ANALISIS ROUTING PADA APLIKASI PEMANGGILAN DARURAT MENGGUNAKAN ALGORITMA HILL CLIMBING DAN GREEDY

Andi Sitti Syathirah[1], Purba Daru Kusuma[2], Casi Setia Ningsih[3]

[1,2,3] Universitas Telkom, Bandung

raraiyy@gmail.com, purbodaru@telkomuniversity.ac.id, setiacasie@telkomuniversity.ac.id

Abstrak

Panggilan darurat di Indonesia masih menerapkan teknologi panggilan satelit untuk memanggil lembaga darurat. Indonesia sudah memiliki nomor darurat universal, namun pelaksanaannya masih belum merata di semua kota/kabupaten, di mana nomor darurat 112. Sebagian besar masyarakat Indonesia jarang mengetahui nomor ini karena informasi dari nomor ini kurang dipublikasikan dan masyarakat cenderung menggunakan internet lebih dari menggunakan panggilan konvensional.

Dengan memanfaatkan aplikasi berbasis internet, penulis dapat membangun sebuah aplikasi panggilan yang dipadukan dengan *Google Maps* untuk memberikan tampilan gambar dari peta digital. Aplikasi ini menyediakan rute otomatis dari lokasi penelepon ke lokasi agensi yang dipanggil secara langsung.

Berdasarkan hasil pengujian, Algoritma *Hill Climbing* dan *Greedy* dapat diimplementasikan dengan baik dengan menggunakan bobot SAW jarak 30% dan durasi 70% yang memiliki nilai *range* total antar *node* adalah 0,022. Algoritma *Greedy* memberikan hasil terbaik berdasarkan waktu respon. Dengan *response time* rata-rata 0,912, sedangkan Algoritma *Hill Climbing* 1,59. Hasilnya, rute terbaik pada penelitian ini yang membandingkan *Hill Climbing* dan algoritma *Greedy* adalah Algoritma *Greedy*.

Kata Kunci: Rute Terbaik, Algoritma *Hill Climbing*, Algoritma *Greedy*, *Google Maps*

Abstract

Emergency calls in Indonesia are still implementing satellite call technology to call emergency agencies. Indonesia already has a universal emergency number, but its implementation is still uneven in all cities/districts, where the emergency number is 112. Most Indonesians rarely know this number because information from this number is less publicized and people tend to use the internet more than using conventional calls.

By utilizing an internet-based application, the author can build a call application that is combined with Google Maps to provide an image display from a digital map. This application provides an automatic route from the location of the caller to the location of the agency being called directly.

Based on the test results, the Hill Climbing and Greedy Algorithm can be implemented properly by using SAW weight distance of 30% and duration of 70% which has a total range value between nodes is 0.022. Greedy Algorithm gives the best result based on response time. With an average response time of 0.912, while the Hill Climbing Algorithm is 1.59. As a result, the best route in this study that compares Hill Climbing and the Greedy algorithm is the Greedy Algorithm.

Keywords: Best Route, Hill Climbing Algorithm, Greedy Algorithm, Google Maps

I. PENDAHULUAN

Telepon merupakan alat komunikasi jarak jauh yang menggunakan suara. Salah satu implementasi penggunaan telepon adalah panggilan darurat ke instansi (polisi, pemadam kebakaran, ambulans, dan lain-lain). *Handphone* merupakan perangkat genggam yang dimiliki oleh sebagian besar masyarakat di dunia untuk keperluan komunikasi, pekerjaan, dan lain-lain. Perkembangan telepon seluler tidak lepas dari perkembangan teknologi yang ada, termasuk internet. Internet adalah singkatan dari Interconnected Network, dimana Internet adalah sistem jaringan komputer yang memungkinkan kita untuk terhubung dengan orang-orang di seluruh dunia menggunakan satu perangkat.

Menurut laporan terbaru *We Are Social*, pada tahun 2020 disebutkan terdapat 175,4 juta pengguna internet di Indonesia, dengan peningkatan pengguna di Indonesia sebesar 17% dari tahun sebelumnya [1]. Internet banyak digunakan untuk kebutuhan sehari-hari, seperti kemudahan mengakses informasi, konektivitas, komunikasi dan sharing, dan masih banyak lagi aktivitas yang lebih mudah dilakukan dengan internet.

Saat ini panggilan darurat di Indonesia masih menggunakan panggilan konvensional, padahal saat ini

masyarakat umum lebih banyak menggunakan internet untuk segala kebutuhannya. Untuk melakukan panggilan darurat, masyarakat harus mengetahui nomor agen yang akan dihubungi. Dalam hal ini, masyarakat jarang mengetahui informasi tentang nomor darurat. Selanjutnya penelepon harus menjelaskan lokasi terjadinya keadaan darurat yang dapat memakan waktu lama jika agen yang dihubungi tidak mengetahui alamat yang tertera oleh penelepon. Dalam menerima panggilan, setelah instansi menerima panggilan darurat, biasanya informasi dari panggilan darurat ini akan disampaikan kepada petugas yang bertugas melalui radio atau orang lain untuk menuju ke tempat penelepon [2].

