Penerapan Traveling Salesman Problem (TSP) untuk Optimalisasi Rute Perjalanan Jasa Kurir Pengantar Barang

Bernardus Willson - 13521021[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521021@std.stei.itb.ac.id

***ABSTRAK* — Tingkat kepadatan lalu lintas yang tinggi dapat menyebabkan kemacetan lalu lintas yang akan menyulitkan pengemudi untuk mencapai tujuan dengan alternatif jalur terpendek tepat waktu. Oleh karena itu, diperlukan rute terpendek dan teroptimal untuk efisiensi serta efektifitas waktu yang diperlukan manusia untuk melakukan mobilitas dari satu tempat ke tempat lain. Untuk mencari rute teroptimal dan terpendek tersebut tentunya diperlukan suatu algoritma yang cocok dan tepat agar rute yang ditemukan pun optimal dan akurat. Rute optimal yang dimaksud dapat dicari dengan menggunakan graf, lebih spesifik lagi menggunakan pendekatan *Traveling Salesman Problem* (TSP). Makalah ini membahas bagaimana pendekatan TSP ini dapat diterapkan di kehidupan sehari-hari, salah satunya adalah mencari rute paling optimal agar jasa kurir pengantar barang dapat mengirim barang-barang ke tempat tujuan dengan cepat dan efektif.**

***Kata Kunci* — graf, kurir, Traveling Salesman Problem (TSP), optimal.**

I. PENDAHULUAN

Kemajuan zaman dan kemajuan teknologi yang pesat berdampak pada kecenderungan manusia untuk melakukan mobilitas yang lebih tinggi. Mobilisasi ini tidak dapat berjalan tanpa adanya transportasi atau pembangunan infrastruktur jalan yang baik. Penataan kota yang kurang baik serta jumlah penduduk yang semakin bertambah menyebabkan masalah perhubungan semakin kompleks, maka dari itu dibutuhkan suatu metode yang tepat untuk menghindari kesalahan dalam menentukan keputusan yang disebabkan oleh kompleksitas dari masalah yang ada. Seiring dengan berkembangnya teknologi informasi, teknologi dalam bidang lain seperti teknologi transportasi dan infrastruktur juga turut berkembang. Berkembangnya teknologi transportasi dan infrastruktur ini merupakan tuntutan dari manusia yang memiliki kebutuhan untuk berpindah tempat dengan waktu sesingkat mungkin.

Mobilitas yang tinggi dapat memakan waktu yang cukup banyak jika tidak diperhitungkan dengan melewati rute terpendek. Masalah utamanya adalah bagaimana mencari lintasan terpendek dan teroptimal yang menghubungkan beberapa wilayah yang memiliki jarak yang berbeda-beda. Oleh karena itu, diperlukan rute optimal untuk efisiensi dan efektifitas waktu yang diperlukan manusia untuk melakukan mobilitas dari satu tempat ke tempat lain. Untuk mencari rute terpendek tersebut tentunya diperlukan suatu algoritma yang cocok dan tepat agar rute yang ditemukan pun optimal dan akurat. Rute efektif yang dimaksud dapat dicari dengan menggunakan graf.

Algoritma Graf merupakan algoritma pendekatan berbasis grafik untuk menganalisis data yang terhubung. Ada berbagai metode yang dapat digunakan, yaitu kueri data grafik, menggunakan dasar statistik, menjelajahi grafik secara visual, atau menggabungkan grafik ke dalam tugas pembelajaran mesin. Implementasi algoritma Graf dikatakan penting karena dapat digunakan untuk membantu memahami data yang terhubung. Menurut Amy E. Hodler, algoritma Graf secara unik dapat memahami struktur dan mengeluarkan pola dalam kumpulan data yang sangat terhubung sehingga dapat dengan mudah untuk memahami hubungan antara satu data dengan data yang lain.

