Pendekatan Heuristik untuk Masalah Kompleks di Mini Metro

Muhammad Iqbal Raihan - 13524011
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknolog Bandung, Jalan ganesha 10 Bandung
E-mail: miqbalr001@gmail.com, 13524011@std.stei.itb.ac.id

***Abstrak*—Makalah ini menganalisis strategi heuristik dalam permainan Mini Metro, yang dimodelkan sebagai masalah optimisasi NP-hard. Beberapa heuristik penyusunan jalur dibentuk dan dibandingkan dengan algoritma komputasional (Nearest Neighbor dan 2-opt). Hasil menunjukkan bahwa heuristik pemain lebih unggul karena keterbatasan algoritma komputasional yang digunakan, tetapi algoritma tersebut tetap lebih unggul dibandingkan dengan heuristik permainan yang sebanding.**

***Kata Kunci*—Teori Graf, Heuristik, Optimisasi Jaringan, Vehicle Routing Problem, Nearest Neighbor, 2-opt Algorithm**

I. Pendahuluan

A. Motivasi

Perencanaan jalur *metro* atau Moda Raya Terpadu (MRT) adalah tantangan yang sangat kompleks. Seiiring dengan bertambahnya stasiun, banyaknyak konfigurasi jalur yang mungkin terbenbentuk akan meningkat dalam skala faktorial. Heuristik diperlukan untuk mendapatkan konfigurasi jalur yang cukup baik tanpa memakan terlalu banyak waktu kalkulasi.

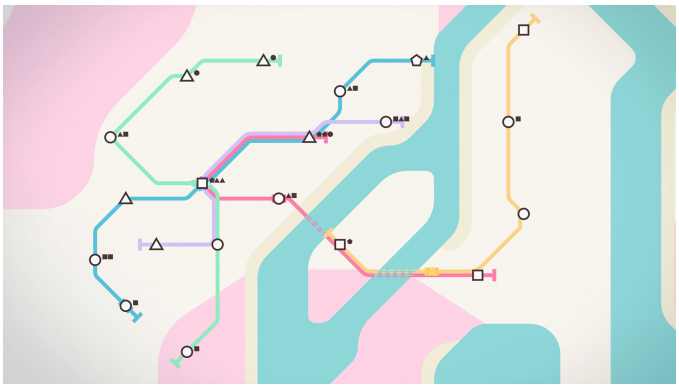
B. Konteks Masalah

Mini Metro adalah permainan sederhana dimana pemain diminta untuk menghubungkan sekumpulan stasiun berupa simpul atau *vertex* dengan jalur berupa sisi atau *edge*. Dalam setiap jalur, ada satu atau lebih kereta yang akan mengangkut penumpang dari satu stasiun ke stasiun tujuan masing-masing penumpang. Seiiring berjalannya waktu, stasiun akan muncul satu persatu.

Stasiun dibagi menjadi dua jenis, biasa, yaitu stasiun yang sering muncul, dan spesial, yaitu stasiun yang muncul hanya sekali. Jenis-jenis stasiun biasa ada tiga, urut dari yang paling umum sampai paling jarang muncul, yaitu *circle* (C), *triangle* (T), dan *square* (S). Stasiun spesial memiliki banyak jenis, seperti *plus*, *football*, *pentagon*, dan lain-lain. Perbedaan dari masing-masing jenis hanyalah simbol dari masing-masing *vertex*.

Setiap gerbong kereta memiliki kapasitas enam penumpang dan jumlah gerbong kereta yang bisa digunakan pemain terbatas. Pemain dapat mengatur jalur kereta dengan bebas setiap saat dengan batasan banyak jalur dan banyak jembatan atau terowongan yang digunakan untuk melintasi sungai. Setiap hari Minggu di dalam permainan, pemain dapat memilih sumber daya baru untuk digunakan, seperti jalur kereta baru atau gerbong kereta baru.

Permainan berakhir ketika ada stasiun yang ”terlalu ramai”, yaitu stasiun yang memiliki penumpang mengantri di luar batas kapasitas stasiun tersebut selama selang waktu tertentu.

