwa penerapan TSP mampu meningkatkan efisiensi dan mengurangi waktu yang diperlukan dalam menyelesaikan suatu proses.

Makalah ini berfokus pada penerapan pendekatan TSP untuk menyusun tutorial aplikasi Gojek secara optimal. Dengan pendekatan ini, tutorial aplikasi Gojek diharapkan dapat membantu pengguna memahami fungsi utama aplikasi secara lebih efektif dan efisien. Selanjutnya, makalah ini akan mengeksplorasi implementasi TSP dalam konteks ini dan memberikan rekomendasi strategis untuk optimasi tutorial aplikasi.

II. DASAR TEORI

A. Graf

1. Definisi

Graf adalah salah satu struktur dasar dalam matematika diskrit yang digunakan untuk merepresentasikan hubungan antar objek. Sebuah graf terdiri dari himpunan simpul (*vertices*) dan himpunan sisi (*edges*) yang menghubungkan pasangan simpul tersebut. Graf dapat diklasifikasikan menjadi beberapa jenis, seperti graf berarah dan graf tak berarah. Dalam graf berarah, setiap sisi memiliki arah tertentu, sedangkan dalam graf tak berarah, sisi tidak memiliki arah tertentu [5]. Graf juga dapat berbobot, yaitu setiap sisi memiliki nilai atau bobot tertentu yang biasanya merepresentasikan jarak, waktu, atau biaya.

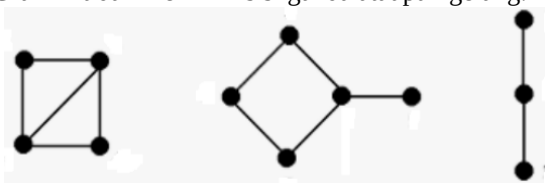
Graf memiliki banyak aplikasi, termasuk dalam perencanaan

jaringan, pengelolaan data, dan pemodelan berbagai masalah optimasi. Dalam konteks Travelling Salesman Problem (TSP), graf berbobot digunakan untuk merepresentasikan kota-kota sebagai simpul dan jarak antar kota sebagai sisi berbobot. Representasi ini memungkinkan algoritma TSP untuk mencari rute optimal berdasarkan kriteria tertentu.

2. Jenis-Jenis Graf

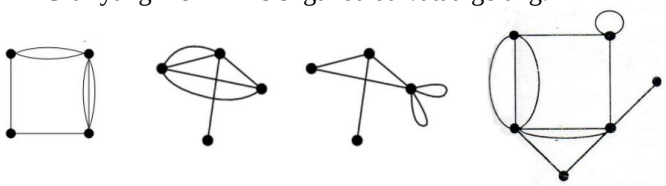
Graf memiliki banyak jenis sesuai dengan bentuk, ada tidaknya arah, dan pola. Berdasarkan keberadaan sisi ganda ataupun gelang di dalam graf, maka graf dapat digolongkan menjadi dua macam:

- Graf sederhana
  Graf ini tidak memiliki sisi ganda ataupun gelang.



Gambar 2.1.1. Contoh Graf Sederhana.
(Sumber: [7])

- Graf tak-sederhana
  Graf yang memiliki sisi ganda dan/atau gelang.



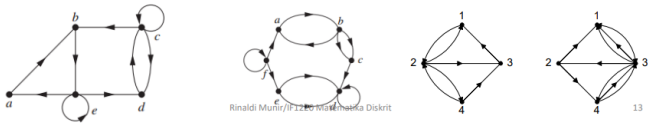
Gambar 2.1.2. Contoh Graf Tak-Sederhana.
(Sumber: [7])

Selain itu, graf tak-sederhana juga dikelompokkan lagi menjadi graf ganda dan graf semu.

- Graf ganda
  Graf yang memiliki sisi ganda. Graf ini juga disebut sebagai *multi-graph*.

- Graf semu
  Graf yang memiliki sisi gelang. Graf ini dikenal juga sebagai *pseudo-graph*.

Graf-graf sebelumnya merupakan contoh jenis graf yang tak-berarah. Terdapat pula graf yang memiliki arah yang disebut sebagai graf berarah atau dikenal juga sebagai *directed graph* atau *digraph*. Graf berarah memiliki orientasi arah untuk setiap sisi pada grafnya dan memiliki bentuk sederhana, gelang, atau bahkan sisi ganda.



