Perbandingan Algoritma Brute Force dan Greedy dalam Optimasi Penjaluran Kabel Udara di Cisitu Indah, Kota Bandung

Muhammad Naufal Aulia - 13522074[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13522074@std.stei.itb.ac.id*

***Abstract*—Makalah ini membahas implementasi dan perbandingan dua algoritma dalam optimasi penjaluran kabel udara untuk menemukan pohon merentang minimum (Minimum Spanning Tree) di daerah Cisitu Indah. Algoritma yang diimplementasikan adalah algoritma brute force dan algoritma greedy. Makalah ini menyoroti pentingnya algoritma efisien dalam aplikasi praktis dan memberikan wawasan tentang bagaimana algoritma greedy dapat digunakan untuk menyelesaikan masalah optimasi jaringan yang kompleks. Melalui implementasi ini, akan didapat optimasi panjang kabel yang cukup untuk mendistribusikan listrik dengan keuntungan berupa hemat secara ekonomis dan kerapian/estetika visual pada lingkungan.**

Keywords—Kabel Udara, Optimasi, Algoritma Brute Force, Algoritma Greedy

I. PENDAHULUAN

Dalam era modern ini, listrik menjadi salah satu kebutuhan utama yang mendukung kehidupan sehari-hari manusia. Sebagai bagian integral dari infrastruktur kota, distribusi listrik memainkan peran krusial untuk menjamin terpenuhinya kebutuhan energi di antara masyarakat. Di Indonesia, permintaan akan pasokan listrik meningkat seiring dengan pertumbuhan populasi dan perkembangan industri, sehingga penyediaan infrastruktur listrik menjadi tantangan yang harus diatasi.

Distribusi listrik di daerah perkotaan memerlukan perencanaan yang cermat untuk memastikan penyaluran yang efisien. Salah satu metode distribusi listrik adalah melalui saluran kabel udara, yang ditopang oleh tiang listrik untuk menghantarkan energi di atas permukaan tanah. Meskipun kabel udara dapat memengaruhi estetika, ruang gerak, dan rentan terhadap cuaca, metode ini lebih ekonomis dalam pengembangan dan pemeliharaan, serta lebih mudah disesuaikan dengan topografi dan kondisi lingkungan sekitar. [1]

Di daerah Cisitu Indah, Kota Bandung, yang merupakan contoh mikro urban yang berkembang pesat, penggunaan kabel udara sebagai saluran distribusi listrik menjadi fokus penelitian. Daerah ini memiliki berbagai fungsi, mulai dari perumahan, bisnis, gedung pemerintahan, fasilitas pendidikan, hingga kos-kosan, yang semuanya menuntut peningkatan dan pemerataan distribusi listrik. Kabel udara di daerah ini menghubungkan gardu listrik dengan berbagai fasilitas untuk memastikan konektivitas yang memadai.

Pada makalah sebelumnya yang berjudul "Analisis Implementasi Graf pada Penjaluran Kabel Udara di Daerah Cisitu Indah, Kota Bandung," telah dibahas tentang optimasi penjaluran kabel udara menggunakan pohon merentang minimum. Dalam penelitian tersebut, pendekatan teori graf digunakan untuk mencapai efisiensi dalam penyaluran listrik.

Pada makalah kali ini, fokus penelitian adalah pada perbandingan antara algoritma Brute Force dan Greedy dalam optimasi penjaluran kabel udara di Cisitu Indah. Algoritma Brute Force, dengan mengevaluasi semua kemungkinan solusi, memberikan hasil yang paling optimal namun membutuhkan waktu komputasi yang lebih lama. Sementara itu, algoritma Greedy, yang mengambil keputusan lokal terbaik pada setiap langkah, memberikan solusi yang cepat meskipun tidak selalu optimal.

Dengan membandingkan kedua algoritma ini, diharapkan dapat ditemukan metode yang paling efektif dan efisien untuk penyaluran kabel udara. Analisis ini diharapkan mampu meningkatkan efisiensi dan keandalan distribusi listrik, yang pada akhirnya berdampak positif terhadap kesejahteraan dan produktivitas masyarakat di daerah tersebut.