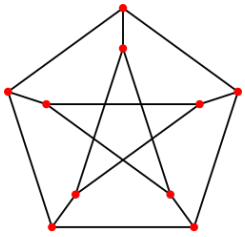


Gambar 1: antarmuka dalam permainan Mini Metro
Sumber: [1]

II. Landasan Teori

A. Teori Graf

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf G dapat dinotasikan sebagai $G = (V, E)$, dengan $V = v_1, v_2, v_3, \ldots$ merupakan himpunan simpul atau *vertex* yang ada dan $E = e_1, e_2, e_3, \ldots$ merupakan himpunan sisi atau *edge* yang menghubungkan satu *vertex* dengan *vertex* lain.



Gambar 2: Contoh graf
Sumber: [2]

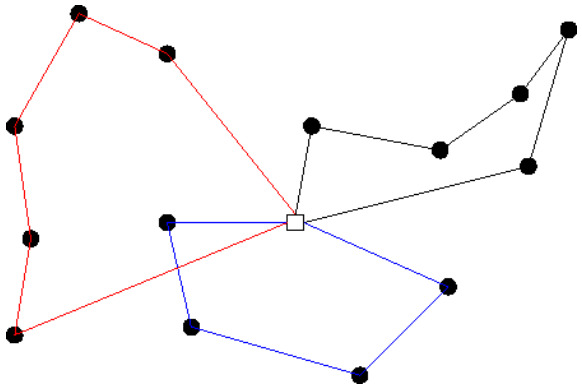
Berdasarkan ada atau tidaknya kalang dan sisi ganda, graf dapat dibagi menjadi dua jenis: sederhana dan tidak sederhana. Graf sederhana adalah graf yang tidak memiliki kalang atau

loop serta tidak memiliki sisi ganda, sedangkan graf tidak sederhana memiliki salah satu atau keduanya.

Berdasarkan orientasi arah, graf dapat dibagi menjadi dua jenis: berarah dan tidak berarah. Semua sisi dari graf berarah memiliki arah, sedangkan semua sisi dari graf tidak berarah tidak memiliki orientasi arah tertentu.

B. Vehicle Routing Problem

Permasalahan Penentuan Rute Kendaraan, atau dalam Bahasa Inggris dikenal sebagai Vehicle Routing Problem (VRP), merupakan salah satu permasalahan optimisasi kombinatorial dalam bidang logistik dan transportasi. VRP pertama kali diperkenalkan oleh Dantzig dan Ramser pada tahun 1959 sebagai generalisasi dari Travelling Salesman Problem (TSP) [3], namun dengan kompleksitas yang lebih tinggi karena melibatkan lebih dari satu kendaraan dan berbagai kendala operasional.



Gambar 3: contoh solusi Vehicle Routing Problem pada suatu graf
Sumber: [4]

VRP adalah masalah optimisasi tentang bagaimana suatu barang dikirimkan dari satu atau lebih *depot* yang memiliki beberapa kendaraan dengan beberapa sopir ke sekumpulan pelanggan. Solusi yang diharapkan adalah satu set rute pengiriman barang sedemikian sehingga semua keinginan konsumen dan batasan operasional terpenuhi dengan biaya pengiriman diminimalisasi. Biaya yang dimaksud dapat berupa uang, jarak, atau yang lainnya.

C. Algoritma Nearest Neighbor

Algoritma *Nearest Neighbor* adalah algoritma untuk mencari solusi aproksimasi dari *Traveling Salesman Problem* (TSP). Algoritma ini dapat menghasilkan solusi dengan cepat, tetapi tidak optimal.

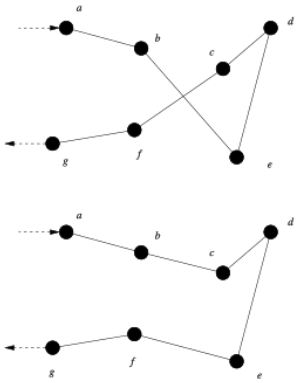
Berikut adalah algoritma *Nearest Neighbor*:

1) Tandai semua simpul sebagai belum dikunjungi.
2) Pilih simpul sembarang, misal u, lalu tandai sebagai sudah dikunjungi.
3) Pilih sisi terpendek yang menghubungkan u ke simpul yang belum dikunjungi, misal v.
4) Atur v sebagai simpul saat ini. Tandai v sebagai sudah dikunjungi.
5) Jika semua simpul telah dikunjungi, maka algoritma selesai. Jika tidak, lanjutkan ke langkah 3.