Gambar 2.1.3. Contoh Graf Berarah.
(Sumber: [7])

Sebaliknya, graf yang tidak memiliki arah pada sisi grafnya disebut sebagai graf tak-berarah atau dikenal juga sebagai *undirected graph.*

Jenis	Sisi	Sisi ganda dibolehkan?	Sisi gelang dibolehkan?
Graf sederhana	Tak-berarah	Tidak	Tidak
Graf ganda	Tak-berarah	Ya	Tidak
Graf semu	Tak-berarah	Ya	Ya
Graf berarah	Berarah	Tidak	Ya
Graf-ganda berarah	Berarah	Ya	Ya

Gambar 2.1.4. Tabel Jenis-Jenis Graf.
(Sumber: [7])

3. Terminologi

Dalam graf, terdapat beberapa terminologi umum yang digunakan untuk mendeskripsikan bentuk, pola, atau sifat dari suatu graf. Terminologi-terminologi tersebut adalah sebagai berikut:

- Ketetanggaan: Dua simpul dikatakan bertetangga jika keduanya dihubungkan oleh sebuah sisi.
- Bersisian: Sebuah sisi dikatakan bersisian dengan simpul-simpul yang menjadi ujungnya.
- Simpul Terpencil: Simpul yang tidak memiliki sisi yang terhubung dengannya.
- Graf Kosong: Graf tanpa sisi sama sekali, hanya terdiri dari simpul.
- Derajat (*Degree*): Jumlah sisi yang terhubung dengan sebuah simpul. Untuk graf berarah, terdapat derajat masuk (*in-degree*) dan derajat keluar (*out-degree*).
- Lintasan (*Path*): Urutan simpul-simpul yang masing-masing dihubungkan oleh sisi-sisi dalam graf.
- Sirkuit (*Circuit*): Lintasan yang dimulai dan berakhir pada simpul yang sama.
- Keterhubungan (*Connectivity*): Graf dikatakan terhubung jika terdapat lintasan antara setiap pasangan simpul.
- Upagraf (*Subgraph*): Sebuah graf yang merupakan bagian dari graf lain, terdiri dari sebagian simpul dan sisi dari graf tersebut.
- Upagraf Merentang (*Spanning Subgraph*): Upagraf yang mencakup semua simpul dari graf induknya.
- Cut-Set: Himpunan sisi yang jika dihapus akan memutuskan keterhubungan graf.
- Graf Berbobot: Graf di mana setiap sisi memiliki nilai atau bobot tertentu.

4. Representasi Graf

Graf dapat direpresentasikan dalam berbagai cara untuk mempermudah analisis dan implementasi algoritma. Berikut adalah tiga metode representasi graf yang umum digunakan:

- Matriks Ketetanggaan (*Adjacency Matrix*)
  Matriks dua dimensi dengan ukuran n x n, di mana n adalah jumlah simpul dalam graf. Elemen pada baris i dan kolom j menunjukkan apakah terdapat sisi yang menghubungkan simpul i dan simpul j. Untuk graf berbobot, elemen matriks berisi bobot sisi tersebut. Keuntungan utama dari representasi ini adalah akses cepat untuk memeriksa keberadaan sisi, tetapi membutuhkan ruang penyimpanan yang besar untuk graf yang jarang (sparse).
- Matriks Bersisian (*Incidence Matrix*)
  Matriks dua dimensi dengan ukuran n x m, di mana n

adalah jumlah simpul dan m adalah jumlah sisi. Elemen pada baris i dan kolom j menunjukkan apakah simpul i terhubung dengan sisi j. Dalam graf berarah, matriks ini juga dapat mencatat arah sisi dengan menggunakan nilai +1 atau -1 untuk menunjukkan derajat masuk dan keluar.

- Senarai Ketetanggaan (*Adjacency List*)
  Representasi yang menggunakan struktur data seperti daftar atau array untuk mencatat simpul-simpul yang bertetangga dengan setiap simpul. Metode ini sangat efisien untuk graf yang jarang karena hanya mencatat simpul yang memiliki hubungan langsung. Namun, akses untuk memeriksa keberadaan sisi membutuhkan waktu lebih lama dibandingkan matriks ketetanggaan.