Jurnal ini bertujuan untuk mengembangkan aplikasi mobile untuk aplikasi panggilan darurat. Aplikasi ini berfokus pada pencarian rute terbaik menggunakan algoritma *Hill Climbing* dan algoritma *Greedy*. Oleh karena itu, kinerja kedua algoritma dibandingkan dalam konteks pencarian rute terbaik dari lokasi pengguna ke instansi terkait menggunakan parameter nilai algoritma dan durasi waktu perjalanan.

I. KERANGKA TEORI

1. Perencanaan Jalur Kendaraan Berpanduan Otomatis Berdasarkan Peningkatan Algoritma A Star (Chunbao Wang, Lin Wang, Jian Qin, Zhengzhi Wu, Lihong Duan, Zhongqiu Li, 2015).

   Pada jurnal ini merupakan perencanaan jalur dalam kasus sistem logistik otomatis yang merupakan kendaraan dipandu otomatis (AGV). Peran AGV di sini adalah untuk meningkatkan efisiensi perusahaan otomasi logistik. Penulis menggunakan algoritma A-star yang ditingkatkan untuk mencari jalur waktu terpendek secara efektif dan menghindari tabrakan. Juga, algoritma menghasilkan k kali jalur yang lebih pendek dengan menghapus tepi. Hasil dari penelitian ini adalah perbaikan algoritma A star lebih smooth, lebih kondusif untuk menjalankan AGV secara otomatis [3].
2. Sistem pengantaran makanan memanfaatkan kendaraan menggunakan *Geographical Information System (GIS) and A Star Algorithm* (B Siregar, D Gunawan, U Andayani, Elita Sari, F Fahmi, 2016).

   Penelitian ini melakukan penentuan jalur terpendek dalam hal pelacakan pergerakan kendaraan pengantar makanan. Penulis menggunakan Sistem Informasi Geografis (SIG) dan Algoritma A Star untuk mencari yang terpendek. Peran GIS di sini adalah untuk meningkatkan manajemen transportasi dan pelacakan kendaraan sedangkan algoritma bintang A adalah untuk menemukan pengiriman rute yang optimal dengan menentukan node ke node berikutnya menggunakan biaya jarak aktual dan heuristik. Selain itu, penelitian ini menggunakan parameter kemacetan lalu lintas. Hasil dari pengujian yang telah dilakukan dijalankan dengan baik dan didapatkan jalur terpendek. juga, pengguna dapat melacak lokasi kendaraan saat memesan makanan [4].

3. Traveling Salesman Problem (TSP) adalah penyelesaian masalah kompleks dalam pencarian rute secara bertahap. Metode ini menjelaskan tentang bagaimana menentukan rute terpendek mulai dari kota awal ke kota tujuan dan setiap node hanya bisa dikunjungi satu kali (Suyanto, 2014).

   TSP diusulkan pada tahun 1800 oleh William Rowan Hamilton dan Thomas Penyngton K, sedangkan bentuk umum TSP pertama kali dipelajari oleh matematikawan pada tahun 1930. TSP adalah himpunan kota dan biaya perjalanan (atau jarak) yang diberikan antara setiap pasangan kota yang digunakan untuk mencari cara. Kunjungan terbaik ke semua kota dan kembali ke titik awal untuk meminimalkan biaya atau jarak perjalanan (Davendra, 2010) [5].
4. Metode *Hill Climbing* adalah metode yang digunakan dalam menyelesaikan masalah pencarian jarak terdekat (Rich et al., 1991 dalam Russel dan Norvig, 2003).

   Hill Climbing merupakan teknik optimasi untuk mencari solusi masalah dari input data menggunakan algoritma. Cara kerja metode ini adalah dengan menentukan langkah selanjutnya dengan radius node terdekat. Proses pengujian dilakukan dengan menggunakan fungsi heuristik. Langkah selanjutnya adalah tergantung pada umpan balik saat proses pengujian. Penggunaan fungsi heuristik menunjukkan seberapa baik kita menebak kemungkinan yang akan terjadi selanjutnya [6].

5. Algoritma *Greedy* telah dikembangkan untuk sejumlah besar masalah kombinatorial. Untuk banyak dari algoritma *Greedy* ini, hasil analisis kasus terburuk yang elegan telah terkandung. Algoritma ini mudah dipahami dan membutuhkan waktu yang singkat untuk dijalankan, sedangkan akurasi dari algoritma *Greedy* terkadang tidak dapat diterima [7].

