Vol. 5 No. 2, 2024, pp. 180-192 **Jurnal Pepadun**

Perbandingan *Cheapest Insertion Heuristic* dan Algoritma *Christofides* untuk Menentukan *Tour* Pasar Tradisional di Kota Bandar Lampung

[1]Micelle Yap Aswin, [*2]Wamiliana, [3]Fitriani, [4]Muslim Ansori, dan [5]Notiragayu
[1,2,3,4,5]Jurusan Matematika, Fakultas MIPA, Universitas Lampung,
Jl. Prof. Soemantri Brojonegoro No. 1, Bandar Lampung, Indonesia 35145
e-mail: [1]micelleyap28@gmail.com, [*2]**wamiliana.1963@fmipa.unila.ac.id,** [3]fitriani.1984@fmipa.unila.ac.id,
[4]muslim.ansori@fmipa.unila.ac.id, [5]notiragayu@fmipa.unila.ac.id

Abstract — *The Traveling Salesman Problem is a problem to determine the cheapest cost for a salesman staring from original location to all destination on his objective and return to the original location. In this study, a comparison of the Cheapest Insertion Heuristic and the Christofides Algorithm will be carried out to determine the traditional market tour in Bandar Lampung. The results obtained show that the solution obtained manually using the Cheapest Insertion Heuristic and the Christofides Algorithm produce the same solution, which is 216 minutes, while using Python programming, the Cheapest Insertion Heuristic provides a solution in 217 minutes and using the Christofides Algorithm in 226 minutes. However, because the solution obtained manually or by programming still produces intersections in the tour, to get better solution, the tour may be revised/repaired by removing the intersections. After repairing the tour, the same solution is obtained, whether using the Cheapest Insertion Heuristic or the Christofides Algorithm, which is 216 minutes.*

Keywords**:** *Traveling Salesman Problem; Cheapest Insertion Heuristic; Christofides Algorithm; Traditional Market.*

1. PENDAHULUAN

Saat ini, matematika telah mengalami pertumbuhan yang signifikan dan menjadi fokus penelitian yang intens oleh para ilmuwan, salah satunya pada bidang riset operasi. Riset operasi berkenaan dengan pengambilan keputusan optimal dalam penyusunan model dari sistem-sistem, baik deterministik maupun probabilistik yang berasal dari kehidupan nyata. Riset operasi banyak digunakan untuk menggambarkan suatu masalah yang melibatkan pengambilan keputusan dan optimalisasi, sementara untuk menyelesaikan masalah tersebut, teori graf seringkali digunakan sebagai alat analisis yang efektif. Salah satu topik bahasan dari teori graf adalah Hamiltonian dan *Travelling Salesman Problem* yang merupakan salah satu contoh Hamiltonian. *Traveling Salesman Problem* (TSP) adalah permasalahan optimasi kombinatorial yang bertujuan mencari rute terpendek bagi seorang *salesman* yang harus mengunjungi setiap kota tepat satu kali dan kembali ke titik asal [1].

TSP dianggap sebagai salah satu masalah klasik dalam masalah desain jaringan dan sangat populer. Permasalahan tersebut dirumuskan secara matematis oleh ahli matematika Irlandia, Willian Rowan Hamilton yang menciptakan permainan *icosian*, sebuah permainan matematika pada tahun 1856. TSP mempunyai sifat yang sama dengan permainan ini, oleh karena itu TSP juga dikenal sebagai salah satu masalah Hamiltonian. TSP mendapatkan popularitas di kalangan ilmiah di Amerika Serikat dan Eropa sepanjang tahun 1950-an dan 1960-an setelah adanya upaya untuk mengatasinya agar dapat memperoleh penghargaan dari RAND Corporation di Santa Monica [2].

TSP dapat direpresentasikan dengan menggunakan graf *G(V,E)* dengan V adalah himpunan titik-titik ($V \neq \varnothing$) yang mewakili lokasi/tempat yang akan dikunjungi dan *E* adalah himpunan sisi e_{ij} yang menghubungkan titik-titik di *V*. Sisi-sisi dapat mewakili jalan-jalan yang menghubungkan antar lokasi. Pada tiap sisi diberikan bobot $c_{ij} \geq 0$ yang dapat mewakili jarak/biaya/waktu yang diperlukan untuk menghubungkan titik *i* dan *j*. TSP merupakan masalah yang populer dan telah banyak diteliti. Sebagai contoh, *Cheapest Insertion Heuristics* telah diteliti antara lain oleh [3-9], *Bruce Force Heuristics* digunakan antara lain oleh [10-12], metode *Nearest Neighbour* digunakan antara lain oleh [12-14], dan metode *Christofides* digunakan dalam evaluasi rute pelayaran kapal [15]. Pada penelitian ini dilakukan perbandingan *Cheapest Insertion Heuritic* dan Algoritma *Christofides* untuk menentukan tur pasar tradisional di Bandar Lampung secara manual dan dengan menggunakan bantuan bahasa pemograman Python.

Vol. 5 No. 2, 2024, pp. 180-192 **Jurnal Pepadun**

2. METODOLOGI PENELITIAN

2. 1 *Cheapest Insertion Heuristic* (CIH)

Secara umum, CIH merupakan algoritma sederhana yang dilakukan penyisipan terhadap tempat yang akan dikunjungi dan menghitung jarak yang ditempuh. Secara lengkap langkah-langkah dalam pengerjaan CIH yaitu sebagai berikut.

i). Langkah 1: Pencarian dimulai dengan menentukan 3 titik yang memiliki jarak terpendek dan jumlahkan jarak antar 3 titik tersebut (total bobot).
ii). Langkah 2: Cari kandidat sisi yang akan dipilih untuk dimasukkan ke dalam *subtour* baru.
iii). Langkah 3: Hitung masing-masing kandidat sisi yang sudah dipilih, dengan rumus total bobot-bobot sisi yang akan dibuang + bobot sisi yang akan ditambah + bobot sisi yang menghubungkan antara titik yang dibuang dengan titik yang ditambah.
iv). Langkah 4: Cari bobot nilai yang terkecil dari kandidat sisi yang sudah dihitung.
v). Langkah 5: Buat *subtour* baru dengan sisi yang sudah dipilih.
vi). Ulangi langkah-langkah tersebut sampai semua titik masuk ke dalam *subtour* (untuk langkah selanjutnya dimulai dari langkah ke-2 dengan total bobot dan *subtour* yang baru, yang sudah diperoleh dari sebelumnya).