5. Graf Euler

Graf Euler adalah graf yang memiliki sirkuit Euler. Graf yang memiliki lintasan Euler juga disebut sebagai graf semi-Euler. Lintasan Euler adalah sebuah lintasan yang melalui tiap sisi pada graf tepat satu kali saja. Sedangkan sirkuit Euler adalah sebuah sirkuit yang melalui tiap sisi pada graf tepat satu kali.

6. Graf Hamilton

Graf Hamilton adalah graf yang memiliki sirkuit Hamilton. Graf yang memiliki lintasan Hamilton disebut juga sebagai graf semi-Hamilton. Lintasan Hamilton adalah sebuah lintasan yang melalui semua simpul (*vertex*) di dalam sebuah graf tepat satu kali. Sedangkan sirkuit Hamilton adalah sebuah sirkuit yang melalui semua simpul pada sebuah graf tepat sebanyak satu kali.

B. Traveling Salesman Problem

Travelling Salesman Problem (TSP) adalah salah satu masalah klasik dalam bidang optimasi kombinatorial. Masalah ini pertama kali diidentifikasi dalam konteks pengiriman barang, yaitu seorang penjual keliling harus mengunjungi sejumlah kota dengan jarak antar kota yang telah ditentukan dan kembali ke kota asal dengan tujuan meminimalkan total jarak yang ditempuh [3]. TSP merupakan masalah NP-Hard yang berarti tidak ada algoritma efisien yang dapat menjamin solusi optimal untuk semua kasus dalam waktu polinomial.

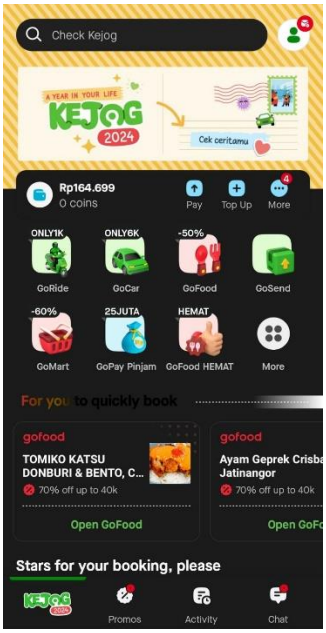
TSP memiliki berbagai variasi dan pendekatan solusi, termasuk metode heuristik seperti algoritma genetika, simulated annealing, dan algoritma semut [1]. Dalam implementasi praktis, TSP digunakan dalam perencanaan rute transportasi, perencanaan jaringan komputer, dan bahkan dalam optimasi pembelajaran, seperti sistem e-learning adaptif yang memanfaatkan pendekatan TSP untuk menentukan urutan materi pembelajaran [2].

Pada Travelling Salesman Problem, terdapat keterkaitan dengan graf Hamilton. Sebab pada persoalan ini memiliki tujuan untuk mengunjungi semua simpul tepat satu kali dalam sebuah siklus atau sirkuit. Hal tersebut sesuai dengan definisi dari graf Hamilton. Oleh sebab itu, Travelling Salesman Problem dapat diselesaikan dengan mencari graf Hamilton yang memiliki total bobot minimum.

C. Gojek

Gojek adalah perusahaan teknologi asal Indonesia yang menyediakan layanan *on-demand* melalui aplikasi, termasuk transportasi, pengiriman barang, pengiriman makanan, dan berbagai layanan lainnya. Gojek didirikan oleh Nadiem Makarim pada tahun 2010, Gojek awalnya berfokus pada layanan ojek *online* untuk mengatasi masalah transportasi di Jakarta. Seiring waktu, Gojek berkembang menjadi *platform* multifungsi dengan puluhan layanan yang membantu kehidupan sehari-hari penggunanya [6].

Sebagai aplikasi dengan berbagai fitur, Gojek menghadapi tantangan dalam memberikan pengalaman pengguna yang optimal, khususnya bagi pengguna baru. Salah satu pendekatan untuk mengatasi masalah ini adalah dengan menyusun tutorial yang efisien, sehingga pengguna dapat memahami cara kerja aplikasi dengan mudah. Dalam konteks ini, Travelling Salesman Problem dapat digunakan untuk mengoptimalkan urutan langkah-langkah dalam tutorial dan memastikan pengguna mendapatkan informasi yang relevan dengan waktu yang minimal.