II. LANDASAN TEORI

A. *Algoritma*

Algoritma adalah serangkaian langkah atau instruksi yang digunakan untuk menyelesaikan suatu masalah atau melakukan suatu tugas tertentu. Algoritma memastikan bahwa setiap langkah dapat diikuti secara konsisten untuk menghasilkan output yang diinginkan dari input yang diberikan.

B. *Algoritma Brute Force*

Algoritma Brute Force adalah pendekatan langsung (*straightforward*) untuk memecahkan suatu masalah. Algoritma ini mengandalkan eksplorasi semua kemungkinan

solusi yang ada hingga menemukan solusi yang tepat. Pendekatan ini biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi atau konsep yang terlibat dalam masalah tersebut.

Karakteristik dari Algoritma Brute Force antara lain [2]:

- Sederhana: Pendekatan ini sangat sederhana karena tidak memerlukan teknik khusus atau strategi kompleks.
- Langsung: Algoritma ini langsung mencari solusi dengan mencoba setiap kemungkinan yang ada.
- Jelas: Cara kerja algoritma ini sangat jelas dan mudah dipahami.
- Just do it! atau Just Solve it!: Prinsip dasar dari algoritma ini adalah mencoba semua solusi hingga menemukan solusi yang benar.

Walaupun mempunyai kelebihan karena dapat dipastikan menemukan solusi yang optimal, namun, karena eksplorasi semua kemungkinan, algoritma Brute Force sering kali membutuhkan waktu komputasi yang sangat lama, terutama untuk masalah dengan ruang solusi yang besar.

C. Algoritma Greedy

Algoritma Greedy adalah metode yang paling populer dan sederhana untuk memecahkan masalah optimasi. Masalah optimasi adalah masalah yang mencari solusi terbaik atau optimal dari sekian banyak kemungkinan solusi. Masalah optimasi dapat dibagi menjadi dua kategori utama:

1. Maksimasi (*maximization*): Mencari solusi yang memaksimalkan suatu nilai atau fungsi.
2. Minimasi (*minimization*): Mencari solusi yang meminimalkan suatu nilai atau fungsi

Karakteristik dari Algoritma Greedy antara lain:

- Optimum lokal: Algoritma Greedy membuat keputusan lokal terbaik pada setiap langkahnya dengan harapan bahwa keputusan ini akan menghasilkan solusi optimal secara keseluruhan.
- Sederhana: Pendekatan ini cukup sederhana dan mudah diimplementasikan.
- Efisien: Algoritma ini umumnya lebih efisien dalam hal waktu komputasi dibandingkan dengan algoritma Brute Force, meskipun tidak selalu menjamin solusi optimal.

Elemen penyusun Algoritma Greedy antara lain [3]:

- Himpunan kandidat: berisi kandidat yang akan dipilih pada setiap langkahnya.
- Himpunan solusi: berisi kandidat yang sudah dipilih.
- Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
- Fungsi seleksi: memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
- Fungsi kelayakan: memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
- Fungsi obyektif: memaksimumkan atau meminimumkan solusi yang diinginkan.

Algoritma Greedy bekerja dengan memilih opsi yang paling optimal pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang. Pendekatan ini dapat sangat efisien dan efektif untuk masalah tertentu, tetapi ada juga situasi di mana algoritma ini tidak memberikan solusi optimal sehingga algoritma ini mempunyai kelebihan dalam hal efisiensi namun belum menjamin optimalnya solusi yang dihasilkan.