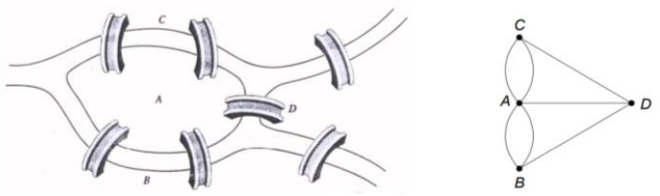
Peran algoritma Graf yang membantu dalam memahami hubungan data-data sangatlah membantu para engineer untuk membuat, mendesain, membangun, ataupun merencanakan proyek mereka karena algoritma Graf dapat mempercepat pembangunan atau perencanaan proyek, algoritma Graf dapat memperjelas struktur data yang digunakan sehingga akan sulit untuk mengalami kesalahan dalam proyek, dan juga jika terjadi kesalahan atau error pada proyek, kesalahan tersebut dapat ditinjau atau dicari menggunakan algoritma Graf yang telah dibuat sebelumnya sehingga tidak terjadi kesalahan pada bagian perbaikan. Singkatnya, algoritma Graf tidak akan membuat pada rekayasawan untuk melakukan tinjauan proyek dalam waktu yang lama karena sudah dibuat algoritma sebelum mulai pengerjaan proyek. Implementasi algoritma Graf dapat ditemukan dalam sistem penerbangan, sistem transportasi darat maupun laut, interaksi manusia, pemetaan, dan bidang lainnya.

II. LANDASAN TEORI

2.1 Graf

Graf dapat diartikan dan didefinisikan sebagai pasangan himpunan (V, E). V merupakan himpunan tidak kosong dari simpul-simpul atau *node*, V = {v_1, v_2, v_3, ...}. Sedangkan E merupakan himpunan sisi atau *edge* yang menghubungkan sepasang simpul. E = {e_1, e_2, e_3, ...}. Maka dari itu, persamaan dari graf ini dapat ditulis dengan notasi G = (V, E).

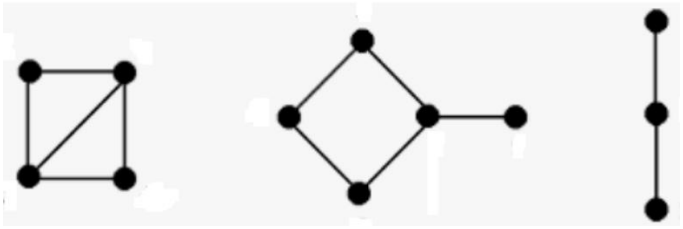
Sebuah graf mungkin untuk tidak mempunyai sisi satu buah pun, akan tetapi simpulnya harus ada minimal satu. Untuk penamaan simpul, simpul pada graf dapat diberi nama dengan huruf, seperti a, b, c, ..., z, atau dengan bilangan asli 1, 2, 3, ..., atau dengan gabungan keduanya. Sedangkan sisi dapat dilambangkan seperti ei, e2, ..., selain itu ketika sisi menghubungkan simpul v_i dengan simpul v_j, sisi tersebut dapat dinyatakan dengan pasangan (v_i, v_j). Maka dari itu, jika e adalah sisi yang menghubungkan simpul vi dengan simpul v_j, maka e dapat ditulis sebagai e = (v_i, v_j).



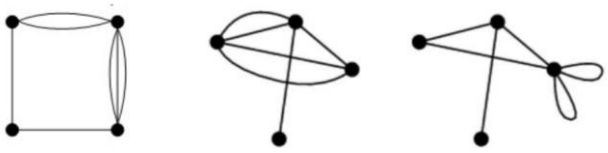
Jembatan Königsberg dan contoh graf, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

2.2 Jenis-jenis Graf

Graf dapat digolongkan menjadi beberapa jenis, yang pertama, berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, yaitu: Graf sederhana (*simple graph*) dimana graf tidak mengandung gelang maupun sisi-ganda; Graf tak-sederhana (*unsimple-graph*) dimana graf mengandung sisi ganda atau gelang.

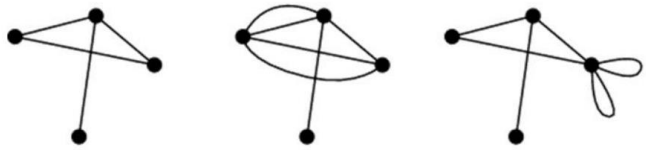


Graf Sederhana, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

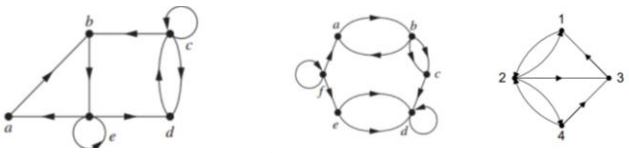


Graf Tak-Sederhana, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Selanjutnya, graf juga dapat dibedakan berdasarkan orientasi arah pada sisi, yaitu: Graf tak-berarah (*undirected graph*) dimana sisi-sisi pada graf tidak mempunyai orientasi arah; Graf berarah (*directed graph* atau *digraph*) dimana setiap sisi-sisi pada graf diberikan orientasi arah.