Gambar 2.3.1. Tampilan *Homepage* Aplikasi Gojek.
(Sumber: Dokumen Pribadi Penulis)

III. METODOLOGI

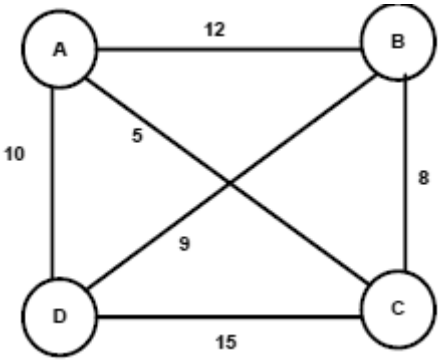
A. Rancangan Rencana Implementasi

Dalam pengaplikasian Travelling Salesman Problem dalam optimasi tutorial aplikasi Gojek, setiap fitur aplikasi akan diperlakukan sebagai simpul (vertex) dalam graf. Setiap simpul yang berupa fitur akan dibentuk menjadi graf lengkap berbobot. Dengan bobotnya adalah jumlah klik dari simpul fitur A ke simpul fitur B. Contohnya, pada fitur *homepage* dengan fitur daftar resto GoFood, jumlah klik yang dilakukan cukup satu kali sehingga bobot dari fitur *homepage* ke fitur daftar resto GoFood adalah satu. Terdapat beberapa langkah untuk menghasilkan graf lengkap berbobot dari fitur-fitur yang ada di Gojek, langkah-langkah tersebut adalah sebagai berikut:

1. *Memodelkan Representasi Fitur Aplikasi sebagai Graf*
   Setiap fitur utama dalam aplikasi Gojek (seperti transportasi, pengiriman barang, dan pemesanan makanan) dianggap sebagai simpul dalam graf. Antara setiap dua fitur, akan ada sebuah sisi yang menghubungkan keduanya. Sisi-sisi ini akan diberi bobot berdasarkan jumlah klik yang diperlukan untuk berpindah dari satu fitur ke fitur lainnya dalam tutorial. Tiap fitur akan dibuat representasinya pada tiap simpul.

2. *Menentukan Bobot pada Setiap Sisi*
   Bobot pada setiap sisi graf dihitung berdasarkan jumlah klik yang diperlukan untuk berpindah dari satu fitur (simpul) ke fitur lainnya. Dalam konteks tutorial aplikasi, jumlah klik ini menggambarkan waktu dan usaha yang diperlukan oleh pengguna untuk memahami dan mengakses fitur-fitur tertentu.

3. *Pembentukan Graf Lengkap*
   Graf yang terbentuk adalah graf lengkap, yang berarti bahwa setiap simpul (fitur) terhubung dengan setiap simpul lainnya. Dengan demikian, semua kemungkinan urutan langkah dalam tutorial dapat dianalisis. Setiap sisi yang menghubungkan dua fitur akan memiliki bobot yang sesuai dengan jumlah klik yang dibutuhkan untuk berpindah antar fitur tersebut.



Gambar 3.1.1. Contoh Graf Lengkap Tiap Fitur.
(Sumber: [10])

4. *Penerapan Algoritma TSP*
   Setelah graf lengkap dibentuk, algoritma Travelling Salesman Problem (TSP) diterapkan untuk menemukan urutan langkah-langkah dalam tutorial yang meminimalkan total jumlah klik. TSP akan menghasilkan jalur yang mengunjungi setiap fitur tepat sekali, dimulai dari titik awal dan kembali ke titik tersebut. Jalur yang dihasilkan ini adalah urutan tutorial yang optimal, dengan tujuan mengurangi waktu yang dibutuhkan oleh pengguna untuk mempelajari aplikasi.

5. *Evaluasi dan Validasi*
   Setelah mendapatkan urutan langkah optimal dari algoritma Travelling Salesman Problem (TSP), langkah selanjutnya adalah evaluasi. Pengujian dilakukan untuk memastikan bahwa urutan tutorial yang dihasilkan benar-benar meminimalkan kebingungan pengguna dan meningkatkan pengalaman pembelajaran. Evaluasi dapat dilakukan dengan meminta sekelompok pengguna untuk mengikuti tutorial yang dihasilkan dan mengukur waktu yang mereka butuhkan untuk memahami aplikasi serta seberapa mudah mereka menavigasi antara fitur-fitur utama.