D. Graf dan Terminologinya

Graf merupakan himpunan objek-objek yang terdiri atas simpul (*vertex*) dan sisi atau busur (*edge*) yang menjadi penghubung antara pasangan simpul. Graf digunakan untuk merepresentasikan hubungan atau keterkaitan antara objek-objek diskrit. Graf $G = (V, E)$ didefinisikan dengan [3]:

V = himpunan tidak kosong dari simpul (*vertices*)

$= \{v1, v2, \ldots, vn\}$

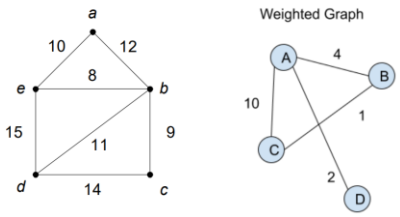
E = himpunan sisi (*edges*) yang menghubungkan dua simpul

$= \{e1, e2, \ldots, en\}$.

Ada beberapa terminologi penting dari graf yang memudahkan dalam pembahasan selanjutnya, yaitu sebagai berikut.

1. Lintasan (*Path*)
   Lintasan adalah urutan atau rute berupa sisi-sisi yang menghubungkan simpul awal v_0 ke simpul tujuan v_n. Panjang lintasan ialah jumlah sisi dalam lintasan tersebut, yakni *n.*
2. Sirkuit (*Circuit*)
   Sirkuit merupakan lintasan yang berawal dan berakhir pada simpul yang sama. Panjang sirkuit ialah jumlah sisi dalam sirkuit tersebut.
3. Keterhubungan (*Connected*)
   Dua simpul dikatakan terhubung Ketika terdapat lintasan yang menghubungkan setiap pasang simpul dalam graf, jika tidak, maka tidak terhubung.
4. Upagraf (*Subgraph*)
   Upagraf dari suatu graf adalah perolehan dari beberapa simpul dan sisi yang dihilangkan dari graf tersebut.
5. *Cut-Set*
   Cut-set dari suatu graf terhubung adalah himpunan sisi yang jika dibuang akan menyebabkan graf tersebut tidak terhubung dan terpisah menjadi dua bagian.
6. Graf Berbobot (*weighted graph*)
   Graf berbobot merupakan jenis graf yang setiap sisinya

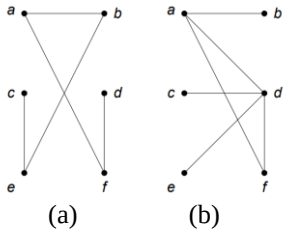
memiliki nilai atau harga, dinamakan bobot. Bobot ini merepresentasikan ukuran atau nilai yang terkait dengan hubungan antara dua simpul dari sisi tersebut, seperti representasi jarak fisik, waktu, biaya, dan hal lain yang relevan dengan konteks persoalan. Jenis graf ini yang umumnya dimanfaatkan untuk menyelesaikan persoalan optimasi suatu rute.



Gambar 1. Graf Berbobot
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

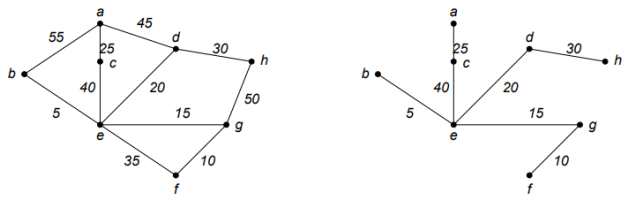
E. Pohon

Pohon merupakan jenis graf tak berarah yang terhubung dan tidak mengandung sirkuit. Tidak ada lintasan tertutup dalam pohon. [4]



Gambar 2. (a) Pohon, (b) Bukan Pohon
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

Pohon merentang dari suatu graf terhubung adalah upagraf yang menghubungkan pohon dan mengandung semua simpul di dalam graf tersebut. Sedangkan pohon merentang minimum (*minimum spanning tree*) merupakan pohon merentang dari graf berbobot yang memiliki bobot paling sedikit dari semua kemungkinan pohon merentangnya.



Gambar 3. Graf Berbobot dan Pohon Merentang Minimumnya
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf)

F. Kabel Udara

Kabel udara merupakan jenis kabel listrik yang terpasang di atas permukaan tanah. Kabel ini digunakan untuk mentransmisikan listrik ke berbagai tempat. Pada transmisi listrik dengan kabel udara, tiang listrik menjadi komponen penting untuk menopang posisi kabel di atas tanah.