II. METODE PENELITIAN

A. Rute Terpendek

Rute terpendek adalah jalur yang diperlukan untuk mencapai tujuan dari lokasi saat ini ke tujuan dengan menghitung biaya minimum (jarak) untuk semua node dalam satu rute. Masalah seperti ini biasanya disajikan

dalam bentuk graf, dimana state yang berhubungan dengan ruang lingkup pencarian direpresentasikan dengan vertex dan transisi yang terjadi digambarkan dalam bentuk edge. Graf itu sendiri merupakan struktur diskrit yang terdiri dari simpul (V) yang merupakan himpunan tidak kosong, dan sisi (E) adalah himpunan sisi yang menghubungkan simpul-simpul tersebut [8].

Kata terpendek dari Shortest Route dapat diartikan sebagai bagaimana proses meminimalkan biaya (bobot) yang digunakan dalam suatu jalur graf. Beberapa masalah jalur terpendek yang biasa digunakan adalah jalur terpendek antara dua node, jalur terpendek antara semua pasangan node, jalur terpendek dari node tertentu ke node lain, jalur terpendek dari dua node yang melewati node tertentu [9].

B. Algoritm*a Hill Climbing*

Ada dua macam algoritma *Hill Climbing*. Yang pertama adalah *Simple Hill Climbing* yang menggunakan fungsi heuristik (fx) yang mengukur jarak dari x node ke tujuan. Teknik *Hill Climbing* ini dapat digunakan untuk memecahkan suatu masalah yang memiliki banyak alternatif solusi dan memilih solusi yang terbaik. Yang kedua adalah *Steepest Ascent Hill Climbing* yang mirip dengan *Simple Hill Climbing*. Perbedaannya terletak pada langkah yang tidak dimulai dari sisi kiri, melainkan dari biaya heuristik terbaik.

Algoritma *Hill Climbing* mengonfigurasi bahwa setiap konfigurasi diberi nilai yang disebut biaya. Yang didefinisikan sebagai fungsi biaya. Suka,

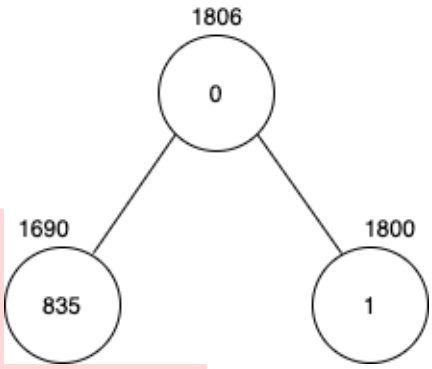
$$f : X \rightarrow R \quad (1)$$

Untuk setiap konfigurasi x X himpunan tetangga (x) didefinisikan. Tujuan dari pencarian ini adalah untuk menentukan nilai biaya terkecil [10]. Dalam hal ini, kami menyimpulkan langkah demi langkah Algoritma Hill Climbing, sebagai berikut:

- Menentukan biaya heuristik di setiap node menggunakan jarak garis lurus. Dimana metode ini mengukur jarak untuk setiap node.
- Membandingkan node saat ini dan node berikutnya (tetangga), jika biaya heuristik node berikutnya kurang dari node saat ini, itu akan menjadi langkah berikutnya.
- Ketika jalan buntu terjadi, algoritma Hill Climbing akan mengambil Back-Track atau jalur ke node sebelumnya dan membandingkan lagi dengan node saat ini.
- Poin ke-2 dan ke-3 diulangi sampai ditemukan rute terbaik dari awal sampai tujuan.

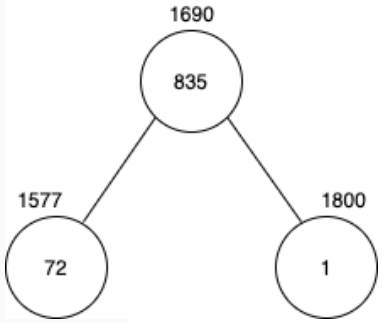
Algoritma *Hill Climbing* menggunakan *Left-Hand Rules*, algoritma melihat jalur kiri terlebih dahulu.

Langkah pertama, lihat gambar di bawah ini:



Untuk node 0, ada 2 tetangga. Node 835 dan node 1. Karena algoritma *Hill Climbing* menggunakan aturan tangan kiri, maka kita dapat melihat heusristik dari node ke-835. Biaya heuristik dari 0 node adalah 1806 dan biaya heuristik dari node 835 adalah 1690. Jelas bahwa 1690 lebih rendah dari 1806, jadi node berikutnya adalah node 835. Kita dapat menghindari simpul pertama.

Langkah kedua :



Sama seperti langkah sebelumnya, ada 2 tetangga untuk node 835. Yaitu node 72 dan node 1. Ingat aturan tangan kiri. Karena node 72 adalah sisi kiri, mari kita lihat biaya heuristik dari node 72. Apakah lebih rendah dari node 835 atau tidak. Jika lebih rendah, maka kita dapat memilih node 72 sebagai node berikutnya.