Dalam kasus terburuk, algoritma ini menghasilkan solusi yang jauh lebih buruk dari solusi optimal. Lebih tepatnya, untuk setiap konstan r, terdapat sebuah konfigurasi TSP sedemikian sehingga panjang dari perjalanan yang terkalkulasi oleh algoritma *Nearest Neighbor* lebih besar dari r dikali panjang perjalanan yang optimal. Lebih lanjut, untuk setiap banyaknya *vertex*, terdapat konfigurasi jarak antar *vertex* sedemikian sehingga algoritma *Nearest Neighbor* menghasilkan solusi unik terburuk. (Jika algoritma ini diterapkan untuk setiap *vertex* sebagai *vertex* awal, lintasan terbaik yang didapat akan lebih baik dari setidaknya $\frac{N}{2} - 1$ lintasan lain, dengan N adalah banyaknya *vertex*) [5].

D. Algoritma 2-opt

Algoritma *2-opt* adalah algoritma pencarian lokal sederhana untuk menyelesaikan *Traveling Salesman Problem* (TSP). Algoritma *2-opt* pertama kali diusulkan oleh Croes pada 1958 [6], walaupun langkah dasarnya sudah disarankan oleh Flood [7].



Gambar 4: Algoritma 2-opt
sumber: [8]

Ide utama dari algoritma *2-opt* adalah menghapus dua sisi yang saling berpotongan dan menggantinya dengan dua sisi baru yang tidak saling berpotongan. Teknik ini dapat diaplikasikan ke *Traveling Salesman Problem* (TSP), *Vehicle Routing Problem* (VRP), dan lain-lain.

Versi lebih kompleks dari algoritma ini adalah algoritma 3-opt.

III. Model Sistem dan Formulasi Masalah

A. Model Graf

Dunia game dapat digambarkan sebagai graf dinamis $G = (V, E)$. Himpunan *vertex* V melambangkan kumpulan stasiun dengan sifat-sifat seperti bentuk, kapasitas, dan antrian penumpang. Himpunan *edge* E melambangkan kumpulan segmen jalur kereta. Jalur kereta digambarkan sebagai *disjoint path* atau sirkuit dalam graf.

B. Analisis Kompleksitas Komputasi

NP-hard adalah kelas masalah yang tiap masalahnya setidaknya sama kompleksnya dengan masalah terkompleks di *NP*. *NP* adalah kelas masalah yang tiap solusi dari masalah tersebut dapat diverifikasi dalam kompleksitas polinomial.

Masalah optimisasi Mini Metro dapat dianalogikan dengan *Vehicle Routing Problem* (VRP), suatu masalah *NP-hard*. Setiap kereta dapat disamakan dengan kendaraan, stasiun spesial dengan *depot*, stasiun dengan pelanggan, dan destinasi penumpang dengan permintaan rute.

Karena masalah Mini Metro masuk dalam klasifikasi *NP-hard*, mencari solusi dalam *real-time* tidak mungkin, sehingga diperlukan adanya heuristik.

C. Parameter dan Asumsi

Asumsi-asumsi yang akan digunakan dalam makalah ini adalah sebagai berikut.

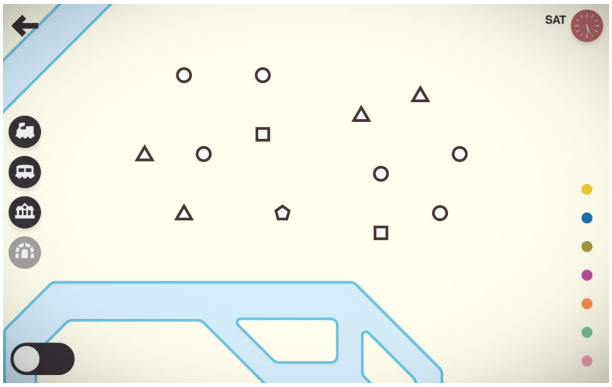
- Kereta selalu bergerak dalam kecepatan tetap.
- Kereta akan selalu berhenti di setiap stasiun dengan waktu henti yang sama.

Parameter-parameter yang akan digunakan dalam makalah ini adalah sebagai berikut.