Sesuai dengan konteks pembahasan, kabel udara pada daerah Cisitu Indah akan merepresentasikan graf yang nantinya akan dibuat pohon merentang. Dengan begitu, representasinya adalah sebagai berikut. [5]

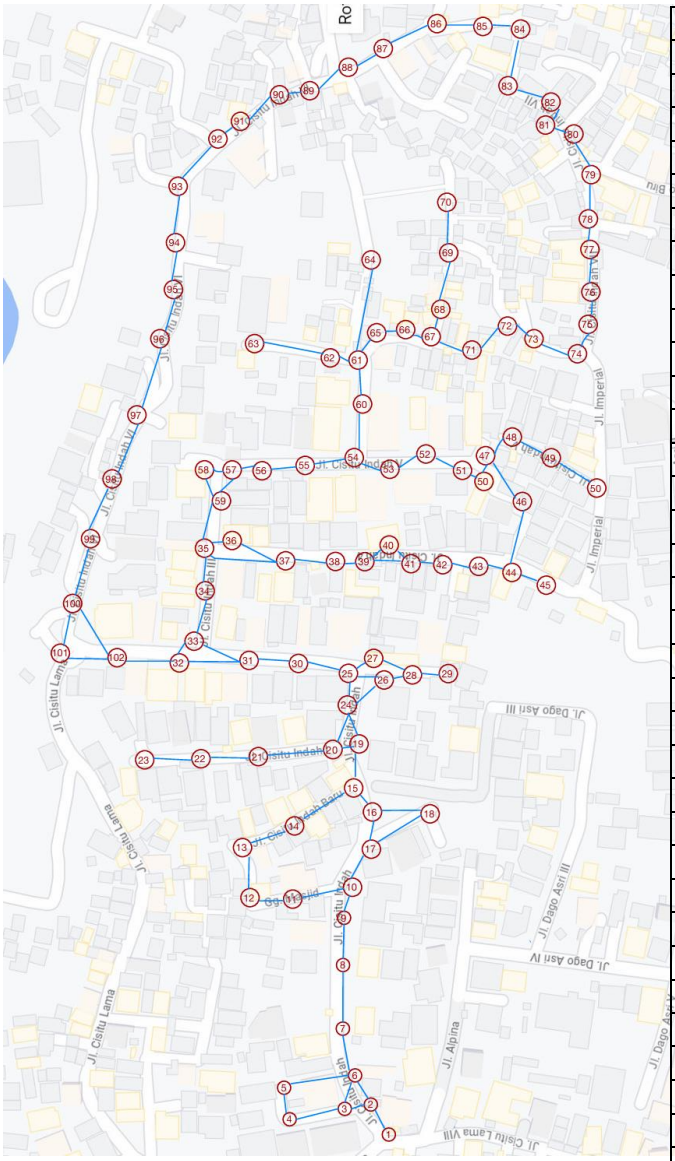
a. Simpul: tiang listrik di sepanjang jaringan mewakili simpul-simpul di mana terjadi pemberhentian atau perubahan arah rute jalur listrik.
b. Sisi: kabel udara yang terhubung dari dua tiang listrik mewakili sisi yang berbobot, dengan bobotnya adalah jarak antara dua tiang listrik tersebut.

III. PEMBAHASAN

A. Pemodelan Denah Lokasi

Pada pembahasan dalam makalah sebelumnya, penulis telah berhasil memetakan semua tiang listrik sebagai penyangga kabel udara yang tersebar di daerah Cisitu Indah. Setiap simpul merepresentasikan tiang listrik, sedangkan setiap sisi merepresentasikan kabel udara yang menghubungkan setiap tiang tersebut. Dalam pengumpulan data ini, penulis menggunakan metode observasi langsung untuk memetakan lokasi tiang listrik di lapangan dan melakukan pengukuran dengan memanfaatkan Google Maps untuk mendapatkan panjang sisi (kabel udara) dengan pembulatan dalam satuan meter.