6. *Penyusunan Tutorial dan Rekomendasi*
   Berdasarkan hasil TSP, tutorial disusun sesuai dengan urutan yang optimal. Penulis kemudian memberikan rekomendasi untuk penyempurnaan lebih lanjut, baik dari sisi desain tutorial maupun interaksi pengguna. Penekanan pada kemudahan akses dan efisiensi waktu adalah kunci dalam membuat tutorial yang efektif.

Dengan penerapan metode ini, diharapkan tutorial aplikasi Gojek dapat dioptimalkan, sehingga pengguna baru dapat mempelajari aplikasi dengan lebih cepat dan efisien, tanpa mengalami kebingungan dalam proses navigasi antar fitur.

B. Implementasi Kode

Dalam mengimplementasikan optimasi tutorial aplikasi Gojek menggunakan pendekatan Travelling Salesman Problem (TSP), digunakan bahasa pemrograman Python menggunakan modul itertools sebagai bantuan. Modul itertools adalah modul di Python yang menyediakan fungsi-fungsi untuk bekerja dengan iterator yang memungkinkan untuk melakukan manipulasi dan menghasilkan kombinasi, permutasi, dan berbagai urutan data lainnya secara efisien.

Terdapat beberapa fungsi yang digunakan dalam menentukan rute sirkuit Hamilton untuk Travelling Salesman Problem pada optimasi tutorial aplikasi Gojek. Fungsi-fungsi tersebut adalah sebagai berikut:

1. *calculate_total_clicks(order, dist_matrix)*
   Fungsi ini bertugas untuk menghitung total jumlah klik (atau bobot) yang diperlukan untuk mengikuti suatu urutan langkah dalam tutorial berdasarkan matriks jarak (klik) antara fitur-fitur aplikasi. Fungsi ini menerima dua parameter, yaitu *order* yang merupakan urutan indeks fitur yang akan dikunjungi dan *dist_matrix* yang merupakan matriks berbobot berisi jumlah klik antar fitur. Fungsi ini menghitung total klik dengan menjumlahkan bobot antara dua fitur berturut-turut dalam urutan, serta menambahkan bobot dari fitur terakhir kembali ke fitur pertama untuk memastikan jalur membentuk sirkuit. Hasil dari fungsi ini adalah total klik yang diperlukan untuk mengikuti urutan langkah tersebut.

```python
# Fungsi untuk menghitung total klik (bobot) untuk sebuah urutan
def calculate_total_clicks(order, dist_matrix):
    total_clicks = 0
    for i in range(len(order) - 1):
        total_clicks += dist_matrix[order[i]][order[i+1]]
    total_clicks += dist_matrix[order[-1]][order[0]]  # Kembali ke titik awal
    return total_clicks
```

Gambar 3.2.1. Fungsi *calculate_total_clicks*.
(Sumber: Dokumen Pribadi Penulis)

2. *tsp_bruteforce(dist_matrix)*
   Fungsi ini adalah implementasi dari algoritma *brute force* untuk menyelesaikan masalah Travelling Salesman Problem (TSP). Fungsi ini menerima parameter *dist_matrix* yang berisi matriks jarak antar fitur aplikasi dan menghasilkan urutan optimal fitur yang meminimalkan total klik. Fungsi ini bekerja dengan menghasilkan semua kemungkinan urutan langkah yang dapat diambil menggunakan *itertools.permutations*, lalu

menghitung total klik untuk setiap urutan dengan memanggil fungsi *calculate_total_clicks*. Selanjutnya, fungsi ini membandingkan total klik untuk menemukan urutan yang memiliki jumlah klik terkecil yang dianggap sebagai solusi optimal. Fungsi ini mengembalikan urutan fitur yang optimal beserta total klik yang diperlukan.

```python
# Fungsi untuk mencari jalur optimal menggunakan algoritma brute force
def tsp_bruteforce(dist_matrix):
    # Membuat semua kemungkinan urutan
    n = len(dist_matrix)
    all_permutations = itertools.permutations(range(n))

    # Menyimpan hasil terbaik
    min_clicks = float('inf')
    best_order = None

    # Memeriksa setiap permutasi untuk menemukan urutan terbaik
    for order in all_permutations:
        total_clicks = calculate_total_clicks(order, dist_matrix)
        if total_clicks < min_clicks:
            min_clicks = total_clicks
            best_order = order

    return best_order, min_clicks
```