- $d(v_i, v_{i+1})$ sebagai jarak antara stasiun v_i dengan v_{i+1}.
- N_{train} sebagai total banyaknya kereta yang ada.
- $s = 1$ (unit/detik) sebagai kecepatan konstan kereta.
- $t_{dwell} = 2$ s sebagai waktu konstan suatu kereta berhenti di setiap stasiun.
- C_{train} sebagai kapasitas tiap kereta.
- D_{total} sebagai total panjang suatu jalur.

D. Pengukuran Performa

Tes untuk pengukuran performa akan menggunakan konfigurasi berikut, dengan $N_{train} = 2$ dan $C_{train} = 6$ tanpa gerbong tambahan.



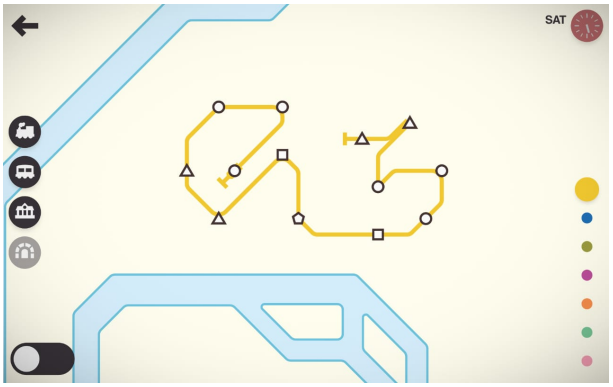
Gambar 5: Konfigurasi awal Mini Metro

Berikut adalah metrik yang akan diukur.

- Waktu Siklus Jalur (T_{cycle}): Waktu total yang diperlukan suatu kereta untuk menyelesaikan satu siklus perjalanan penuh dan kembali lagi ke titik awal. Untuk suatu jalur kereta dengan n stasiun,
  - Untuk jalur non-sirkuit:
$$T_{cycle} = \sum_{i=1}^{n-1} \frac{2 \cdot d(v_i, v_{i+1})}{s} + 2n \cdot t_{dwell}$$
  - Untuk jalur sirkuit:
$$T_{cycle} = \sum_{i=1}^{n-1} \frac{d(v_i, v_{i+1})}{s} + \frac{d(v_n, v_1)}{s} + n \cdot t_{dwell}$$
- Kapasitas Jalur (Φ): Banyak maksimal penumpang yang dapat ditampung suatu jalur kereta per unit waktu. Untuk suatu jalur dengan N_t kereta dengan kapasitas C_{train}:
$$\Phi = \frac{N_t \cdot C_{train}}{T_{cycle}} \quad \text{(penumpang/detik)}$$
- *Shape Diversity Index* (I_{SD}): Indeks kualitatif untuk menentukan seberapa baik suatu jalur kereta dalam menghubungkan stasiun secara berselang-seling jenis stasiunnya.
$$I_{SD} = \frac{\text{banyak koneksi antar stasiun dengan jenis berbeda}}{\text{total koneksi antar stasiun}}$$

IV. Analisis dan Kalkulasi

Berikut adalah konfigurasi jalur kereta awal:



Gambar 6: Konstruksi jalur kereta tanpa heuristik

Dengan konfigurasi tersebut, didapat:

$$D_{total} = 25 + 18\sqrt{2} \approx 50.45 \quad (1)$$

$$T_{cycle} = 2 \times \frac{25 + 18\sqrt{2}}{1} + 2 \cdot 2 \cdot 13 = 102 + 36\sqrt{2} \text{ s} \quad (2)$$

$$\Phi = \frac{2 \times 6}{102 + 36\sqrt{2}} \approx 0.0785 \text{ penumpang/s} \quad (3)$$

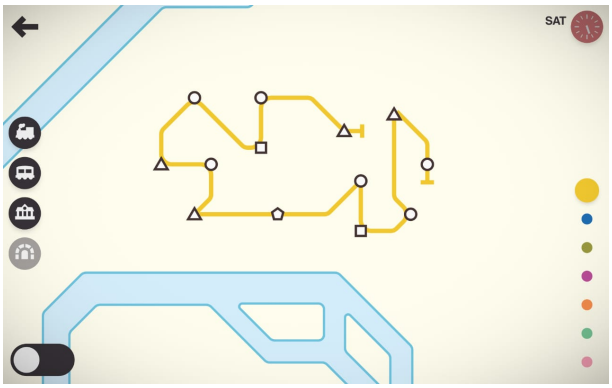
$$I_{SD} = \frac{6}{12} = 0.5 \quad (4)$$