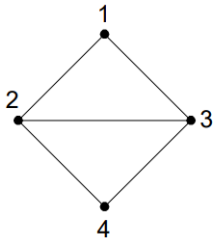
Graf Tak-Berarah, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf



Graf Berarah, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

2.3 Terminologi Graf

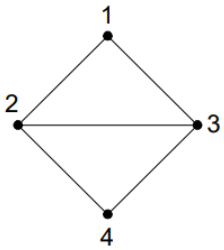
Terminologi graf dapat digolongkan menjadi beberapa jenis, yang pertama, Ketetanggaan (*Adjacent*) dimana dua buah simpul dikatakan bertetangga bila keduanya terhubung langsung.



G_1

Simpul 1 bertetangga dengan simpul 2 dan 3, namun simpul 1 tidak bertetangga dengan simpul 4, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

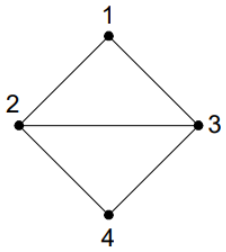
Selanjutnya adalah Beririsan (*Incidency*) dimana untuk sembarang sisi e = (v_j, v_k) dikatakan e bersisian dengan simpul vj , atau e bersisian dengan simpul v_k.



G_1

Sisi (2, 3) bersisian dengan simpul 2 dan simpul 3, sisi (2, 4) bersisian dengan simpul 2 dan simpul 4, tetapi sisi (1, 2) tidak bersisian dengan simpul 4, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

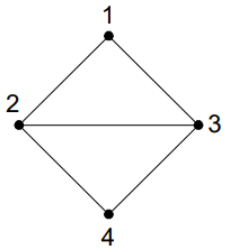
Ada juga Derajat (*Degree*) dimana derajat suatu simpul d(v) diartikan sebagai jumlah sisi yang bersisian dengan simpul tersebut.



G_1

d(1) = d(4) = 2, d(2) = d(3) = 3, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

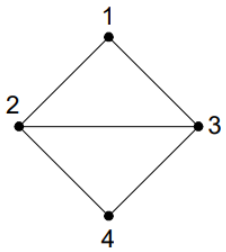
Selanjutnya yang tidak kalah penting adalah Lintasan (*Path*) dimana lintasan yang panjangnya n dari simpul awal ke simpul tujuan di dalam graf merupakan barisan berselang-seling simpul-simpul dan sisi-sisi. Bentuknya v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, v_n, sehingga $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, ..., $e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf. Sedangkan panjang lintasannya adalah jumlah sisi dalam lintasan tersebut.



G_1

Lintasan 1, 2, 4, 3 adalah lintasan dengan barisan sisi (1, 2), (2, 4), (4, 3), memiliki panjang lintasan 3, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

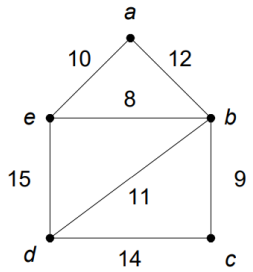
Selanjutnya adalah Siklus (*Cycle*) atau Sirkuit (*Circuit*) dimana lntasan berawal dan berakhir pada simpul yang sama.



G_1

1, 2, 3, 1 adalah sebuah sirkuit, memiliki panjang sirkuit 3, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Ada pula jenis-jenis graf lain, salah satunya adalah Graf Berbobot (*weighted graph*) yang biasa digunakan untuk mencari *shortest* path, dimana graf tersebut setiap sisinya diberi sebuah harga (bobot).

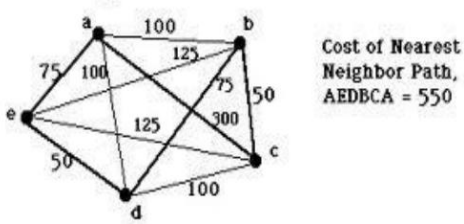


Graf Berbobot, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

2.4 Aplikasi Graf

Aplikasi graf di kehidupan sehari-hari cukup banyak seperti dalam jaringan internet, pengaturan jalan raya, pemodelan basis data, pencarian rute terpendek, dan masih banyak lagi. Namun pada makalah ini, penulis akan membahas tentang pencarian rute paling optimal untuk jasa kurir agar proses pengiriman dapat dilakukan dengan efektif. Pencarian rute optimal ini dapat dilakukan dengan pendekatan *Traveling Salesman Problem* (TSP).