2. 2 Algoritma *Christofides*

Algoritma *Christofides* digunakan untuk menghasilkan solusi aproksimasi dalam TSP pada graf tak berarah berbobot lengkap. Algoritma ini mencari nilai minimum bobot dengan menggunakan *Minimum Spanning Tree* (MST) sehingga menghasilkan graf yang memiliki nilai optimal. Selanjutnya, hasil dari algoritma MST tersebut akan digunakan untuk membentuk sirkuit Euler sehingga dapat menjadi aproksimasi dari solusi TSP [15] . Berikut ini adalah langkah-langkah Algoritma Christofides untuk menyelesaikan TSP:

i). Cari *minimum spanning tree* yang menghubungkan tiap n titik dari graf (dapat menggunakan Algoritma Kruskal, Algoritma Prim, dan lainnya. Pada penelitian ini digunakan Algoritma Kruskal).
ii). Tentukan titik graf yang berderajat ganjil dan hubungkan dengan titik yang berderajat ganjil juga sehingga diperoleh titik berderajat genap.
iii). Bentuk sirkuit Euler.
iv). Kemudian reduksi titik yang berderajat lebih dari dua dengan cara menghapus sisi sehingga seluruh titik menjadi berderajat dua.

2. 3 Data yang Digunakan.

Data yang digunakan pada penelitian ini merupakan data yang diperoleh dari Google Maps. Tabel 1 berikut ini merupakan data waktu tempuh antar pasar tradisional yang ada di Bandar Lampung dalam satuan menit.

Tabel 1. Jarak 25 pasar tradisional di Kota Bandar Lampung (dalam satuan menit).

KODE	NAMA PASAR	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}
v_1	Pasar Bambu Kuning	0	6	2	15	6	12	21	2	14	17	12	15	27	30	23	12	18	6	18	25	18	14	11	7	21
v_2	Pasar Tugu	6	0	6	15	9	15	25	6	16	21	14	22	26	24	22	11	20	10	20	24	23	19	10	9	15
v_3	Pasar Bawah	2	6	0	16	7	11	22	1	13	18	11	18	26	29	24	13	16	6	16	24	19	14	12	5	19
v_4	Pasar Rakyat Way Halim	15	15	16	0	11	24	13	21	13	17	11	16	26	29	23	11	16	6	17	24	20	14	10	6	19
v_5	Pasar Koga	6	9	7	11	0	18	16	7	20	11	18	16	33	29	17	8	22	12	23	30	19	8	7	13	14
v_6	Pasar Kangkung	12	15	11	24	18	0	34	13	5	35	3	25	14	25	38	25	7	17	8	17	25	30	22	9	27
v_7	Pasar Tempel Rajabasa Raya	21	25	22	13	16	34	0	18	31	7	28	16	36	32	19	16	34	24	34	33	19	11	16	24	15
v_8	Pasar Tengah	2	6	1	21	7	13	18	0	15	23	13	22	24	33	27	15	18	7	19	26	21	17	14	7	23
v_9	Pasar Cimeng	14	16	13	13	20	5	31	15	0	28	3	20	23	27	34	24	3	12	3	21	21	25	23	8	28
v_{10}	Pasar Untung Suropati	17	21	18	17	11	35	7	23	28	0	26	17	30	24	12	10	31	21	32	27	21	5	12	21	9
v_{11}	Pasar Mambo	12	14	11	11	18	3	28	13	3	26	0	24	20	29	38	27	5	15	6	22	24	28	24	9	31
v_{12}	Pasar Tani Kemiling	15	22	18	16	16	25	16	22	20	17	24	0	37	39	28	20	22	14	23	34	6	16	19	18	25
v_{13}	Pasar Senggol	27	26	26	26	33	14	36	24	23	30	20	37	0	19	33	28	22	28	23	5	38	35	31	24	27
v_{14}	Pasar Phillip	30	24	29	29	29	25	32	33	27	24	29	39	19	0	29	22	25	31	25	16	40	28	24	26	19
v_{15}	Pasar Baru Way Kandis	23	22	24	23	17	38	19	27	34	12	38	28	33	29	0	13	34	26	34	30	32	18	14	26	11

KODE	NAMA PASAR	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}
v_{16}	Pasar Way Halim Permai	12	11	13	11	8	25	16	15	24	10	27	20	28	22	13	0	24	15	25	26	24	9	2	15	8
v_{17}	Pasar Kota Karang	18	20	16	16	22	7	34	18	3	31	5	22	22	25	34	24	0	15	1	24	25	29	26	11	32
v_{18}	Pasar Tamin	6	10	6	6	12	17	24	7	12	21	15	14	28	31	26	15	15	0	14	26	15	16	13	6	22
v_{19}	Pasar Koppamastera	18	20	16	17	23	8	34	19	3	32	6	23	23	25	34	25	1	14	0	23	23	26	23	10	30
v_{20}	Pasar Panjang	25	24	24	24	30	17	33	26	21	27	22	34	5	16	30	26	24	26	23	0	38	32	27	21	24
v_{21}	Pasar Tempel Beringin Raya	18	23	19	20	19	25	19	21	21	21	24	6	38	40	32	24	25	15	23	38	0	18	20	18	26
v_{22}	Pasar Stasiun Labuhan Ratu	14	19	14	14	8	30	11	17	25	5	28	16	35	28	18	9	29	16	26	32	18	0	8	15	14
v_{23}	Pasar Tempel Way Halim	11	10	12	10	7	22	16	14	23	12	24	19	31	24	14	2	26	13	23	27	20	8	0	13	10
v_{24}	Pasar Tempel Gotong Royong	7	9	5	6	13	9	24	7	8	21	9	18	24	26	26	15	11	6	10	21	18	15	13	0	22
v_{25}	Pasar Tempel Way Dadi	21	15	19	19	14	27	15	23	28	9	31	25	27	19	11	8	32	22	30	24	26	14	10	22	0

3. HASIL DAN PEMBAHASAN

3.1. Penyelesaian dengan *Cheapest Insertion Heuristics*

Berdasarkan langkah-langkah penyelesaian yang sudah dijelaskan pada Metodologi Penelitian, berikut ini diberikan penyelesaian secara manual dalam bentuk tablel iterasi yang ada pada Tabel 2. Akan tetapi, karena iterasi yang dilakukan ada sebanyak 23 iterasi, maka pada tabel hanya ada perhitungan untuk tiga iterasi pertama dan dua iterasi terakhir.