A. Heuristik 1: Hubungkan Stasiun Secara Bergantian dalam Jenis

Hubungkan stasiun sebisa mungkin secara bergantian, misal S-T-C-T-C-C-S (*loop*). Jika terdapat tiga stasiun berjenis sama yang terhubung secara berurutan, misal T-C-C-C-S, maka gerbong kereta akan penuh dan tidak dapat menampung penumpang lagi ketika sampai di stasiun C ketiga, karena penumpang dari suatu stasiun tidak akan turun di stasiun

dengan jenis yang sama. Dengan kata lain, penumpang dari suatu stasiun X tidak akan memiliki tujuan ke stasiun X.

Berikut adalah konfigurasi jalur kereta dengan heuristik 1:



Gambar 7: Konstruksi jalur kereta dengan heuristik 1

Dengan konfigurasi tersebut, didapat:

$$D_{total} = 33 + 14\sqrt{2} \approx 52.80 \tag{5}$$

$$T_{cycle} = 2 \times \frac{33 + 14\sqrt{2}}{1} + 2 \cdot 2 \cdot 13 = 118 + 28\sqrt{2} \quad \text{s} \tag{6}$$

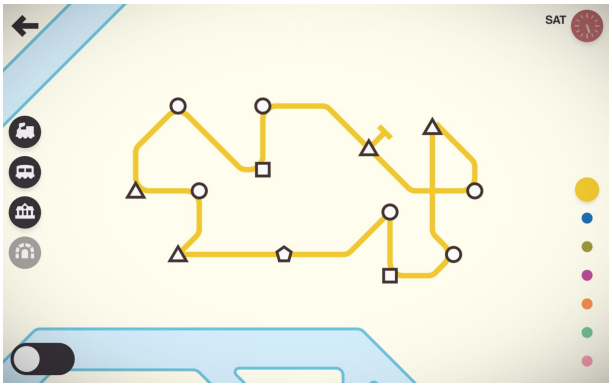
$$\Phi = \frac{2 \times 6}{118 + 28\sqrt{2}} \approx 0.0761 \quad \text{penumpang/s} \tag{7}$$

$$I_{SD} = \frac{12}{12} = 1 \tag{8}$$

B. Heuristik 2: Buat Jalur Kereta Menjadi Sirkuit

Jalur kereta berbentuk sirkuit akan meratakan waktu kedatangan kereta di tiap stasiun. Jalur yang tidak berbentuk sirkuit, misal C-C-T-T-C-S, akan memiliki jadwal kedatangan kereta yang tidak merata. Kereta akan jarang mengunjungi stasiun di ujung lintasan dan sering mengunjungi stasiun di tengah lintasan.

Berikut adalah konfigurasi jalur kereta dengan heuristik 2:



Gambar 8: Konstruksi jalur kereta dengan heuristik 2

Dengan konfigurasi tersebut, didapat:

$$D_{total} = 36 + 16\sqrt{2} \approx 58.63 \tag{9}$$

$$T_{cycle} = \frac{36 + 16\sqrt{2}}{1} + 2 \cdot 13 = 62 + 16\sqrt{2} \quad \text{s} \tag{10}$$

$$\Phi = \frac{2 \times 6}{62 + 16\sqrt{2}} \approx 0.1418 \quad \text{penumpang/s} \tag{11}$$

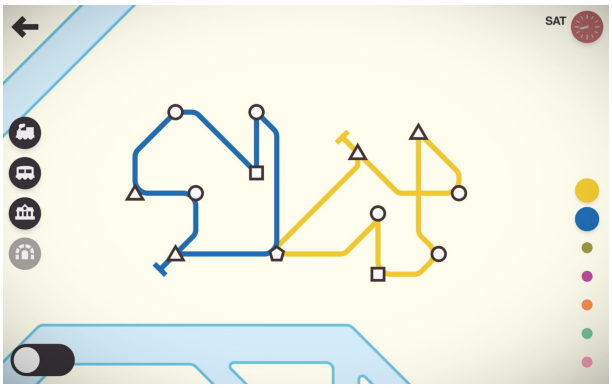
$$I_{SD} = \frac{13}{13} = 1 \tag{12}$$

C. Heuristik 3: Gunakan Stasiun Spesial Sebagai Stasiun Pusat

Gunakan stasiun spesial sebagai stasiun pusat, daripada stasiun biasa seperti C atau T. Penumpang akan melakukan transfer jalur kereta ketika stasiun tujuannya tidak ada di jalur saat ini. Semakin sering penumpang membutuhkan transfer jalur, semakin tidak efisien jaringan kereta yang ada.