Travelling Salesman Problem (TSP) adalah sebuah persoalan dalam teori graf dimana seorang pedagang keliling harus mengujungi sejumlah kota, dan harus dicari sirkuit terpendek jarak antarkota yang kemudian akan dilalui pedagang tersebut, dengan syarat pedagang itu harus berangkat dari kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal. Misal kota direpresentasikan sebagai simpul (*node*) dalam graf, jalan yang menghubungkan antarkota direpresentasikan sebagai sisi (*edge*) dari graf, dan bobot pada sisi menyatakan jarak antara dua buah kota. Persoalan ini dapat diselesaikan dengan menentukan sirkuit Hamilton (sirkuit yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal sekaligus simpul akhir yang dilalui dua kali) yang memiliki bobot paling kecil. Semua simpul graf memiliki sisi yang menghubungkan simpul tersebut dengan seluruh simpul lain pada graf, dengan kata lain graf lengkap. Untuk mencari jumlah sirkuit Hamiltonnya gunakan persamaan (n−1)!/2. Graf tidak lengkap sebenarnya juga memiliki sirkuit Hamilton, jadi persoalan TSP ini tidak hanya berlaku untuk graf lengkap saja. Untuk menyelesaikan permasalahan TSP dengan n sembarang, sampai saat ini belum ada algoritma yang dapat menyelesaikannya.



Traveling Salesman Problem, sumber:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian3.pdf

III. METODOLOGI

Pada makalah ini, penulis akan mencoba menyelesaikan masalah TSP dengan menggunakan metode Heuristik karena metode ini dapat diterapkan dengan relatif cepat untuk menemukan solusi, walaupun belum tentu solusinya optimal. Namun di saat pendekatan secara eksak tidak praktis untuk diterapkan, metode Heuristik dapat menyelesaikan masalah dengan cara *trial and error*.

Terdapat beberapa cara penyelesaian TSP dengan metode Heuristik, salah satunya adalah *Nearest-Neighbor Heuristic*. Langkah-langkahnya adalah sebagai berikut:

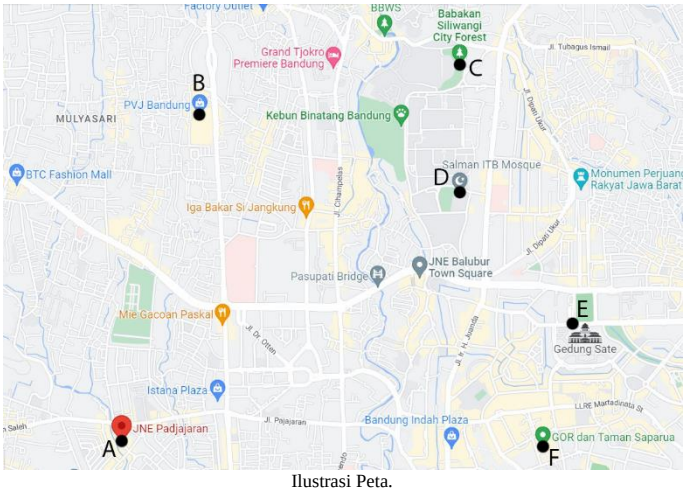
1. Mulai dari sebarang kota, dan kunjungi kota terdekat
2. Pergi ke kota yang belum dikunjungi, yang terdekat dengan kota terakhir yang saat ini dikunjungi
3. Lanjutkan langkah ini sampai sebuah *tour* didapatkan
4. Ulangi langkah-langkah di atas, dimulai dengan kota yang berbeda
5. Lalu pilih *tour* terbaik.

Selain itu, penyelesaian dengan metode Heuristik dapat dilakukan dengan cara *Cheapest-Insertion Heuristic*. Langkah-langkahnya adalah sebagai berikut:

1. Mulai dari sebarang kota, dan kunjungi kota terdekat
2. Buat *subtour* yang menghubungkan kedua kota
3. Ganti sebuah *arc* (*edge* atau sisi) di subtour tersebut (katakanlah *arc* (i, j) dengan kombinasi dari 2 *arcs*, yaitu (i, k) dan (k, i), dengan k tidak berada dalam *subtour* saat ini, yang akan meningkatkan panjang *subtour* dengan nilai yang terkecil. Jika C_{ij} adalah panjang *arc* (i, j), perhatikan bahwa jika *arc* (i, j) digantikan oleh (i, k) dan (k, j), maka panjang $C_{ik} + C_{kj} - C_{ij}$ ditambahkan ke *subtour*
4. Lanjutkan langkah ini sampai sebuah tour kita dapatkan

V. IMPLEMENTASI

Berikut ini merupakan contoh simulasi penerapan TSP untuk mencari rute paling optimal untuk pengiriman barang ke beberapa tujuan sekaligus.