Biaya heuristik node 2 adalah 1577, sedangkan biaya heuristik node pertama adalah 1800. Apakah 1577 lebih rendah dari 1690? Ya itu. Jadi, kita dapat memilih node ke-72 sebagai node berikutnya dan mengabaikan node 1.

Langkah ketiga diulangi langkah sebelumnya sampai node tujuan.

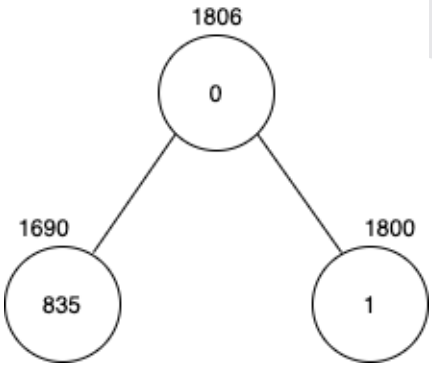
C. *Algoritma Greedy*

Algoritma *Greedy* membentuk solusi langkah demi langkah. Karena terlalu banyak pilihan yang harus dipertimbangkan pada setiap langkah, untuk menghasilkan keputusan terbaik, hanya saja satu keputusan telah diambil, tidak dapat diubah lagi pada langkah berikutnya [11].

Dalam algoritma *Greedy*, kami menyimpulkan bahwa langkahnya mudah dipahami, sebagai berikut:

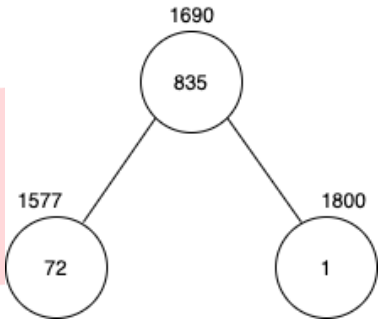
- Menentukan biaya heuristik di setiap node menggunakan jarak garis lurus. Dimana metode ini mengukur jarak untuk setiap node.
- Membandingkan node saat ini dan node berikutnya (tetangga), jika biaya heuristik node berikutnya kurang dari node saat ini, itu akan menjadi langkah berikutnya. Karena tidak ada *back-track* (langkah kembali) dalam algoritma ini, maka titik 2 diulangi sampai ditemukan rute terbaik dari awal sampai tujuan.

Berbeda dengan algoritma Hill Climbing, tidak ada aturan tangan kiri dalam algoritma Greedy. Algoritma ini membandingkan semua node tetangga dalam radius. Jika node berikutnya lebih rendah dari node saat ini, maka akan dipilih untuk node saat ini berikutnya. Sebagai contoh lihat gambar di bawah ini.



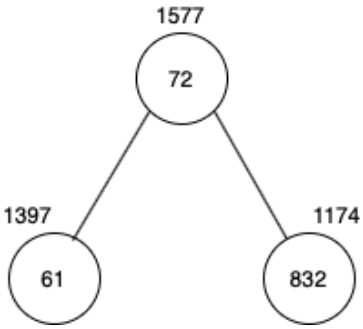
Untuk 0 node, ada 2 tetangga. Node ke-835 dan node ke-1. Algoritma Greedy membandingkan keduanya dan memilih node yang nilai biaya heuristiknya lebih kecil dari node saat ini. jadi kita bisa melihat heusristik dari node ke-835. Biaya heuristik dari 0 node adalah 1806 dan biaya heuristik dari 835 node adalah 1690. Jelas bahwa 1690 lebih rendah dari 1806, jadi node berikutnya adalah 835 node. Kita dapat menghindari simpul pertama.

Langkah selanjutnya, mari kita lihat lagi pada gambar di bawah ini:



Sama seperti langkah sebelumnya, ada 2 tetangga untuk node 835. Algoritma Greedy membandingkan lagi antara node 72 dan node 1. Biaya heuristik node 72 adalah 1577, sedangkan biaya heuristik node pertama adalah 1800. Apakah 1577 lebih rendah dari 1690? Ya itu. Jadi, kita dapat memilih node 72 sebagai node berikutnya dan mengabaikan node 1.

Langkah selanjutnya, mari kita lihat lagi pada gambar di bawah ini:



Ada dua node yang terhubung ke node 72, yang pertama adalah node 61 dan yang kedua adalah node 832. kita dapat melihat biaya heuristik antara node 61 dan node 832 adalah 1397 dan 1174. Keduanya memiliki biaya heuristik lebih rendah dari node 72. Tetapi algoritma Greedy memilih yang terbaik yang memiliki biaya heuristik terendah. Jadi, untuk langkah selanjutnya, kami memilih node 832 dengan biaya heuristik os 1174.

Langkah selanjutnya akan seperti itu, mengulangi cara yang sama sampai node tujuan.