Tabel 2. Tabel iterasi penghitungan dengan menggunakan *Cheapest Insertion Heuristics*.

Iterasi	Subtour	Total Bobot	|E|=n?	Kandidat Sisi yang akan dipilih	Perhitungan	Sisi yang dipilih dan yang ditambah	Sisi yang dibuang	Total bobot baru	Subtour Baru
1	$v_1 - v_3 - v_8 - v_1$	5	Tidak	$e_{1,2}$= 6 $e_{3,24}$= 5 $e_{8,2}$= 6	$5 - c_{1,3} + c_{1,2} + c_{2,3} = 5 - 2 + 6 + 6 = 15$ $5 - c_{3,1} + c_{1,24} + c_{3,24} = 5 - 2 + 7 + 5 = 15$ $5 - c_{1,8} + c_{8,2} + c_{1,2} = 5 - 2 + 6 + 6 = 15$	$e_{1,2}$ $e_{2,3}$	$e_{1,3}$	15	$v_1 - v_2 - v_3 - v_8 - v_1$
2	$v_1 - v_2 - v_3 - v_8 - v_1$	15	Tidak	$e_{1,5}$ = 6 $e_{2,5}$ = 9 $e_{3,24}$=5 $e_{8,5}$ =7	$15 - c_{1,2} + c_{1,5} + c_{2,5} = 15 - 6 + 6 + 9 = 24$ $15 - c_{2,3} + c_{2,5} + c_{3,5} = 15 - 6 + 9 + 7 = 25$ $15 - c_{2,3} + c_{3,24} + c_{2,24} = 15 - 6 + 5 + 9 = 23$ $15 - c_{1,8} + c_{8,5} + c_{1,5} = 15 - 2 + 7 + 6 = 26$	$e_{3,24}$ – $e_{2,24}$	$e_{2,3}$	23	$v_1 - v_2 - v_{24} - v_3 - v_8 - v_1$
3	$v_1 - v_2 - v_{24} - v_3 - v_8 - v_1$	23	Tidak	$e_{1,5}$= 6 $e_{2,5}$= 9 $e_{3,18}$= 6 $e_{24,5}$= 6 $e_{8,5}$= 7	$23 - c_{1,2} + c_{1,5} + c_{2,5} = 23 - 6 + 6 + 9 = 32$ $23 - c_{2,24} + c_{2,5} + c_{24,5} = 23 - 9 + 9 + 13 = 36$ $23 - c_{24,3} + c_{3,18} + c_{24,18} = 23 - 5 + 6 + 6 = 30$ $23 - c_{2,24} + c_{4,24} + c_{2,4} = 23 - 9 + 6 + 15 = 35$ $23 - c_{1,8} + c_{8,5} + c_{1,5} = 23 - 2 + 7 + 6 = 34$	$e_{3,18}$- $e_{24,18}$	$e_{24,3}$	30	$v_1 - v_2 - v_{24} - v_{18} - v_3 - v_8 - v_1$
				Iterasi ke 4 sampai 21 tidak dimasukkan					
22	$v_1 - v_5 - v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_{23} - v_{16} - v_{15} - v_{25} - v_{14} - v_{20} - v_2 - v_{24} - v_{19} - v_{17} - v_9 - v_6 - v_{11} - v_4 - v_{18} - v_3 - v_8 - v_1$	209	Tidak	$e_{1,13}$ =27 $e_{5,13}$ = 33 $e_{21,13}$= 38 $e_{12,13}$= 37 $e_{7,13}$= 36 $e_{10,13}$= 30 $e_{22,13}$= 35 $e_{23,13}$= 31 $e_{16,13}$ = 28 $e_{15,13}$= 33 $e_{25,13}$= 27 $e_{14,13}$= 19 $e_{20,13}$= 5 $e_{2,13}$= 26 $e_{24,13}$= 24 $e_{19,13}$= 23 $e_{17,13}$= 22 $e_{9,13}$= 23 $e_{6,13}$= 14 $e_{11,13}$= 20 $e_{4,13}$= 26 $e_{18,13}$= 28 $e_{3,13}$= 26 $e_{8,13}$= 24	$209 - c_{1,5} + c_{1,13} + c_{5,13} = 209 - 6 + 27 + 33 = 263$ $209 - c_{5,21} + c_{5,13} + c_{21,13} = 209 - 19 + 33 + 38 = 261$ $209 - c_{5,21} + c_{21,13} + c_{5,13} = 209 - 19 + 38 + 33 = 261$ $209 - c_{12,7} + c_{12,13} + c_{7,13} = 209 - 16 + 37 + 36 = 266$ $209 - c_{12,7} + c_{7,13} + c_{12,13} = 209 - 16 + 36 + 37 = 266$ $209 - c_{7,10} + c_{10,13} + c_{7,13} = 209 - 7 + 30 + 36 = 268$ $209 - c_{22,23} + c_{22,13} + c_{23,13} = 209 - 8 + 35 + 31 = 267$ $209 - c_{22,23} + c_{23,13} + c_{22,13} = 209 - 8 + 31 + 35 = 267$ $209 - c_{16,15} + c_{16,13} + c_{15,13} = 209 - 13 + 28 + 33 = 257$ $209 - c_{16,15} + c_{15,13} + c_{16,13} = 209 - 13 + 33 + 28 = 257$ $209 - c_{1,25} + c_{25,13} + c_{14,13} = 209 - 15 + 27 + 19 = 236$ $209 - c_{14,2} + c_{14,13} + c_{25,13} = 209 - 24 + 19 + 27 = 231$ $209 - c_{20,2} + c_{20,13} + c_{2,13} = 209 - 24 + 5 + 26 = 216$ $209 - c_{2,14} + c_{2,13} + c_{20,13} = 209 - 24 + 26 + 5 = 216$ $209 - c_{24,19} + c_{24,13} + c_{19,13} = 209 - 10 + 24 + 23 = 246$ $209 - c_{24,19} + c_{19,13} + c_{24,13} = 209 - 10 + 23 + 24 = 246$ $209 - c_{17,9} + c_{17,13} + c_{9,13} = 209 - 3 + 22 + 23 = 251$ $209 - c_{9,6} + c_{9,13} + c_{6,13} = 209 - 5 + 23 + 14 = 241$ $209 - c_{9,6} + c_{6,13} + c_{9,13} = 209 - 5 + 14 + 23 = 241$ $209 - c_{11,4} + c_{11,13} + c_{4,13} = 209 - 11 + 20 + 26 = 244$ $209 - c_{11,4} + c_{4,13} + c_{11,13} = 209 - 11 + 26 + 20 = 244$ $209 - c_{18,4} + c_{18,13} + c_{4,13} = 209 - 6 + 28 + 26 = 257$ $209 - c_{3,18} + c_{3,13} + c_{18,13} = 209 - 6 + 26 + 28 = 257$ $209 - c_{1,8} + c_{8,13} + c_{1,13} = 209 - 2 + 24 + 27 = 258$	$e_{20,13}$ – $e_{2,13}$	$e_{20,2}$	216	$v_1 - v_5 - v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_{23} - v_{16} - v_{15} - v_{25} - v_{14} - v_{20} - v_{13} - v_2 - v_{24} - v_{19} - v_{17} - v_9 - v_6 - v_{11} - v_4 - v_{18} - v_3 - v_8 - v_1$