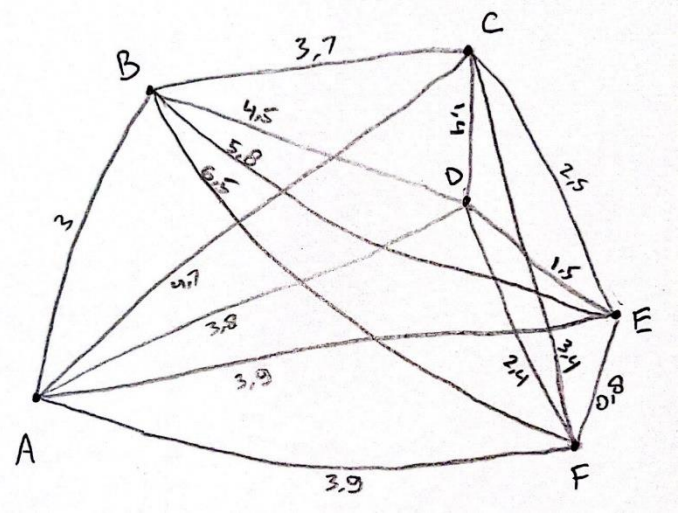
Ilustrasi Peta.

Pada kasus kali ini, titik start berada di JNE Padjajaran (A), setelah pengiriman barang selesai, kurir harus kembali ke titik asal. Kurir harus mengirimkan barang ke beberapa tempat: PVJ Bandung (B), Babakan Siliwangi City Forest (C), Salman ITB Mosque (D), Gedung Sate (E), GOR dan Taman Saparua (F).

Selanjutnya, diperlukan informasi jarak antar titik. Berdasarkan hasil pengecekan jarak menggunakan Google Maps, jarak antar titik adalah sebagai berikut:

	A	B	C	D	E	F
A	0	3	4.7	3.8	3.9	3.9
B	3	0	3.7	4.5	5.8	6.5
C	4.7	3.7	0	1.4	2.5	3.4
D	3.8	4.5	1.4	0	1.5	2.4
E	3.9	5.8	2.5	1.5	0	0.8
F	3.9	6.5	3.4	2.4	0.8	0

Tabel jarak antar titik dalam Kilometer.

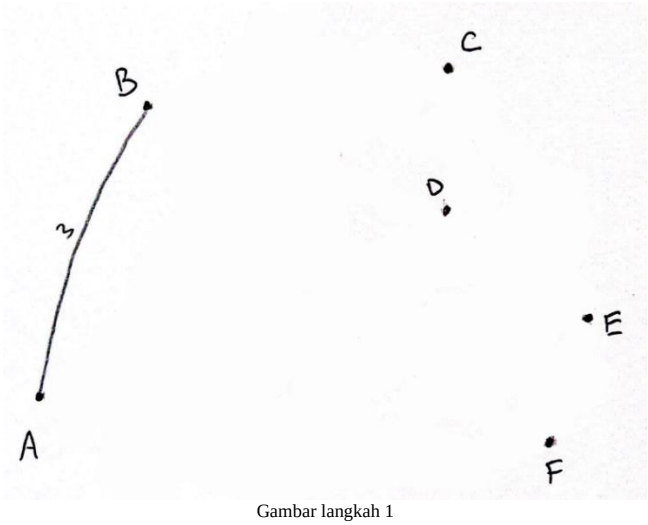


Ilustrasi Graf.