Menjadikan stasiun spesial sebagai stasiun pusat akan meminimalisir terjadinya transfer jalur karena semua stasiun ada di tiap jalur, termasuk stasiun spesial itu sendiri. Menjadikan stasiun biasa sebagai stasiun pusat akan membuat banyak transfer terjadi karena stasiun spesial tidak terdapat di semua jalur.

Berikut adalah konfigurasi jalur kereta dengan heuristik 3:



Gambar 9: Konstruksi jalur kereta dengan heuristik 3

Dengan konfigurasi tersebut, pada jalur biru didapat:

$$D_{total} = 22 + 7\sqrt{2} \approx 31.90 \tag{13}$$

$$T_{cycle} = \frac{22 + 7\sqrt{2}}{1} + 2 \cdot 7 = 36 + 7\sqrt{2} \quad \text{s} \tag{14}$$

$$\Phi = \frac{1 \times 6}{36 + 7\sqrt{2}} \approx 0.1307 \quad \text{penumpang/s} \tag{15}$$

$$I_{SD} = \frac{7}{7} = 1 \tag{16}$$

Dengan konfigurasi tersebut, pada jalur kuning didapat:

$$D_{total} = 18 + 12\sqrt{2} \approx 34.97 \tag{17}$$

$$T_{cycle} = \frac{18 + 12\sqrt{2}}{1} + 2 \cdot 7 = 32 + 12\sqrt{2} \quad \text{s} \tag{18}$$

$$\Phi = \frac{1 \times 6}{32 + 12\sqrt{2}} \approx 0.1225 \quad \text{penumpang/s} \tag{19}$$

$$I_{SD} = \frac{7}{7} = 1 \tag{20}$$

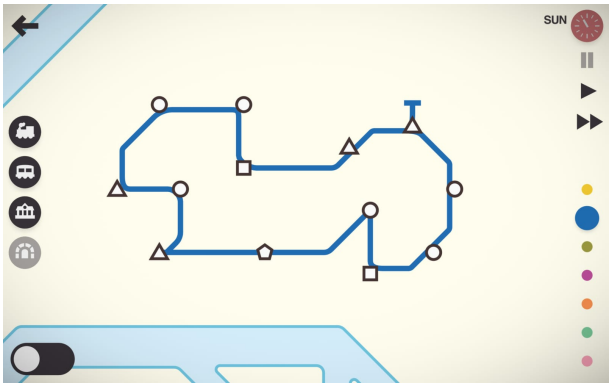
D. Algoritma Nearest Neighbor dan 2-opt

Digunakan algoritma *Nearest Neighbor* dan *2-opt* untuk membuat lintasan baru, berupa sebuah jalur kereta dengan dua kereta. Untuk simplisitas, algoritma tidak menilai jenis stasiun. Berikut adalah langkah membentuk sirkuit lintasan kereta:

1) Dibuat graf lengkap berbobot dengan masing-masing stasiun sebagai *vertex* dan jarak antar stasiun sebagai bobot dari tiap sisi yang bersesuaian.
2) Pilih stasiun acak
3) Mulai dari stasiun saat ini, bentuk sirkuit dengan algoritma *Nearest Neighbor* dan *2-opt*. Catat hasilnya.
4) Pilih stasiun lain, lalu ulangi langkah 3 sampai semua stasiun sudah pernah terpilih.
5) Ambil hasil terbaik dari seluruh hasil yang sudah tercatat.

Algoritma diatas berjalan dengan kompleksitas $O(N^3)$.

Berikut adalah konfigurasi jalur kereta dengan algoritma *Nearest Neighbor* dan *2-opt*:



Gambar 10: Konstruksi jalur kereta dengan algoritma *Nearest Neighbor* dan *2-opt*

Dengan konfigurasi tersebut, didapat:

$$D_{total} = 35 + 11\sqrt{2} \approx 50.56 \tag{21}$$

$$T_{cycle} = \frac{35 + 11\sqrt{2}}{1} + 2 \cdot 13 = 61 + 11\sqrt{2} \quad \text{s} \tag{22}$$

$$\Phi = \frac{2 \times 6}{61 + 11\sqrt{2}} \approx 0.1567 \quad \text{penumpang/s} \tag{23}$$

$$I_{SD} = \frac{10}{13} \approx 0.77 \tag{24}$$

V. Kesimpulan

Dari hasil kalkulasi, didapat bahwa hasil heuristik 1 sampai 3 paling unggul. Akan tetapi, algoritma *Nearest Neighbor* dan *2-opt* dapat dikembangkan lagi untuk mencapai hasil yang lebih baik.