Gambar 3.2.2. Fungsi *tsp_bruteforce*.
(Sumber: Dokumen Pribadi Penulis)

Selain itu, terdapat artefak-artefak lainnya yang mendukung implementasi optimasi tutorial aplikasi Gojek dengan pendekatan Travelling Salesman Problem (TSP). Artefak kode tersebut meliputi:

1. *Matrix dist_matrix*
   Matriks ini menggambarkan hubungan antara fitur-fitur aplikasi dalam bentuk bobot (jumlah klik) yang diperlukan untuk berpindah dari satu fitur ke fitur lainnya. Matriks *dist_matrix* adalah representasi graf lengkap berbobot, dengan setiap elemen *dist_matrix[i][j]* menunjukkan jumlah klik yang diperlukan untuk berpindah dari fitur i ke fitur j. Nilai-nilai di sepanjang diagonal matriks (seperti *dist_matrix[i][i]*) diatur menjadi 0 karena tidak ada klik yang diperlukan untuk tetap berada pada fitur yang sama. Matriks ini berfungsi sebagai masukan bagi algoritma TSP untuk menentukan urutan langkah optimal dalam tutorial.

```python
# Matriks jarak (jumlah klik) antar fitur aplikasi
dist_matrix = [
    [0, 2, 9, 10, 1],  # Fitur A
    [2, 0, 6, 4, 3],   # Fitur B
    [9, 6, 0, 3, 7],   # Fitur C
    [10, 4, 3, 0, 8],  # Fitur D
    [1, 3, 7, 8, 0],   # Fitur E
]
```

Gambar 3.2.3. Contoh Matriks *dist_matrix*.
(Sumber: Dokumen Pribadi Penulis)

2. *Array features*
   Array tersebut berisi nama-nama fitur aplikasi (misalnya "Pemesanan Makanan", "Transportasi", dll.) yang diindeks dari 0 hingga n-1. Setelah algoritma TSP menemukan urutan optimal fitur dalam bentuk indeks (misalnya (4, 0, 1, 3, 2)), array *features* mengonversi indeks-indeks tersebut menjadi nama fitur yang lebih mudah dipahami. Hal ini memungkinkan untuk menampilkan hasil urutan tutorial yang lebih jelas dan mudah dipahami oleh pengguna atau pengembang, berupa nama fitur. Contoh array *features* adalah sebagai berikut:
   features = ['Pemesanan Makanan', 'Transportasi', 'Pengiriman Barang', 'Pembayaran', 'Akun Pengguna'].

IV. HASIL DAN PEMBAHASAN

A. Hasil Berbagai Skenario

1. Matrix dist_matrix terurut

Berikut adalah data *dist_matrix*, *features*, dan keluarannya.

```python
dist_matrix = [
    [0, 1, 2, 3, 1, 2, 3, 4, 1],  # Home Page
    [1, 0, 1, 2, 3, 4, 5, 6, 3],  # GoRide
    [2, 1, 0, 1, 2, 3, 4, 5, 4],  # Pilih Destinasi GoRide
    [3, 2, 1, 0, 3, 4, 5, 6, 5],  # Pesan GoRide
    [1, 3, 2, 3, 0, 1, 2, 3, 6],  # GoFood
    [2, 4, 3, 4, 1, 0, 1, 2, 5],  # Pilih Resto GoFood
    [3, 5, 4, 5, 2, 1, 0, 1, 6],  # Pilih Makanan Resto
    [4, 6, 5, 6, 3, 2, 1, 0, 7],  # Pesan Makanan
    [1, 3, 4, 5, 6, 5, 6, 7, 0]   # Profile
]
```

Gambar 4.1.1. *dist_matrix* terurut dengan keterangan *features*.
(Sumber: Dokumen Pribadi Penulis)

```
Urutan optimal fitur aplikasi (indeks fitur): (0, 4, 5, 6, 7, 2, 3, 1, 8)
Total klik (waktu yang dibutuhkan): 16
Urutan optimal fitur aplikasi: ['Home Page', 'GoFood', 'Pilih Resto
GoFood', 'Pilih Makanan Resto', 'Pesan Makanan', 'Pilih Destinasi GoRide',
'Pesan GoRide', 'GoRide', 'Profile']
```

Gambar 4.1.2. Keluaran hasil *dist_matrix* terurut
(Sumber: Dokumen Pribadi Penulis)