5.1 Nearest-Neighbor Heuristic

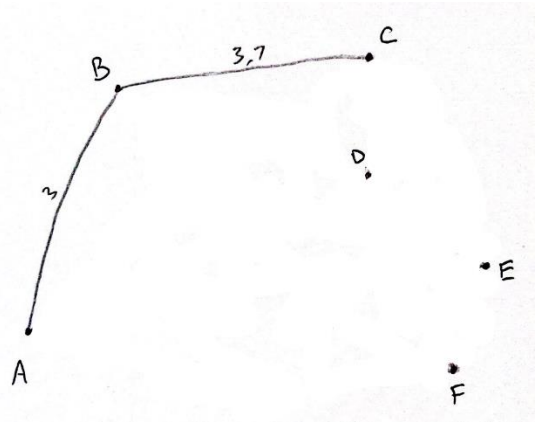
Berikut ini merupakan contoh perhitungan rute paling optimal menggunakan pendekatan *Nearest-Neighbor Heuristic.*

1. *Node* awal berada di A, maka kita harus mencari *node* dengan jarak terdekat dengan A, yaitu B sejauh 3 kilometer.



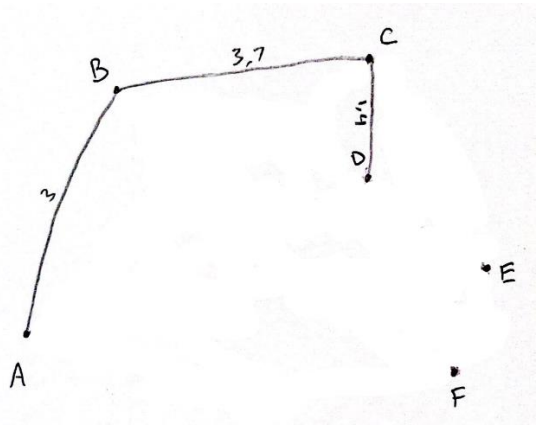
Gambar langkah 1

2. *Node* awal berada di B, maka kita harus mencari *node* dengan jarak terdekat dengan B, yaitu A. Namun *node* A sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* C sejauh 3.7 kilometer.



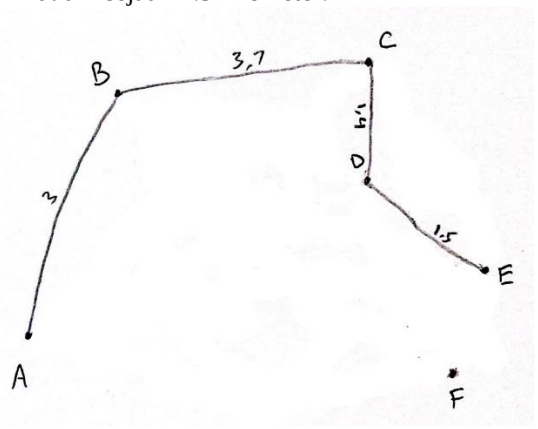
Gambar langkah 2

3. *Node* awal berada di C, maka kita harus mencari *node* dengan jarak terdekat dengan C, yaitu *node* D sejauh 1.4 kilometer.



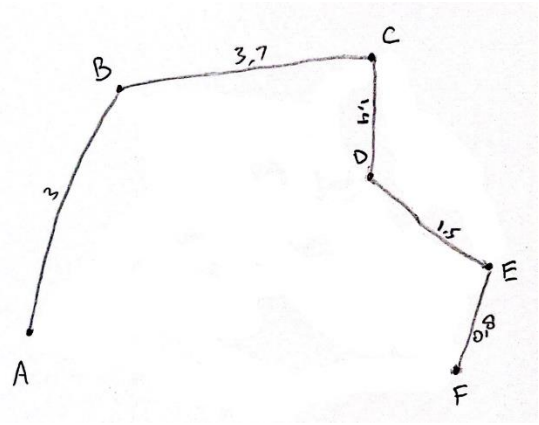
Gambar langkah 3

4. *Node* awal berada di D, maka kita harus mencari *node* dengan jarak terdekat dengan D, yaitu C. Namun *node* C sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* E sejauh 1.5 kilometer.



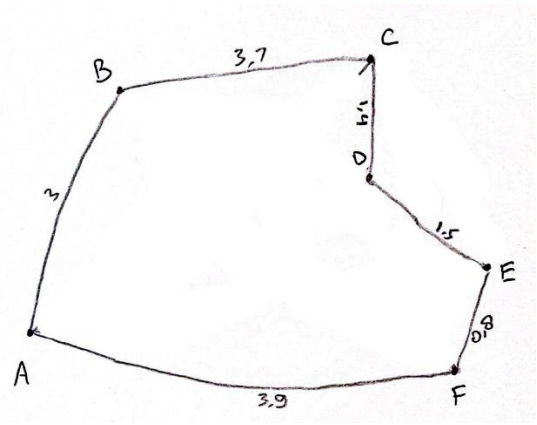
Gambar langkah 4

5. *Node* awal berada di E, maka kita harus mencari *node* dengan jarak terdekat dengan E, yaitu *node* F sejauh 0.8 kilometer.