Gambar di bawah ini, merupakan hasil graf awal dari kondisi aktual jaringan listrik dengan kabel udara di daerah Cisitu Indah ketika penulis melakukan observasi.



Gambar 4. Model Denah Lokasi dan Graf Hasil Observasi
(Sumber : Arsip Pribadi)

Berikut adalah tabel ukuran bobot pada graf. Bobot dalam konteks ini adalah hasil pengukuran jarak antara dua simpul, yakni antara tiang A dan B.

Tabel 1. Simpul (Tiang Listrik) dan Sisi (Jarak Antartiang)

(Tiang A, Tiang B)	Jarak Tiang (m)	(Tiang A, Tiang B)	Jarak Tiang (m)
(47, 50)	5	(3, 6)	19
(50, 51)	6	(15, 16)	19
(57, 58)	6	(24, 25)	19
(35, 36)	7	(40, 41)	19
(81, 82)	7	(53, 54)	19
(2, 3)	9	(2, 6)	20
(39, 40)	9	(9, 10)	20
(19, 20)	10	(67, 68)	20
(77, 78)	10	(75, 76)	20
(25, 27)	11	(71, 72)	21
(32, 33)	11	(16, 17)	22
(57, 59)	12	(15, 19)	22
(25, 26)	13	(41, 42)	22
(74, 75)	13	(80, 82)	22
(61, 62)	14	(87, 88)	22
(66, 67)	14	(89, 90)	22
(72, 73)	14	(11, 12)	23
(42, 43)	15	(24, 26)	23
(65, 66)	15	(43, 44)	23
(91, 92)	15	(51, 52)	23
(58, 59)	17	(55, 56)	23
(80, 81)	17	(19, 24)	24
(26, 28)	18	(52, 53)	24
(38, 39)	18	(1, 2)	25
(44, 45)	18	(8, 9)	25
(47, 48)	18	(56, 57)	25
(61, 65)	18	(27, 28)	26
(79, 80)	18	(60, 61)	26
(84, 85)	18	(6, 7)	27
(4, 5)	19	(20, 24)	27
(37, 38)	27	(101, 102)	34
(39, 41)	27	(16, 18)	35
(90, 91)	27	(22, 23)	35
(96, 95)	27	(46, 47)	35
(28, 29)	28	(92, 93)	35
(31, 33)	28	(100, 101)	35
(48, 49)	28	(21, 22)	36
(76, 77)	28	(3, 4)	37
(82, 83)	28	(16, 19)	37
(88, 89)	28	(68, 69)	37
(10, 11)	29	(31, 32)	38
(10, 17)	29	(5, 6)	39
(35, 59)	29	(7, 8)	39
(69, 70)	29	(83, 84)	39
(33, 34)	30	(32, 102)	39
(12, 13)	31	(27, 30)	40
(30, 31)	31	(44, 46)	41
(67, 71)	31	(100, 102)	41
(34, 35)	32	(98, 99)	42
(54, 60)	32	(17, 18)	44
(73, 74)	32	(36, 37)	44
(25, 30)	33	(14, 15)	45
(54, 55)	33	(35, 37)	49
(85, 86)	33	(20, 21)	50
(96, 97)	33	(97, 98)	50
(13, 14)	34	(62, 63)	53
(78, 79)	34	(61, 64)	64
(86, 87)	34		
(93, 94)	34		
(99, 100)	34		

Dari data tersebut didapatkan panjang total kabel udara adalah sebesar 3.093 meter.