Didapat bahwa heuristik 1 sampai 3 terbukti efektif. Dalam permainan *real-time*, pemain lebih baik menggunakan heuristik 1 sampai 3 untuk mencapai hasil yang cukup baik, dibandingkan dengan algoritma *Nearest Neighbor* dan *2-opt* yang memiliki kompleksitas $O(N^3)$.

A. Heuristik 1

Dengan menggunakan heuristik 1, walaupun nilai Φ serupa, nilai I_{SD} jauh lebih baik dibandingkan konstruksi jalur kereta tanpa heuristik.

B. Heuristik 2

Dengan menggunakan heuristik 1 dan 2, nilai Φ meningkat dua kali lipat dibandingkan dengan hanya menggunakan heuristik 1, tanpa mengorbankan metrik lainnya.

C. Heuristik 3

Dengan menggunakan heuristik 1 sampai 3, didapat nilai Φ. Walaupun nilainya lebih rendah daripada hanya menggunakan heuristik 1 dan 2, didapat dua lintasan yang dua kali lebih pendek. Karena masing-masing lintasan melayani stasiun dua kali lebih sedikit dan tetap memiliki nilai Φ yang serupa dengan hasil heuristik sebelumnya, hasilnya tetap jauh lebih efisien.

D. Algoritma Nearest Neighbor dan 2-opt

Karena keterbatasan algoritma dan untuk menjaga kompleksitas algoritma rendah, hanya dibuat satu jalur kereta. Oleh karena itu, hasil dari algoritma ini hanya bisa dibandingkan dengan hasil heuristik 1 dan 2.

Hasil dari Heuristik 1 dan 2 memiliki jalur kereta yang 16.0% lebih panjang daripada hasil algoritma *Nearest Neighbor* (NN) dan *2-opt*. Nilai Φ dari hasil algoritma NN + *2-opt* juga 10.5% lebih tinggi. Namun karena keterbatasan algoritma, nilai dari I_{SD} algoritma NN + *2-opt* lebih rendah.

Ucapan Terimakasih

Penulis mengucapkan terima kasih kepada:

1) Tuhan Yang Maha Esa,
2) orang tua penulis,
3) Bapak dosen pengampu mata kuliah Matematika Diskrit IF1220,
4) teman-teman penulis, dan
5) pihak-pihak lain

yang telah mendukung penulis selama pembelajaran dan proses pengerjaan makalah ini.

Daftar Pustaka

[1] https://dinopoloclub.com/games/mini-metro/ [Diakses 18 Juni 2025]
[2] https://id.wikipedia.org/wiki/Graf_Petersen#/media/Berkas:Petersen_graph.svg [Diakses 18 Juni 2025]
[3] Dantzig, George Bernard; Ramser, John Hubert (October 1959). "The Truck Dispatching Problem". Management Science. 6 (1): 80–91. doi:10.1287/mnsc.6.1.80
[4] https://en.wikipedia.org/wiki/Vehicle_routing_problem#/media/File:Figure_illustrating_the_vehicle_routing_problem.png [Diakses 18 Juni 2025]
[5] G. Gutin, A. Yeo and A. Zverovitch, Exponential Neighborhoods and Domination Analysis for the TSP, in The Traveling Salesman Problem and Its Variations, G. Gutin and A.P. Punnen (eds.), Kluwer (2002) and Springer (2007).
[6] G. A. Croes, A method for solving traveling salesman problems. Operations Res. 6 (1958), pp., 791-812.
[7] M. M. Flood, The traveling-salesman problem. Operations Res. 4 (1956), pp., 61-75.
[8] https://en.wikipedia.org/wiki/2-opt#/media/File:2-opt_wiki.svg [Diakses 19 Juni 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.

Jatinangor, 20 Juni 2025

Muhammad Iqbal Raihan - 13524011