Gambar langkah 5

6. *Node* awal berada di F, maka kita harus mencari *node* dengan jarak terdekat dengan F, yaitu E. Namun *node* E sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* C. Namun *node* C sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* D. Namun *node* D sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* C. Namun *node* C sudah pernah dikunjungi, maka pilih *node* lain yaitu *node* A sejauh 3.9 kilometer.



Gambar langkah 6

7. Maka dari itu, diperoleh solusi dari permasalahan pencarian rute paling optimal untuk kurir mengantar barang-barangnya. Menurut hasil yang diperoleh, rute paling optimal adalah mulai dari JNE Padjadjaran (A) -> PVJ Bandung (B) -> Babakan Siliwangi City Forest (C) -> Salman ITB Mosque (D) -> Gedung Sate (E) -> GOR dan Taman Saparua (F) -> JNE Padjadjaran (A). Dengan total jarak adalah 3 + 3.7 + 1.4 + 1.5 + 0.8 + 3.9 = 14.3 km. Jika rute dibalik juga tidak masalah.

5.2 Nearest-Neighbor Heuristic

Berikut ini merupakan contoh perhitungan rute paling optimal menggunakan pendekatan *Cheapest-Insertion Heuristic.*

1. (A, B) – (B, A) {belum dikunjungi C, D, E, F}

Arc yg diganti	Arc yg ditambahkan	Panjang yg ditambah
(A, B)	(A, C) – (C, B)	$C_{AC} + C_{CB} – C_{AB} = 5.4$
(A, B)	(A, D) – (D, B)	**5.3**
(A, B)	(A, E) – (E, B)	6.7
(A, B)	(A, F) – (F, B)	7.4
(B, A)	(B, C) – (C, A)	5.4
(B, A)	(B, D) – (D, A)	**5.3**
(B, A)	(B, E) – (E, A)	6.7
(B, A)	(B, F) – (F, A)	7.4

Langkah pertama adalah menentukan node terdekat dari *node* awal (A) yaitu *node* B. Lakukan perhitungan-perhitungan seperti langkah di atas, lalu cari "Panjang yg ditambah" terkecil. Pada tabel di atas, yang terkecil adalah (A, D) – (D, B) dengan "Panjang yg ditambah" sebesar 5.3. Maka dari itu, rute diganti menjadi (A, D) – (D, B) – (B, A).

2. (A, D) – (D, B) – (B, A) {belum dikunjungi C, E, F}

Arc yg diganti	Arc yg ditambahkan	Panjang yg ditambah
(A, D)	(A, C) – (C, D)	2.3
(A, D)	(A, E) – (E, D)	1.6
(A, D)	(A, F) – (F, D)	2.5
(D, B)	(D, C) – (C, B)	**0.6**
(D, B)	(D, E) – (E, B)	2.8
(D, B)	(D, F) – (F, B)	4.4
(B, A)	(B, C) – (C, A)	5.4
(B, A)	(B, E) – (E, A)	6.7
(B, A)	(B, F) – (F, A)	7.4

Lakukan hal yang sama seperti langkah sebelumnya. Pada tabel di atas, yang terkecil adalah (D, C) – (C, B) dengan "Panjang yg ditambah" sebesar 0.6. Maka dari itu, rute diganti menjadi (A, D) – (D, C) – (C, B) – (B, A).

3. (A, D) – (D, C) – (C, B) – (B, A) {belum dikunjungi E, F}

Arc yg diganti	Arc yg ditambahkan	Panjang yg ditambah
(A, D)	(A, E) – (E, D)	**1.6**
(A, D)	(A, F) – (F, D)	2.5
(D, C)	(D, E) – (E, C)	2.6
(D, C)	(D, F) – (F, C)	4.4
(C, B)	(C, E) – (E, B)	4.6
(C, B)	(C, F) – (F, B)	6.2
(B, A)	(B, E) – (E, A)	6.7
(B, A)	(B, F) – (F, A)	7.4

Lakukan hal yang sama seperti langkah sebelumnya. Pada tabel di atas, yang terkecil adalah (A, E) – (E, D) dengan "Panjang yg ditambah" sebesar 1.6. Maka dari itu, rute diganti menjadi (A, E) – (E, D) – (D, C) – (C, B) – (B, A).