III. DESAIN SISTEM DAN GAMBARAN UMUM

A. Gambaran Umum Sistem

Sistem yang akan dirancang pada aplikasi panggilan darurat ini adalah mencari rute terbaik menggunakan algoritma Hill Climbing dan Greedy, kemudian membandingkan kedua algoritma tersebut dan menentukan algoritma mana yang terbaik untuk mencari rute terbaik dengan memperhatikan jarak dan waktu.

Aplikasi yang dibuat adalah untuk mencari rute terbaik menuju instansi seperti kantor polisi, rumah sakit dan pemadam kebakaran di wilayah Kabupaten Bandung, Indonesia. Kedua algoritma di atas digunakan untuk membandingkan algoritma mana yang memberikan rute terpendek kepada pengguna. Saat pengguna menggunakan aplikasi dan mengaktifkan GPS di Smartphonenya, pengguna hanya perlu mencari lokasi instansi yang akan dituju sehingga sistem dapat segera menemukan rute terbaik yang akan dilalui oleh pengguna.

Karakteristik pengguna dalam menggunakan aplikasi adalah mereka yang dalam keadaan darurat dan ingin mencari rute terbaik menuju lokasi kantor polisi, rumah sakit dan/atau pemadam kebakaran. Pengguna mengaktifkan mode GPS pada aplikasi sistem dan mengirimkan lokasi saat ini.

Ketika pengguna telah menentukan lokasi tujuan, sistem akan mencari jalur terpendek atau tercepat yang akan dilalui oleh pengguna. Dalam hal ini pengguna tidak perlu memikirkan kemacetan yang terjadi, karena aplikasi telah menghitung kemacetan yang akan dilalui oleh pengguna. Aplikasi ini juga memiliki fitur panggilan darurat, dimana fitur ini dapat menelepon langsung ke instansi terkait.

B. Alur Sistem

Untuk lebih memahami alur sistem dalam mencari rute terbaik, berikut adalah tahapan alur sistem.

Pengguna mendaftar pada aplikasi dengan memasukkan data pribadi yang diperlukan, seperti Nomor Identitas, Nama Lengkap, Nomor Telepon, Email dan Kata Sandi.

1. Setelah registrasi berhasil, pengguna dapat masuk ke aplikasi menggunakan data yang telah dimasukkan sebelumnya.
2. Untuk melakukan fitur panggilan, pengguna diharapkan mengizinkan aplikasi untuk mengakses lokasi dan kontak perangkat.
3. Pengguna dapat menggunakan fitur menelepon, jika bingung dapat mengklik tombol bantuan untuk mengetahui cara menggunakan aplikasi.
4. Pengguna menekan tombol panggil, maka sistem akan mengambil lokasi pengguna dan akan dikirimkan ke instansi terkait.
5. Setelah pengguna menelepon dan lokasi dikirim, agensi yang dipanggil akan mendapatkan panggilan dan lokasi dari penelepon.
6. Agensi menjawab panggilan dan mendapatkan lokasi penelepon. Sistem akan memproses algoritma penentuan rute dengan menjadikan lokasi agen sebagai node awal dan lokasi pemanggil sebagai node tujuan.
7. Setelah proses algoritma selesai, hasil penentuan rute akan dikeluarkan berupa gambar menggunakan Google Maps dengan rute yang sudah didapatkan pada proses algoritma sebelumnya.

C. *Data Requirements*

Dalam pembuatan aplikasi ini, ada beberapa data yang dibutuhkan sebagai berikut

• Data Lokasi Instansi

Data lokasi instansi diperlukan sebagai tujuan dalam proses algoritma. Untuk menggunakan data lokasi untuk kebutuhan algoritma, data lokasi berupa garis lintang dan garis bujur. Berikut ini adalah lokasi instansi:

TABLE I. DATA LOKASI INSTANSI

No	Type Agency	Agency Name	Latitude	Longitude
1.	Police	Polrestabes Bandung	-6.913919331147 92	107.610500254 78556
2.	Hostpital	Ali Ihsan	-7.007741026496 179	107.622853981 7707
3	Firefighter Department	Dinas Kebakaran Kota Bandung	-6.940491740523 8855	107.672534410 63059

- Data Lokasi Pengguna

Untuk menggunakan aplikasi ini, pengguna dianjurkan untuk mengaktifkan GPS pada telepon genggam *(Smartphone).*

- Data Lokasi Titik

Sama seperti lokasi instance, lokasi node diperlukan dalam hal lintang dan bujur. Simpul di sini adalah lokasi pertigaan dan simpang di wilayah Kabupaten Bandung. Di sini kita menggunakan jalan ring 1 dan ring 2. Dimana ring 1 merupakan jalan utama atau jalan yang sering digunakan untuk lintas provinsi dan ring 2 merupakan jalan yang sering digunakan dalam kota. Berikut adalah data node lintang-bujur yang digunakan dalam aplikasi ini:

TABLE II. DATA LOKASI TITIK

No	Latitude	Longitude
1.	-6.972155430431573	107.63355639562371
2.	-6.9725440852050955	107.63404130699072
3	6.972822566764564	107.6361690705928
4.	6.967833757459738	107.6345979024971
5.	6.9694526888995005	107.63700865268265
6.	6.968378417066985	107.63727269250158
7.	6.966245742526659	107.63481445664898
8.	6.965109882535197	107.6357325121516
9.	6.965407884088816	107.63795585983732
10.	6.96192962029697	107.63853700033711

- Data Parameter

Untuk menjalankan algoritma pencarian rute, diperlukan data seperti biaya aktual dan heuristik. Pada aplikasi ini, untuk mendapatkan nilai heuristik digunakan *library geo-lib-distance*, dimana fungsi ini memberikan nilai jarak garis lurus antara satu lokasi dengan lokasi lainnya dengan memasukkan garis lintang bujur lokasi yang akan dihitung. Contoh bisa dilihat pada gambar 1.

```
getPreciseDistance(
  { latitude: 20.0504188, longitude: 64.4139099 },
  { latitude: 51.528308, longitude: -0.3817765 }
);
```

Gambar 1. Contoh Penggunaan Geo-Lib.

Untuk nilai parameter lainnya, yang merupakan biaya aktual dan durasi perjalanan, digunakan Google Distance Matrix API. Sama seperti geo-lib, untuk mendapatkan nilai dan durasi sebenarnya, Anda perlu memasukkan garis lintang bujur untuk lokasi yang ingin Anda cari nilainya. Contoh penggunaan API ini dapat dilihat pada Gambar 2.


```
{
  “destination_addresses” : [
    “Jl. Raya Dayeuhkolot No.325, Citeureup, Kec. Dayeuhkolot, Bandung, Jawa Barat 40257, Indonesia”
  ],
  “origin_addresses” : [
    “Masjid Subarul Ma’mur, Sukapura, Kec. Dayeuhkolot, Bandung, Jawa Barat, Indonesia”
  ],
  “rows” : [
    {
      “elements” : [
        {
          “distance” : {
            “text” : “1.6 mi”,
            “value” : 2587
          },
          “duration” : {
            “text” : “7 mins”,
            “value” : 418
          },
          “status” : “OK”
        }
      ]
    }
  ],
  “status” : “OK”
}
```


Gambar 2. Contoh penggunaan *Distance Matrix*

Seperti gambar di atas ada dua parameter yang digunakan. Pertama adalah jarak dan kedua adalah durasi. Di dalam parameter juga kami memiliki 2 kueri, teks dan nilainya. Dalam aplikasi ini kami menggunakan nilai untuk mendapatkan nilai yang tepat.

Coba kita lihat di teks, dikatakan 1,6 mil, artinya jaraknya sekitar 2 kilometer. Seperti apa nilai yang menunjukkan kepada kita itu menggunakan meter sebagai nilai satuan. Jika kita ubah ke kilometer, itu akan menjadi 2,5 kilometer.

Di sisi lain, untuk durasi, ia menggunakan detik untuk nilainya. Jika kita ubah ke menit, itu akan menjadi 6,9 menit.

IV. IMPLEMENTASI DAN HASIL PENGUJIAN

A. Sistem Implementasi

Implementasi sistem ini menggunakan *framework React Native* berbasis bahasa pemrograman *JavaScript* dan perangkat yang digunakan untuk menjalankan aplikasi ini adalah *smartphone* Android. Kemudian untuk penulisan *source code* menggunakan Visual Studio Code.

B. Implementasi Algoritma

Algoritma yang digunakan untuk mencari rute terbaik dalam penelitian ini adalah algoritma *Hill Climbing* dan *Greedy*. Dalam proses pencarian rute terbaik menggunakan 2 parameter sebagai patokan, yaitu parameter jarak dan durasi. Metode ini digunakan dalam menggabungkan kedua variabel yang nantinya akan mendapatkan bobot akhir yang digunakan dalam perhitungan pada Algoritma *Hill Climbing* dan *Greedy*. Langkah-langkah untuk mengimplementasikan algoritma ke dalam aplikasi adalah sebagai berikut:

1. Nilai latitude longitude node dan lokasi instance yang telah disiapkan sebelumnya akan dimasukkan ke dalam array.

2. Lokasi pemanggil akan diambil melalui GPS dan lokasi tujuan instansi akan dipilih sesuai instansi yang dipilih oleh penelepon kemudian lokasi tersebut akan diubah menjadi latitude longitude untuk kebutuhan implementasi.
3. Setelah lokasi pemanggil didapatkan maka akan dicari jarak terdekat sesuai perhitungan algoritma yang digunakan dari lokasi pemanggil dengan salah satu lokasi node dan akan digunakan sebagai rute selanjutnya. Proses penentuan node berikutnya dan seterusnya hingga nilai algoritma dan durasi waktu tempuh yang digunakan dengan masing-masing persentase atau perbandingan bobot SAW akan ditentukan setelah dilakukan pengujian bobot SAW.