Iterasi	Subtour	Total Bobot	|E|=n?	Kandidat Sisi yang akan dipilih	Perhitungan	Sisi yang dipilih dan yang ditambah	Sisi yang dibuang	Total bobot baru	Subtour Baru
23	$v_1 - v_5 - v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_{23} - v_{16} - v_{15} - v_{25} - v_{14} - v_{20} - v_{13} - v_2 - v_{24} - v_{19} - v_{17} - v_9 - v_6 - v_{11} - v_4 - v_{18} - v_3 - v_8 - v_1$	216	Ya						

Gambar 1 berikut merupakan gambar pemetaan hasil *tour* menggunakan *Cheapest Insertion Heuristic* yang didapat dari Tabel 1. Pada Gambar 1 tersebut didapat bahwa pada *tour* yang dihasilkan terdapat perpotongan. Hal ini mengindikasikan bahwa *tour* tersebut masih dapat diperbaiki untuk mendapatkan hasil yang lebih baik. Oleh karena itu, dilakukan perbaikan *tour* dengan cara menentukan perpotongan, menghapus perpotongan, dan menambahkan sisi baru sebagai berikut:

Gambar 1. Pemetaan *tour* berdasarkan hasil pada Tabel 1.

i). Hapus sisi $e_{5,21}$ dan sisi $e_{4,18}$ tambah sisi $e_{4,5}$ dan sisi $e_{18,21}$, dengan penambahan/pengurangan waktu adalah: $-19 - 6 + 11 + 15 = 1$.

ii). Hapus sisi $e_{3,18}$ dan sisi $e_{2,24}$ tambah sisi $e_{18,24}$ dan sisi $e_{2,3}$, dengan penambahan/pengurangan waktu adalah: $-6 - 9 + 6 + 6 = -3$.

iii). Hapus sisi $e_{19,24}$, sisi $e_{4,11}$ dan sisi $e_{6,9}$, tambah sisi $e_{9,24}$ dan sisi $e_{11,19}$, dengan penambahan/pengurangan waktu adalah: $-10 - 11 - 5 + 8 + 6 = -12$.

iv). Hapus sisi $e_{2,13}$ dan tambah sisi $e_{13,6}$, dengan penambahan/pengurangan waktu adalah: $-26 + 14 = -12$.

v). Hapus sisi $e_{1,5}$ dan sisi $e_{22,23}$, tambah sisi $e_{1,22}$ dan sisi $e_{5,23}$, dengan penambahan/pengurangan waktu adalah: $-6 - 8 + 14 + 7 = 7$.

vi). Hapus sisi $e_{4,5}$ dan sisi $e_{15,16}$, tambah sisi $e_{2,5}$, sisi $e_{4,15}$ dan sisi $e_{4,16}$, dengan penambahan/pengurangan waktu adalah: $-11 - 13 + 9 + 23 + 11 = 19$.

Selisih bobot setelah proses penghapusan dan penambahan sisi : $1 - 3 - 12 - 12 + 7 + 19 = 0$ menit. Total jarak awal + selisih bobot setelah proses penghapusan dan penambahan sisi: $216 + 0 = 216$ menit. Berdasarkan hasil perhitungan, diperoleh *tour* yang baru $v_1 - v_8 - v_3 - v_2 - v_5 - v_{23} - v_{16} - v_4 - v_{15} - v_{25} - v_{14} - v_{20} - v_{13} - v_6 - v_{11} - v_{19} - v_{17} - v_9 - v_{24} - v_{18} - v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_1$ dengan total bobot 216 menit. *Tour* yang baru tersebut ditunjukkan pada Gambar 2 berikut.

Gambar 2. Hasil yang didapat setelah perbaikan *tour*.