2. Matrix dist_matrix tidak terurut

Berikut adalah data *dist_matrix* dengan keterangan *features* dan keluarannya.

```python
dist_matrix = [
    [0, 2, 1, 4, 1, 3, 2, 1, 3],  # Home Page
    [2, 0, 3, 2, 1, 5, 4, 3, 1],  # Pilih Resto GoFood
    [1, 3, 0, 5, 2, 2, 1, 2, 4],  # GoRide
    [4, 2, 5, 0, 3, 7, 6, 5, 1],  # Pesan Makanan
    [1, 1, 2, 3, 0, 4, 3, 2, 2],  # GoFood
    [3, 5, 2, 7, 4, 0, 1, 4, 6],  # Pesan GoRide
    [2, 4, 1, 6, 3, 1, 0, 3, 5],  # Pilih Destinasi GoRide
    [1, 3, 2, 5, 2, 4, 3, 0, 4],  # Profile
    [3, 1, 4, 1, 2, 6, 5, 4, 0]   # Pilih Makanan Resto
]
```

Gambar 4.1.3. *dist_matrix* tidak terurut dengan keterangan *features*.
(Sumber: Dokumen Pribadi Penulis)

```
Urutan optimal fitur aplikasi (indeks fitur): (0, 1, 3, 8, 4, 2, 5, 6, 7)
Total klik (waktu yang dibutuhkan): 16
Urutan optimal fitur aplikasi: ['Home Page', 'Pilih Resto GoFood', 'Pesan
Makanan', 'Pilih Makanan Resto', 'GoFood', 'GoRide', 'Pesan GoRide',
'Pilih Destinasi GoRide', 'Profile']
```

Gambar 4.1.4. Keluaran hasil *dist_matrix* tidak terurut.

(Sumber: Dokumen Pribadi Penulis)

B. *Pembahasan*

Pada percobaan dengan *dist_matrix* terurut dihasilkan data yang sesuai dengan seharusnya. Akan tetapi, ketika dilakukan percobaan dengan *dist_matrix* yang tidak terurut dihasilkan data yang acak. Hal tersebut tentu tidak sesuai dengan yang seharusnya. Walaupun keduanya memiliki data yang sama dan menghasilkan total klik minimum yang sama, yaitu 16. Akan tetapi, hasil urutan kedua percobaan tersebut berbeda. Hal ini dapat terjadi karena algoritma yang digunakan adalah algoritma heuristis atau berbasis pencarian solusi lokal. Algoritma ini tidak selalu menemukan solusi optimal global karena bersifat membuat keputusan berdasarkan kondisi saat itu tanpa mempertimbangkan seluruh jalur yang mungkin. Algoritma yang digunakana adalah algoritma brute force dan ini menyebabkan masalah dalam pengurutannya. Algoritma brute force akan mengevaluasi semua permutasi titik dalam matriks dan dengan *dist_matrix* yang terurut, maka akan didapatkan permutasi dengan urutan lebih sistematis atau lebih terstruktur yang lebih cepat menuju solusi optimal. Sedangkan pada *dist_matrix* yang tidak terurut bisa menyebabkan algoritma memilih permutasi yang lebih acak yang mungkin tidak langsung menuju urutan yang optimal atau logis, meskipun total klik akhirnya sama.

V. KESIMPULAN DAN SARAN

Optimasi tutorial aplikasi Gojek menggunakan pendekatan Travelling Salesman Problem (TSP) merupakan hal yang menjanjikan sebab dapat memberikan urutan yang lebih efisien dalam tutorial aplikasi. Akan tetapi, pada implementasi makalah ini ditemukan permasalahan pada penggunaan TSP sebagai pendekatan optimasi tutorial aplikasi, yaitu jika algoritma yang digunakana adalah algoritma heuristis seperti *brute force*, maka urutan optimal tutorial berpengaruh juga pada urutan dari data *adjacency matrix* sebagai data masukan algoritma. Jika masukan matriks terurut maka sistem akan menjalankan tugasnya dengan baik, akan tetapi jika urutannya acak maka sistem akan gagal dalam melakukan pengurutan paling optimal.