4. (A, E) – (E, D) – (D, C) – (C, B) – (B, A) {belum dikunjungi F}

Arc yg diganti	Arc yg ditambahkan	Panjang yg ditambah
(A, E)	(A, F) – (F, E)	**0.8**
(E, D)	(E, F) – (F, D)	1.7
(D, C)	(D, F) – (F, C)	4.4
(C, B)	(C, F) – (F, B)	6.2
(B, A)	(B, F) – (F, A)	7.4

Lakukan hal yang sama seperti langkah sebelumnya. Pada tabel di atas, yang terkecil adalah (A, F) – (F, E) dengan "Panjang yg ditambah" sebesar 0.8. Maka dari itu, rute diganti menjadi (A, F) – (F, E) – (E, D) – (D, C) – (C, B) – (B, A).

5. Maka dari itu, diperoleh solusi dari permasalahan pencarian rute paling optimal untuk kurir mengantar barang-barangnya. Menurut hasil yang diperoleh, rute paling optimal adalah mulai dari JNE Padjadjaran (A) -> GOR dan Taman Saparua (F) -> Gedung Sate (E) -> Salman ITB Mosque (D) -> City Forest (C) -> PVJ Bandung (B). Dengan total jarak adalah 3.9 + 0.8 + 1.5 + 1.4 + 3.7 + 3 = 14.3 km. Jika rute dibalik juga tidak masalah.

V. KESIMPULAN

Solusi yang dapat memudahkan pencarian rute paling optimal adalah dengan menggunakan berbagai macam pendekatan graf, salah satunya adalah menggunakan konsep *Traveling Salesman Problem* (TSP). Setelah dilakukan pencarian solusi menggunakan 2 metode, hasil yang diperoleh adalah sama. Kesimpulannya kedua metode dapat dipakai untuk mencari solusi rute paling optimal. Algoritma atau perhitungan dapat diimplementasikan pada aplikasi jasa kurir seperti JNE,

GoSend, AnterAja, dan masih banyak lagi, sebagai pencari rute paling optimal dengan waktu yang relatif singkat dan biaya yang lebih hemat sehingga menjadi solusi pekerjaan dengan mobilitas tinggi, dengan asumsi tidak ada kemacetan, dengan kata lain faktor kemacetan tidak diperhitungkan.

VI. PENUTUP

Segala puji syukur dan terima kasih penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya makalah ini dapat selesai. Ucapan terima kasih penulis sampaikan kepada orang tua dan keluarga, serta seluruh kolega yang telah mendukung penulis dari segi moral maupun material. Tidak lupa penulis juga mengucapkan terima kasih kepada seluruh Bapak dan Ibu dosen pengampu mata kuliah Matematika Diskrit Institut Teknologi Bandung yang senantiasa menjadi pembimbing dalam mengajarkan dan menurunkan ilmu – ilmu matematika diskrit kepada penulis sehingga makalah berjudul "Penerapan Traveling Salesman Problem (TSP) untuk Optimalisasi Rute Perjalanan Jasa Kurir Pengantar Barang" dapat selesai dengan baik dan tepat waktu. Penulis juga ingin meminta maaf jika makalah ini belum sempurna dan masih ada salah kata dalam penulisan makalah. Kiranya makalah ini dapat menjadi sumber pembelajaran dan dapat bermanfaat bagi orang banyak, terutama yang ingin mempelajari hal-hal tentang graf dan Matematika Diskrit.

REFERENSI

[1] Needham dan Hodler, "Graph Algorithms". 1st ed. United State of America: y O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2019.

[2] Munir, Rinaldi (2003). Graf (Bag.1) Bahan Kuliah IF2120 Matematika Diskrit, URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf
Diakses: 3 Desember 2022, pukul 1:51.

[3] Munir, Rinaldi (2003). Graf (Bag.3) Bahan Kuliah IF2120 Matematika Diskrit, URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian3.pdf
Diakses: 3 Desember 2022, pukul 23:55.

[4] Rachmawati, Ramya (2020). Metode Heuristik untuk Menyelesaikan Masalah TSP, URL: https://youtu.be/dHseCyY5g2U
Diakses: 4 Desember 2022, pukul 0:55.

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 Desember 2022

Bernardus Willson 13521021