Selain secara manual, penyelesaian dengan CIH juga dilakukan dengan menggunakan bahasa pemograman Python. Gambar 3 berikut adalah potongan program dari CIH untuk kasus penentuan *tour* dari 25 pasar tradisional di Kota Bandar Lampung.

```
# Fungsi untuk menghitung total bobot dari sebuah rute
def calculate_total_distance(route, distance_matrix):
    total_distance = 0
    for i in range(len(route) - 1):
        from_point = route[i]
        to_point = route[i + 1]
        total_distance +=
distance_matrix[from_point][to_point]
    return total_distance

# Inisialisasi matriks jarak antara titik-titik
distance_matrix = {}

# Inisialisasi rute awal dengan 3 titik terdekat
start_points = ['V1', 'V2', 'V3']
remaining_points = ['V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19',
'V20', 'V21', 'V22', 'V23', 'V24', 'V25']

route = start_points
total_distance = calculate_total_distance(route,
distance_matrix)

# Fungsi untuk mencari kandidat garis
def find_candidate_insertion(route, remaining_points,
distance_matrix):
    min_insertion_cost = float('inf')
    best_insertion = None

    for point_to_insert in remaining_points:
        for i in range(len(route) - 1):
            from_point = route[i]
            to_point = route[i + 1]
            new_distance =
(distance_matrix[from_point][point_to_insert] +
                            distance_matrix[point_to_insert][t
o_point] -
                            distance_matrix[from_point][to_poi
nt])
            if new_distance < min_insertion_cost:
                min_insertion_cost = new_distance
                best_insertion = (point_to_insert, i)

    return best_insertion, min_insertion_cost

# Langkah 2-5
```

Gambar 3. *Screenshoot* potongan program CIH.

Program tersebut menghasilkan *output* dengan *tour* $v_1 - v_5 - v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_{23} - v_{16} - v_{24} - v_9 - v_{19} - v_{17} - v_{14} - v_{20} - v_{13} - v_6 - v_{11} - v_4 - v_{18} - v_{25} - v_{15} - v_2 - v_8 - v_3 - v_1$ dengan bobot 217 menit. Gambar 4 berikut adalah gambar dari *tour* yang didapat dengan menggunakan bahasa pemograman Python.

Gambar 4. Pemetaan *tour* berdasarkan hasil dari pemograman Python untuk CIH.

Seperti paga Gambar 1, pada Gambar 4 didapat juga perpotongan dari *tour* yang dihasilkan. Dengan demikian, perlu dilakukan perbaikan *tour* dengan cara menentukan perpotongan, menghapus perpotongan, dan menambahkan sisi baru sebagai berikut:

i). Hapus sisi $e_{17,14}, e_{18,25}, e_{24,16}$ tambah $e_{25,14}, e_{18,24}, e_{16,17}$ dengan penambahan/pengurangan waktu adalah $-25-22-15+19+6+24=-13$.

ii). Hapus $e_{1,3}$, $e_{2,8}$, tambah $e_{8,1}$, $e_{3,2}$ dengan penambahan/pengurangan waktu adalah $-2-6+2+6=0$.

iii). Hapus $e_{21,5}$, $e_{18,4}$, $e_{2,15}$ tambah $e_{21,18}$, $e_{4,15}$ dengan penambahan/pengurangan waktu adalah $-19-6-22+15+23=1$.

iv). Hapus $e_{11,4}$, $e_{9,19}, e_{16,17}$ tambah $e_{19,11}$, $e_{17,9}$, $e_{16,4}$ dengan penambahan/pengurangan waktu adalah $-11-3-24+6+18+11=-3$.

v). Hapus $e_{1,5}$, $e_{22,23}$ tambah $e_{22,1}$, $e_{5,23}$, $e_{5,2}$ dengan penambahan/pengurangan waktu adalah $-6-8+14+7+9=14$.

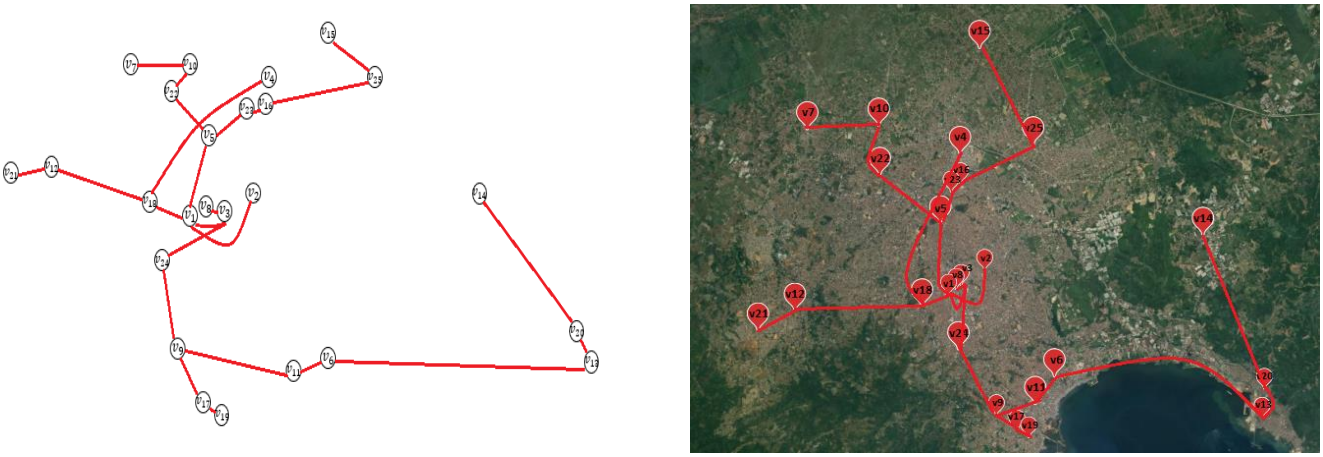
Selisih bobot setelah proses penghapusan dan penambahan sisi : $-13+0+1-3+14=-1$. Bobot awal dikurangi selisih bobot setelah penghapusan dan penambahan sisi adalah $217-1=216$ menit, sehingga *tour* terpendek yang dihasilkan setelah proses penghitungan kembali adalah 216 menit. *Tour* yang didapat setelah dilakukan proses penghapusan dan penambahan sisi untuk menghilangkan perpotongn adalah $v_1-v_8-v_3-v_2-v_5-v_{23}-v_{16}-v_4-v_{15}-v_{25}-v_{14}-v_{20}-v_{13}-v_6-v_{11}-v_{19}-v_{17}-v_9-v_{24}-v_{18}-v_{21}-v_{12}-v_7-v_{10}-v_{22}-v_1$ dengan bobot 216 menit. Gambar 5 berikut merupakan *tour* tersebut.