C. Pengujian Bobot SAW

Pengujian bobot SAW dilakukan untuk mengetahui berapa rasio bobot yang optimal untuk jarak dan durasi. Pengujian ini dilakukan untuk mengetahui apakah pemilihan bobot dapat mempengaruhi pemilihan rute yang dipilih. Pengujian dilakukan dengan lokasi awal Telkom University Bandung dan lokasi tujuan RSU Bina Sehat pada hari yang sama dengan jam yang berbeda. Hasil uji bobot dapat dilihat pada tabel berikut:

TABLE III. HASIL PENGUJIAN BOBOT SAW

No	Jam	50% Jarak + 50% Durasi	50% Jarak + 50% Durasi	50% Jarak + 50% Durasi	50% Jarak + 50% Durasi	50% Jarak + 50% Durasi
1.	9.30	berhasil	berhasil	berhasil	berhasil	berhasil
2.	13.00	berhasil	berhasil	berhasil	berhasil	berhasil
3.	16.45	berhasil	berhasil	berhasil	berhasil	berhasil
4.	20.00	berhasil	berhasil	berhasil	berhasil	berhasil

Berdasarkan percobaan yang dilakukan untuk memilih bobot optimum dengan jam yang berbeda pada hari yang sama, semua percobaan menggunakan Algoritma *Hill Climbing* dan *Greedy* berhasil dilakukan. Pada pengujian SAW menggunakan Algoritma *Hill Climbing* dan *Greedy* ditemukan 11 rute yang serupa. Dari empat kali pengujian, lalu lintas terberat terjadi pada pukul 09.30 dan masing-masing bobot yang berbeda memiliki jangkauan relatif minimal. Karena kedua algoritma tersebut menggunakan biaya heuristik, maka tabel iv dibawah ini akan menampilkan hasil akhir dari nilai alternatif

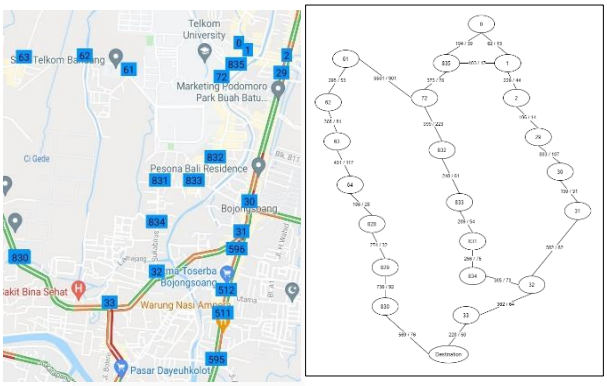
TABLE IV. HASIL PENGUJIAN NILAI ALTERNATIF

Algoritma *Hill Climbing* dan *Greedy*							
Jam	Lokasi	Tujuan	SAW				
			50:50	60:40	70:30	40:60	30:70
09.30	0	1	0,138	0,163	0,188	0,114	0,089
		835	0,154	0,176	0,197	0,133	0,111
Rentang			0,016	0,013	0,009	0,019	0,022

Pada tabel IV dapat dilihat bahwa range algoritma SAW Hill Climbing dan algoritma Greedy untuk nilai range tertinggi terdapat pada jarak SAW 30% dan durasi 70% dengan nilai range 0,022. Dapat disimpulkan bahwa bobot SAW yang digunakan untuk masing-masing algoritma adalah 30% jarak dan 70% durasi.

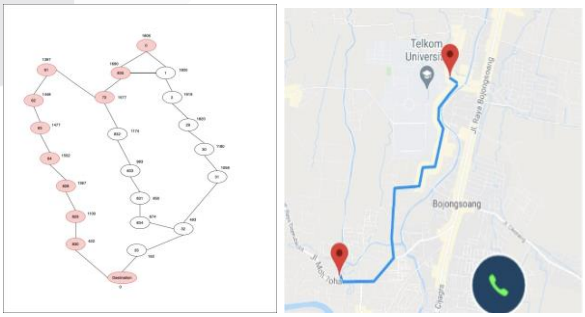
D. Pengujian Akurasi

Pengujian akurasi merupakan pengujian yang dilakukan untuk mengetahui performa dari algoritma yang digunakan pada aplikasi. Pengujian akurasi akan dihitung akurasinya dengan membandingkan hasil perhitungan atau pemilihan rute oleh sistem dengan hasil pemilihan rute manual. Pemilihan rute disini akan digunakan oleh Telkom University sebagai lokasi awal dan Rumah Sakit Umum Bina Sehat sebagai lokasi tujuan. Node yang digunakan dalam pengujian ini dapat dilihat pada Gambar 3 berikut ini:

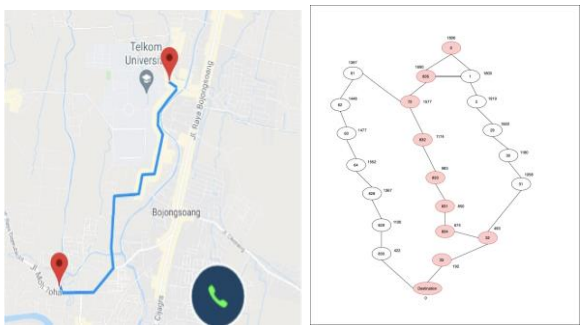


Gambar 3. Maps Node and Grafik

Untuk rute dengan menggunakan algoritma Hill Climbing dan Greedy, hasilnya dapat dilihat pada gambar 4 dan 5 di bawah ini.



Gambar. 4. Perbandingan perhitungan algoritma *Hill Climbing* secara manual dan perhitungan pada aplikasi



Gambar. 5. Perbandingan perhitungan algoritma *Greedy* secara manual dan perhitungan pada aplikasi

Untuk pengujian waktu respon, dapat dilihat pada tabel dibawah ini:

TABLE V. WAKTU RESPON

No.	Waktu Respon *Hill Climbing* (detik)	Waktu Respon *Greedy* (detik)
1	15,9	10,1
2	14,6	09,5
3	19,6	12,3
4	14,5	08,2
5	17,4	11,3
6	12,3	06,4
7	17,7	12,4
8	16,3	11,1
9	16,1	09,5
10	15,1	10,0

Rata-rata waktu respon *Hill Climbing* = $\frac{total\ \ nilai\ waktu\ respon}{total\ waktu\ respon}\ x\ 100\% = \frac{159,5}{10} =$ 15,95 detik

Rata-rata waktu respon *Greedy* = $\frac{total\ \ nilai\ waktu\ respon}{total\ waktu\ respon}\ x\ 100\% = \frac{100,8}{10} =$ 10,08 detik

V. KESIMPULAN

Dari penelitian ini, kesimpulan yang dapat kami peroleh adalah:

1. Algoritma Hill Climbing dan Greedy diimplementasikan dengan baik pada aplikasi menggunakan bobot SAW 30% jarak dan 70% durasi.
2. Untuk pengujian dengan lokasi saat ini adalah Telkom University dan RSU Bina Sehat sebagai tujuan berdasarkan response time, Greedy memberikan rute terbaik dengan response time 10,08 detik.

REFERENSI

[1] A. T. Haryanto, "detikinet," Detik Network, 20 February 2020. [Online]. Available: https://inet.detik.com/cyberlife/d-4907674/riset-ada-1752-juta-pengguna-internet-di-indonesia. [Accessed 20 November 2020].

[2] T. Narognsak, "Development of Metropolitan Police Emergency Call System," *International Journal of Crime, Law and Social Issues*, vol. 5, no. 1, pp. 134 - 145, 2018.

[3] W. L. Q. J. W. Z. D. L. L. Z. W. Q. Wang C., "Path Planning of Automated Guided Vehicles Based on Improved A-Star Algorithm," in *International Conference on Information and Automation Lijiang*, Lijiiang, 2015.

[4] B. Siregar, "Food Delivery System with the Utilization of Vehicle using Geographical Information System (GIS) and A Star Algorithm," in *International Conference on Computing and Applied Informatics*, 2017.

[5] F. Desti and M. N. Nura, "IMPLEMENTASI ALGORITMA HILL CLIMBING PADA PENENTUAN JARAK TERPENDEK KOTA WISATA DI INDONESIA," *Riset Informatika*, vol. I, no. 3, Juni 2019.

[6] G. Xieji, "A Routing Algorithm based on zigbee technology," *iJOE*, vol. 14, no. 11, 2018.

[7] C. Rafaele, C. Carmine and G. Bruce, "Carousel Greedy: A Generalized Greedy Algorithm with Applications in Optimization," 2018.

[8] J. Daud, "Studi Efektifitas Pengunaan Halte di Kota Medan," *Jurnal Sistem Teknik Industri*, vol. VI, no. 3, pp. 73-80, 2005.

[9] M. K. H and N. Khairina, "Pencarian Jalur Terpendek Dengan Algoritma Dijkstra," *Jurnal dan Penelitian Teknik Informatika*, vol. II, no. 2, pp. 18-23, 2017.

[10] B. Julia, R. K. Lars and M. Krystian, "Hill Climbing Algorithm and Trivium," *Lecture Notes in Computer Science*, vol. 6544, pp. 57-73, 2011.

[11] E. Ilham, "IT-Jurnal.com," 2018. [Online]. Available: https://www.it-jurnal.com/pengertian-algoritma-greedy/. [Accessed 29 July 2021].