B. Implementasi Brute Force

Algoritma Brute Force bekerja dengan mencoba semua kemungkinan kombinasi dari sisi yang ada kemudian memeriksa apakah kombinasi tersebut membentuk pohon merentang yang valid. Setelah memeriksa apakah kombinasi tersebut valid, akan dihitung total bobot dari sisi-sisi dalam kombinasi tersebut.

```
# Inisialisasi jumlah simpul
vertices = max(max(u, v) for u, v, w in edges) + 1
# Membuat daftar sisi
adj_list = {i: set() for i in range(vertices)}
for u, v, w in edges:
    adj_list[u].add(v)
    adj_list[v].add(u)

# Fungsi mengecek apakah subset membentuk pohon merentang
def is_spanning_tree(subset, num_vertices):
    if len(subset) != num_vertices - 1:
            return False
        # Membuat graf dari subset edge
        adj = {i: set() for i in range(num_vertices)}
        for u, v, w in subset:
            adj[u].add(v)
            adj[v].add(u)
        # Fungsi DFS untuk mengecek konektivitas
        def dfs(v, visited):
            visited.add(v)
            for neighbor in adj[v]:
                if neighbor not in visited:
                    dfs(neighbor, visited)
        visited = set()
        dfs(next(iter(adj)), visited)
        return len(visited) == num_vertices

# Fungsi untuk menghitung total bobot dari subset edge
def total_weight(subset):
    return sum(w for u, v, w in subset)
# Menghasilkan semua kombinasi edge sebanyak num_vertices-1
all_edges = list(edges)
min_weight = float('inf')
best_subset = None
for subset in combinations(all_edges, vertices - 1):
    if is_spanning_tree(subset, vertices):
        weight = total_weight(subset)
        if weight < min_weight:
           min_weight = weight
           best_subset = subset
```

Pada algoritma tersebut, kombinasi yang dihasilkan adalah sebanyak $C(E,V-1)$ di mana E adalah jumlah tepi dan V adalah jumlah simpul. Untuk setiap kombinasi ini, diperiksa apakah membentuk pohon merentang menggunakan DFS sehingga kompleksitas waktu untuk pemeriksaan ini adalah $O(V+E)$. Setelah itu, dihitung total bobot dengan kompleksitas $O(V-1)$. Maka, secara keseluruhan kompleksitas waktunya adalah $C(E,V-1)\cdot O(V+E)$ yang berarti akan sangat tidak efisien, dan benar saja berdasarkan percobaan dengan data pada tabel 1, ternyata belum dapat ditemukan solusi akibat waktu eksekusinya yang sangat lama.

C. Implementasi Greedy

Strategi Greedy yang diimplementasikan pertama adalah dengan memilih sisi berbobot terkecil yang menghubungkan pohon yang sedang dibangun dengan simpul yang belum terhubung. Strategi ini juga dikenal dengan Algoritma Prim. Berikut adalah pemetaan masalah dalam strategi Greedy.

- Himpunan Kandidat: berisi semua sisi yang menghubungkan simpul yang sudah ada di dalam MST (*Minimum Spanning Tree*) dengan simpul di luar MST.
- Himpunan Solusi: kumpulan sisi yang sudah dipilih untuk dimasukkan ke dalam MST.
- Fungsi Solusi: menentukan apakah himpunan sisi yang dipilih membentuk MST yang valid, yakni jika sudah mencakup semua simpul dalam graf.
- Fungsi Seleksi: memilih tepi dengan bobot terkecil dari himpunan kandidat.
- Fungsi Kelayakan: memeriksa apakah tepi yang dipilih dari himpunan kandidat belum ada dalam MST dan dapat dimasukkan ke dalam himpunan solusi tanpa membentuk siklus.
- Fungsi Objektif: meminimumkan bobot total dari MST.