Gambar 5. *Tour* yang didapat setelah dilakukan perbaikan *tour* untuk Gambar 4.

Vol. 5 No. 2, 2024, pp. 180-192 **Jurnal Pepadun**

3.1.2 Penyelesaian dengan Algoritma *Christofides*

Algoritma *Christofides* memiliki banyak kemungkinan solusi, karena banyak kemungkinan cara untuk menghubungkan titik yang berderajat ganjil. Algoritma *Christofides* dilakukan dengan menentukan dahulu MST dari 25 titik menggunakan Algoritma Kruskal. Gambar 6 berikut menunjukkan hasil yang didapat.



(a) (b)

Gambar 6. (a) *Minimum spanning tree* dari 25 pasar tradisional di Kota Bandar Lampung (b) *Minimum spanning tree* dari 25 titik pasar tradisional di Kota Bandar Lampung di *Google Earth*.

Langkah selanjutnya adalah membentuk *Eulerian tour* dengan cara titik graf yang berderajat ganjil akan dihubungkan dengan titik yang berderajat ganjil juga, sehingga diperoleh titik berderajat genap. Seperti telah dinyatakan sebelumnya bahwa banyak cara untuk membentuk *Eulerian tour* karena banyaknya titik yang berderajat ganjil. Salah satunya adalah Gambar 7 yang merupakan gambar yang diperoleh setelah menghubungkan titik berderajat ganjil, sehingga berderajat genap. Untuk membentuk *Hamiltonian tour*, titik yang berderajat lebih dari 2 akan dioptimalkan dengan cara membuang sisinya.

Gambar 7. Pembentukan *Eulerian tour*.

Berdasarkan Gambar 7 dapat dilihat bahwa titik v_9 berderajat 4, maka akan dibuat menjadi berderajat 2 dengan mengeliminasi 2 sisi. Selanjutnya dipilih antara sisi $e_{9,24}, e_{9,11}, e_{9,17}$ dan $e_{9,19}$. Misalkan yang dipilih sisi $e_{9,19}$ dan $e_{9,11}$ untuk dihapus, kemudian ditambah $e_{19,11}$, maka hasil yang diperoleh sebagaimana ditunjukkan pada Gambar 8 sebagai berikut.

Gambar 8. Hasil yang didapat pada opsi pertama setelah titik v_9 berderajat 2.

Dari Gambar 8 didapat bahwa titik v_3 masih berderajat 4. Oleh karena itu, titik v_3 akan dibuat berderajat 2 dengan mengeliminasi 2 sisinya. Selanjutnya akan dipilih sisi $e_{3,8}, e_{3,1}, e_{3,24}$ dan $e_{3,2}$. Misal dipilih sisi $e_{3,24}$ dan $e_{3,1}$, maka titik v_1 dan v_{24} akan berderajat ganjil. Kedua titik akan dibuat berderajat genap dengan cara menghubungkan kedua titik tersebut dan didapatkan hasil sebagaimana Gambar 9 berikut.

Gambar 9. Hasil yang didapat pada opsi pertama setelah titik v_3 berderajat 2.

Berdasarkan Gambar 9 diketahui bahwa ada tiga titik yang masih berderajat lebih dari 2, yaitu titk v_1, v_5 , dan v_{18}. Pada tahap selanjutnya akan terbagi menjadi 2 opsi. Hal ini karena algoritma ini memiliki banyak kemungkinan solusi, sehingga terdapat banyak kemungkinan cara untuk menghubungkan titik yang berderajat ganjil.

3.1.2.1 Opsi Pertama

Karena masih terdapat titik yang berderajat 4, maka akan dibuat menjadi berderajat 2.

i). Untuk v_1, pilih sisi $e_{1,5}$, $e_{1,2}$, $e_{1,24}$, dan $e_{1,18}$. Hapus sisi $e_{1,24}$, dan $e_{1,18}$ kemudian hubungkan titik v_{18} dan titik v_{24}.

ii). Untuk v_5, pilih sisi $e_{5,22}$, $e_{5,14}$, $e_{5,23}$, dan $e_{5,1}$. Hapus sisi $e_{5,22}$, dan $e_{5,1}$ kemudian hubungkan titik v_{22} dan titik v_1.

iii). Untuk v_{18}, pilih sisi $e_{18,12}$, $e_{18,4}$, $e_{18,8}$, dan $e_{18,24}$. Hapus sisi $e_{18,4}$, dan $e_{18,8}$ kemudian hubungkan titik v_8 dan titik v_4.

Gambar 10 berikut ini merupakan hasil *tour* dari opsi pertama.

Gambar 10. *Tour* yang didapat dari opsi pertama.

Hasil *tour* opsi pertama memiliki bobot 241 menit. Karena masih terdapat perpotongan, maka selanjutnya dilakukan penghitungan kembali dengan penghapusan dan penambahan sisi dengan cara menghapus sisi $e_{1,2}$, $e_{8,4}$, $e_{5,14}$, $e_{16,25}$, $e_{21,7}$, dan $e_{12,18}$, dan menambah sisi $e_{1,8}$, $e_{2,5}$, $e_{16,4}$, $e_{14,25}$, $e_{12,7}$, dan $e_{21,18}$. Selisih penghapusan dan penambahan bobot: $-6-21-29-8-19-14+2+9+11+19+16+15=-25$. Dengan demikian, bobot *tour* opsi pertama adalah $241-25=216$ menit, dan *tour* yang didapat adalah $v_1-v_8-v_3-v_2-v_5-v_{23}-v_{16}-v_4-v_{15}-v_{25}-v_{14}-v_{20}-v_{13}-v_6-v_{11}-v_{19}-v_{17}-v_9-v_{24}-v_{18}-v_{21}-v_{12}-v_7-v_{10}-v_{22}-v_1$. Gambar 11 berikut adalah gambar *tour* opsi pertama setelah proses perbaikan *tour*.