Pada pengaplikasiannya, penulis menemukan beberapa saran untuk mendapatkan hasil pengurutan yang lebih baik untuk penelitian selanjutnya, yaitu:

1. Penerapan Algoritma Lain yang Lebih Stabil
   Salah satu saran utama adalah mengganti algoritma *brute force* dengan algoritma optimasi yang lebih stabil dan terstruktur, seperti *Dynamic Programming* (DP) atau *Branch and Bound*. Algoritma-algoritma ini lebih efisien dalam menangani masalah TSP dan tidak terlalu terpengaruh oleh urutan input data yang acak karena keduanya secara sistematis mengeksplorasi solusi yang optimal tanpa bergantung pada urutan awal. Penggunaan algoritma ini dapat memastikan bahwa sistem menghasilkan urutan optimal meskipun urutan input dalam *adjacency matrix* acak.
2. Penggunaan Heuristik atau Algoritma Aproksimasi
   Mengingat TSP adalah masalah NP-Hard, maka untuk aplikasi dengan skala besar, algoritma heuristik seperti *Nearest Neighbor*, *Genetic Algorithm*, atau *Simulated Annealing* dapat digunakan. Algoritma ini dapat memberikan solusi yang mendekati optimal dalam waktu yang lebih singkat, meskipun tidak selalu menemukan solusi yang benar-benar optimal. Dengan demikian, meskipun urutan input acak, heuristik ini dapat memberikan solusi yang baik dalam waktu yang efisien.
3. Pemberian Batasan Pembentukan Graf Hamilton
   Dengan memberikan batasan yang jelas, contohnya jalur harus dari bobot terendah, maka dapat ditentukan urutan yang lebih terperinci dan sesuai. Dengan pemberian batasan tersebut pula maka variasi jalur yang didapatkan dapat diperkecil menuju jalur-jalur yang paling optimal.

VI. LAMPIRAN

Kode sumber lengkap dalam makalah ini dapat diakses melalui *link repository* GitHub berikut https://github.com/andi-frame/TSP-Tutorial-Gojek.

VII. UCAPAN TERIMA KASIH

Penulis ingin menyampaikan rasa syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, yang memungkinkan penulis untuk menyelesaikan makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Pak Arrival Dwi Sentosa, S.Kom., M.T., dan Pak Dr. Ir. Rinaldi Munir, M.T., sebagai dosen mata kuliah IF1220 Matematika Diskrit yang telah memberikan bimbingan, saran, serta masukan yang sangat berharga selama proses pembelajaran dan penulisan makalah ini. Dukungan dan pengetahuan yang diberikan sangat membantu penulis dalam menyelesaikan tugas ini. Penulis juga ingin mengucapkan terima kasih yang mendalam kepada orang tua dan keluarga yang senantiasa memberikan dukungan moral dan materiil serta doa tanpa henti. Tidak lupa, penulis juga berterima kasih kepada teman-teman dan rekan-rekan seperjuangan yang selalu memberikan semangat dan bantuan, baik secara langsung maupun tidak langsung, dalam proses penyusunan makalah ini. Penulis menyadari bahwa makalah ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang konstruktif sangat diharapkan untuk perbaikan di masa depan. Terima kasih.

REFERENSI

[1] Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). The traveling salesman problem: A computational study. Princeton University Press.
[2] Chen, C. M., Liu, H., & Chang, C. Y. (2016). Personalized e-learning system using adaptive learning and TSP-based path optimization. Computers & Education, 88, 241-256.
[3] Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6(1), 80-91.
[4] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1985). The traveling salesman problem: A guided tour of combinatorial optimization. Wiley.
[5] Budayasa, I. K. (2007). Teori Graf dan Aplikasinya. Surabaya: Unesa University Press.
[6] Gojek. (n.d.). About Us. Diambil dari https://www.gojek.com/en-id/about/ (Diakses 08 Januari 2025)
[7] Munir, Rinaldi. 2024. "Graf Bagian 1". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf. (Diakses 08 Januari 2025)

[8] Munir, Rinaldi. 2024. “Graf Bagian 2”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf. (Diakses 08 Januari 2025)
[9] Munir, Rinaldi. 2024. “Graf Bagian 3”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf. (Diakses 08 Januari 2025)
[10] Rifpanna, Arya. “Lulus arya rifpanna 09305141001.pdf”. https://eprints.uny.ac.id/28787/2/c.BAB%20II.pdf. (Diakses 08 Januari 2025)

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, 08 Januari 2025

Andi Farhan Hidayat
13523128