Untuk menerapkan pengurutan bobot sisi terkecil, digunakan heap sebagai *priority queue* yang akan selalu memilih sisi dengan bobot terkecil sebagai prioritas. Kemudian dilakukan iterasi berulang sampai semua simpul sudah berada di MST.

```
def greedy_mst(vertices, edges):
    # Inisialisasi struktur data
    adj = {i: [] for i in range(vertices)}
    for u, v, w in edges:
        adj[u].append((w, v))
        adj[v].append((w, u))

    # Fungsi untuk menambahkan tepi ke dalam heap
    def add_edges(node):
        for weight, to in adj[node]:
            if to not in mst_set:
                heapq.heappush(edge_list, (weight, to, node))
```

```
    mst_set = set()
    edge_list = []
    total_weight = 0

    # Hasil akhir MST
    result = []

    # Mulai dari simpul pertama
    start_node = 1
    mst_set.add(start_node)
    add_edges(start_node)

    while len(mst_set) < vertices and edge_list:
        weight, node, from_node = heapq.heappop(edge_list)
        if node not in mst_set:
            mst_set.add(node)
            total_weight += weight
            result.append((from_node, node, weight))
            add_edges(node)
    return result, total_weight

# Inisialisasi jumlah simpul
vertices = max(max(u, v) for u, v, w in edges) + 1
mst, total_weight = greedy_mst(vertices, edges)
```

Berikut adalah solusi yang dihasilkan berupa keluaran sisi mana saja yang terpilih oleh strategi Greedy tersebut, total jarak kabel setelah optimasi menggunakan algoritma ini dalam satuan meter, dan waktu eksekusinya dalam satuan detik.

```
1 -- 2 == 25m
2 -- 3 == 9m
3 -- 6 == 19m
6 -- 7 == 27m
3 -- 4 == 37m
4 -- 5 == 19m
7 -- 8 == 39m
8 -- 9 == 25m
9 -- 10 == 20m
10 -- 11 == 29m
11 -- 12 == 23m
10 -- 17 == 29m
17 -- 16 == 22m
16 -- 15 == 19m
15 -- 19 == 22m
19 -- 20 == 10m
19 -- 24 == 24m
24 -- 25 == 19m
25 -- 27 == 11m
25 -- 26 == 13m
26 -- 28 == 18m
28 -- 29 == 28m
12 -- 13 == 31m
25 -- 30 == 33m
30 -- 31 == 31m
31 -- 33 == 28m
33 -- 32 == 11m
33 -- 34 == 30m
34 -- 35 == 32m
35 -- 36 == 7m
35 -- 59 == 29m
59 -- 57 == 12m
57 -- 58 == 6m
57 -- 56 == 25m
56 -- 55 == 23m
55 -- 54 == 33m
54 -- 53 == 19m
53 -- 52 == 24m
52 -- 51 == 23m
51 -- 50 == 6m
50 -- 47 == 5m
47 -- 48 == 18m
48 -- 49 == 28m
54 -- 60 == 32m
60 -- 61 == 26m
61 -- 62 == 14m
61 -- 65 == 18m
65 -- 66 == 15m
66 -- 67 == 14m
67 -- 68 == 20m
67 -- 71 == 31m
71 -- 72 == 21m
72 -- 73 == 14m
73 -- 74 == 32m
74 -- 75 == 13m
75 -- 76 == 20m
76 -- 77 == 28m
77 -- 78 == 10m
13 -- 14 == 34m
78 -- 79 == 34m
79 -- 80 == 18m
80 -- 81 == 17m
81 -- 82 == 7m
82 -- 83 == 28m
16 -- 18 == 35m
47 -- 46 == 35m
68 -- 69 == 37m
69 -- 70 == 29m
83 -- 84 == 39m
84 -- 85 == 18m
85 -- 86 == 33m
86 -- 87 == 34m
87 -- 88 == 22m
88 -- 89 == 28m
89 -- 90 == 22m
90 -- 91 == 27m
91 -- 92 == 15m
92 -- 93 == 35m
93 -- 94 == 34m
32 -- 102 == 39m
102 -- 101 == 34m
101 -- 100 == 35m
100 -- 99 == 34m
46 -- 44 == 41m
44 -- 45 == 18m
44 -- 43 == 23m
43 -- 42 == 15m
42 -- 41 == 22m
41 -- 40 == 19m
40 -- 39 == 9m
39 -- 38 == 18m
38 -- 37 == 27m
99 -- 98 == 42m
20 -- 21 == 50m
21 -- 22 == 36m
22 -- 23 == 35m
98 -- 97 == 50m
97 -- 96 == 33m
96 -- 95 == 27m
62 -- 63 == 53m
61 -- 64 == 64m
Total jarak kabel: 2554m
Waktu eksekusi: 0.001007080078125 detik
```