Gambar 11. *Tour* opsi pertama setelah proses perbaikan *tour*.

3.1.2.2 Opsi Kedua

Selanjutnya akan dilakukan perhitungan opsi kedua sebagai berikut:

i). Untuk v_1, pilih sisi $e_{1,5}$, $e_{1,2}$, $e_{1,24}$, dan $e_{1,18}$. Hapus sisi $e_{1,5}$, dan $e_{1,2}$ kemudian hubungkan titik v_2 dan titik v_5.

ii). Untuk v_5, pilih sisi $e_{5,22}$, $e_{5,14}$, $e_{5,23}$, dan $e_{5,1}$. Hapus sisi $e_{5,14}$, dan $e_{5,2}$ kemudian hubungkan titik v_2 dan titik v_{14}.

iii). Untuk v_{18}, pilih sisi $e_{18,12}$, $e_{18,4}$, $e_{18,8}$, dan $e_{18,24}$. Hapus sisi $e_{18,4}$, dan $v_{18}v_8$ kemudian hubungkan titik v_8 dan titik v_4.

Gambar 12 berikut ini merupakan hasil *tour* dari opsi kedua.

Gambar 12. *Tour* yang didapat dari opsi kedua.

Hasil *tour* opsi kedua ini memiliki bobot 231 menit. Karena masih terdapat perpotongan, maka selanjutnya dilakukan revisi *tour* dengan melakukan penghapusan sisi $e_{8,4}$, $e_{16,25}$, $e_{2,14}$, $e_{1,18}$, $e_{1,24}$, $e_{22,5}$, $e_{21,7}$, dan $e_{12,18}$, dan penambahan sisi $e_{16,4}$, $e_{25,14}$, $e_{18,24}$, $e_{1,8}$, $e_{22,1}$, $e_{5,2}$ $e_{12,7}$, dan $e_{21,18}$. Hasil penghapusan dan penambahan bobot adalah: $-21-8-24-6-7-8-19-14+11+19+6+2+14+9+16+15=-15$. Oleh karena itu, bobot setelah revisi *tour* untuk opsi kedua adalah $231-15=216$ menit, dengan *tour* $v_1-v_8-v_3-v_2-v_5-v_{23}-v_{16}-v_4-v_{15}-v_{25}-v_{14}-v_{20}-v_{13}-v_6-v_{11}-v_{19}-v_{17}-v_9-v_{24}-v_{18}-v_{21}-v_{12}-v_7-v_{10}-v_{22}-v_1$. Gambar 13 merupakan gambar *tour* yang didapat setelah melakukan revisi *tour*.

Gambar 13. *Tour* yang didapat dari opsi kedua setelah dilakukan revisi *tour*.

Opsi pertama dan opsi kedua ternyata menghasilkan total bobot dan *tour* yang sama. Selain secara manual, penyelesaian dengan Algoritma *Christofides* juga dilakukan dengan menggunakan bahasa pemograman Python. Gambar 14 berikut adalah potongan program dari Algoritma *Christofides* untuk kasus penentuan *tour* dari 25 pasar tradisional di Kota Bandar Lampung.

```
def christofides_tsp(self):
        adj_matrix = [[0] * self.V for _ in range(self.V)]
        for u, v, w in self.graph:
            adj_matrix[u][v] = w
            adj_matrix[v][u] = w
        euler_tour = self.find_euler_tour(adj_matrix)
        visited = [False] * self.V
        path = []
        for vertex in euler_tour:
            if not visited[vertex]:
                path.append(vertex)
                visited[vertex] = True
        return path

def main():
    V = 25  # Jumlah titik
    g = Graph(V)
    g.add_edge(x, y, distance)

    tsp_path = g.christofides_tsp()
    print("Christofides TSP Path:", tsp_path)

if __name__ == "__main__":
    main()
```

Gambar 14. *Screenshoot* potongan program Algoritma *Christofides.*

Tour yang diperoleh adalah $v_{21} - v_{12} - v_7 - v_{10} - v_{22} - v_{15} - v_{25} - v_{16} - v_{23} - v_5 - v_2 - v_{19} - v_{17} - v_{14} - v_{20} - v_{13} - v_6 - v_{11} - v_9 - v_{24} - v_8 - v_3 - v_1 - v_{18} - v_4 - v_{21}$ dengan total bobot 226 menit. Gambar 15 merupakan hasil pemetaan *tour* yang diperoleh dari bahasa pemrograman Python.

Gambar 15. Pemetaan *tour* berdasarkan hasil dari pemograman Phyton untuk Algoritma *Christofides.*

Karena masih terdapat perpotongan pada *tour*, maka dilakukan revisi *tour* dengan menghapus dan menambah sisi sebagai berikut:

i) Hapus $e_{1,3}$ dan $e_{8,24}$, tambah $e_{1,8}$ dan $e_{3,24}$ dengan penambahan/pengurangan waktu adalah $-2 - 7 + 2 + 5 = -2$.
ii) Hapus $e_{4,21}$, $e_{4,18}$, tambah $e_{21,18}$ dengan penambahan/pengurangan waktu adalah $-20 - 6 + 15 = -11$.
iii) Hapus $e_{3,24}$, $e_{1,18}$, tambah $e_{18,24}$ dengan penambahan/pengurangan waktu adalah $-5 - 6 + 6 = -5$.
iv) Hapus $e_{9,11}$, $e_{17,14}$, $e_{16,25}$, $e_{2,19}$, tambah $e_{9,17}$, $e_{25,14}$, $e_{11,19}$, $e_{2,3}$ dengan penambahan/pengurangan waktu adalah$-3 - 25 - 8 - 20 + 3 + 19 + 6 + 6 = -22$.
v) Hapus $e_{22,15}$ tambah $e_{22,1}$, $e_{16,4}$, $e_{4,15}$ dengan penambahan/pengurangan waktu adalah $-18 + 14 + 11 + 23 = 30$.

Gambar 16. *Tour* yang didapat setelah dilakukan perbaikan *tour* untuk Gambar 15.

Total penambahan/pengurangan waktu yaitu $-2-11-5-22+30=-10$. Bobot *tour* baru setelah revisi *tour* adalah $226-10=216$ menit dengan *tour* $v_1-v_8-v_3-v_2-v_5-v_{23}-v_{16}-v_4-v_{15}-v_{25}-v_{14}-v_{20}-v_{13}-v_6-v_{11}-v_{19}-v_{17}-v_9-v_{24}-v_{18}-v_{21}-v_{12}-v_7-v_{10}-v_{22}-v_1$. Gambar 16 merupakan hasil pemetaan dari perhitungan revisi *tour* terakhir dalam penelitian ini.

4. KESIMPULAN

Hasil yang diperoleh menunjukkan bahwa solusi yang diperoleh secara manual dengan menggunakan *Cheapest Insertion Heuristic* dan Algoritma *Christofides* menghasilkan solusi yang sama yaitu 216 menit. Di sisi lain, dengan menggunakan bahasa pemrograman Python, *Cheapest Insertion Heuristic* memberikan solusi dalam waktu 217 menit sedangkan Algoritma Christofides dalam waktu 226 menit. Namun, karena solusi yang diperoleh secara manual maupun dengan bahasa pemrograman Python masih menghasilkan perpotongan dalam *tour*, maka untuk mendapatkan solusi yang lebih baik, *tour* direvisi/diperbaiki dengan menghilangkan perpotongan tersebut. Setelah dilakukan perbaikan *tour*, diperoleh solusi yang sama, baik menggunakan *Cheapest Insertion Heuristic* maupun Algoritma Christofides, yaitu 216 menit.

DAFTAR PUSTAKA

[1] K. Saleh, Helmi & B. Prihandono, Penentuan Rute Terpendek Dengan Menggunakan Algoritma Cheapest Insertion Heuristic (Studi Kasus: PT. Wicaksana Overseas International Tbk. Cabang Pontianak). *Bimaster: Buletin Ilmiah Math. Stat. dan Terapannya*, vol. 4, no. 3, pp. 295–304, 2015, https://doi.org/10.26418/bbimst.v4i03.11888.

[2] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, & D. B. Shmyos, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1991.

[3] K. Kusrini & J. E. Istiyanto, Penyelesain Travelling Salesman Problem dengan Algoritma Cheapest Insertion Heuristic dan Basis Data. *Jurnal Informatika*, vol. 8, no. 2, pp. 109-114, 2008, https://doi.org/10.9744/informatika.8.2.pp.%20109-114.

[4] F. Fargiana, R. Respitawulan, Y. Fajar, D. Suhaedi, & E. Harahap, Implementation of Cheapest Insertion Heuristic Algorithm in Determining Shortest Delivery Route, *International Journal of Global Operations Research*, vol. 3, no. 2, pp. 37–45, 2022, https://doi.org/10.47194/ijgor.v3i2.137.

[5] M. Hilman & Y. Y. Sidik, Penentuan Rute Distribusi Menggunakan Metode Cheapest Insertion Heuristic (CIH) Guna Meminimalkan Pengeluaran Biaya Pada Ukm Aren Creativity di Kabupaten Ciamis, *Jurnal Industrial Galuh*, vol. 4, no. 2, pp. 51–61, 2022, https://doi.org/10.25157/jig.v4i2.3017.

[6] D. T. Wiyanti, Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem, *Jurnal Transformatika*, vol. 11, no. 1, pp. 1-6, 2013, https://doi.org/10.26623/transformatika.v11i1.76.

[7] E. Yulianto E & A. Setiawan, Optimasi Rute Sales Coverage Menggunakan Algoritma Cheapest Insertion Heuristic dan Layanan Google Maps API, *INTERNAL (Information System Journal)*, vol. 1, no. 1, pp. 39–54, 2018, https://doi.org/10.32627/internal.v1i1.326.

[8] M. Amelia, I. I. Sholeha, Y. R. Revangga, & Wamiliana, The Comparison of Brute Force, Cheapest-Insertion, and Nearest-Neighbor Heuristics for Determining the Shortest Tour for Visiting Malls in Bandar Lampung. *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, vol. 14, no. 1 , pp. 51-54 , 2024. http://dx.doi.org/10.36448/expert.v14i1.3715.

[9] L. V. Hignasari & E. D. Mahira, Optimization of Goods Distribution Route assisted by Google Maps with Cheapest Insertion Heuristic Algorithm (CIH). *SINERGI*, vol. 22, no. 2, pp. 132-138, 2018, https://dx.doi.org/10.22441/sinergi.2018.2.010.

[10] M. Kurniawan, F. Farida, & S. Agustini, Rute Terpendek Algoritma Particle Swarm Optimization dan Brute Force untuk Optimasi Travelling Salesman Problem, *Jurnal Teknik Informatika*, vol. 14, no. 2, pp. 191-200, 2021, https://doi.org/10.15408/jti.v14i2.19094.

[11] S. Violina, Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP), *TURCOMAT*, vol. 12, no. 8, pp. 1226–1229, 2021.

[12] A. Wilson, Y. Sibaroni, & I. Ummah, Analisis Penyelesaian Traveling Salesman Problem Dengan Metode Brute Force Menggunakan Graphic Processing Unit, *e-Proceeding of Engineering*, vol. 2, no. 1, pp. 1874-1883, 2015.

[13] S. A. Nene & S. K. Nayar, A simple algorithm for nearest neighbor search in high dimensions, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 989-1003, 1997, https://doi.org/10.1109/34.615448.

[14] I. Sutoyo, Penerapan Algoritma Nearest Neighbour untuk Menyelesaikan Travelling Salesman Problem, *Paradigma - Jurnal Komputer dan Informatika*, vol. 20, no. 1, pp. 101–106, 2018, https://doi.org/10.31294/p.v20i1.3155.

[15] F. J. Tjoea & S. Halim, Evaluasi Rute Pelayaran Kapal dengan Pendekatan Modified Minimum Spanning Tree, *Jurnal Titra*, vol. 11, no. 2, pp. 265-272, 2023.