Gambar 5. Hasil Keluaran Program Sederhana Algoritma Greedy
(Sumber : Arsip Pribadi)

Pada algoritma tersebut, untuk membuat list adjacency memerlukan waktu $O(E)$, di mana E adalah jumlah tepi dalam graf. Dalam mengelola sisi-sisi kandidat yang menggunakan heap, setiap operasi penambahan atau penghapusan dalam heap memerlukan waktu $O(\log E)$, sedangkan dalam kasus terburuk, setiap simpul dan tepi diproses satu kali, sehingga kompleksitas untuk operasi heap adalah $O((V+E)\log E)$ di mana V adalah jumlah simpul. Maka total kompleksitas waktu untuk algoritma greedy tersebut adalah $O((V+E)\log E)$.

Terlihat dari hasil keluaran di atas, algoritma Greedy mampu menyelesaikan kasus persoalan minimasi penjaluran kabel udara dengan mudah dan relatif cepat, yakni hanya memakan waktu selama 0,001 detik saja. Solusi yang didapatkan pun mampu meminimasi kabel yang dibutuhkan menjadi hanya 2.554 meter dibanding panjang semula yakni 3.093 meter, memberikan optimasi sebesar 17,5%.

IV. KESIMPULAN

Dalam studi kasus optimasi penjaluran kabel udara di daerah Cisitu Indah, telah diimplementasikan dan dibandingkan dua algoritma berbeda: algoritma brute force dan algoritma greedy. Kedua algoritma ini diterapkan untuk menemukan pohon merentang minimum (Minimum Spanning Tree, MST) yang akan meminimalkan total panjang kabel yang dibutuhkan untuk menghubungkan semua tiang listrik. Hasil dari implementasi algoritma brute force menunjukkan bahwa algoritma ini tidak efisien dalam konteks penjaluran kabel udara yang melibatkan banyak simpul dan sisi. Sebaliknya, algoritma greedy berhasil memberikan solusi yang optimal dengan waktu eksekusi yang cepat. Pada kasus ini, algoritma greedy mampu menghasilkan pohon merentang minimum dengan total panjang kabel sebesar 2.554 meter dalam waktu 0.001 detik, menjadikannya lebih cocok dan efisien untuk diaplikasikan pada persoalan representasi graf dengan jumlah simpul dan sisi yang besar.

Pemanfaatan algoritma ini dalam menentukan pohon merentang minimum terbukti dapat mengurangi jarak kabel udara yang dibutuhkan, dari yang semula memiliki panjang total 3.093 meter, menjadi hanya 2.554 meter (17,5% lebih baik). Algoritma ini tidak hanya memastikan bahwa total panjang kabel yang dibutuhkan adalah yang terkecil, tetapi juga dapat diterapkan dengan cepat dan efektif pada kasus-kasus serupa di lapangan.

REFERENSI

[1] Rania, A. Tinjauan Terhadap penyaluran listrik di Indonesia: Menilik Peran saluran Udara Dan Saluran Kabel. Diakses pada 7 Desember 2023 dari https://ftmm.unair.ac.id/tinjauan-terhadap-penyaluran-listrik-di-indonesia-menilik-peran-saluran-udara-dan-saluran-kabel/.

[2] Munir, R. 2022. Algoritma Brute Force (Bag.1). Diakses pada 1 Juni 2024 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[3] Munir, R. 2021. Algoritma Greedy (Bag.1). Diakses pada 1 Juni 2024 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[4] Munir, R. 2023. Graf (Bag.1). Diakses pada pada 1 Juni 2024 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf.

[5] Munir, R. 2023. Pohon (Bag.1). Diakses pada pada 1 Juni 2024 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/22-Pohon-Bag1-2023.pdf.

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, 1 Juni 2024

Muhammad Naufal Aulia 13522074