Implementasi Algoritma Dijkstra Untuk Analisis Rute Transportasi Umum Transjogja Berbasis Android

Sunardi[a], Anton Yudhana [b], Ahmad Azhar Kadim[c]

[a,b] Teknik Elektro, Fakultas Teknologi Industri, Universitas Ahmad Dahlan
[c] Magister Teknik Informatika, Fakultas Pascasarjana, Universitas Ahmad Dahlan

Naskah Diterima : 24 Oktober 2018; Diterima Publikasi : 9 Mei 2019

DOI : 10.21456/vol9iss1pp32-38

Abstract

TransJogja is one of the public transportation media used by Yogyakarta residents. Currently TransJogja has served many passengers with various routes and has become a favorite transportation for locals and tourists visiting Yogyakarta. TransJogja continues to provide the best services for its users to attract potential passengers. Lack of information about bus routes and stops creates a negative impression for TransJogja users such as using the bus or going down to the bus which is not the destination of the user. This study aims to implement and analyze the TransJogja route using the Dijkstra Algorithm and utilize Android as its medium so that it can increase the interest of Transjogja users and facilitate service. Dijkstra is an algorithm to find the best route by comparing all the good weights in terms of distance, time and route to be passed. In this study the weights used are distance in kilometers (KM) and the best route results are obtained with shorter execution time which is <0.0076 seconds and the results are faster than the Ant Colony algorithm which has the fastest result is 0.0102. Based on the results obtained it can be concluded that Dijkstra's algorithm is the right algorithm used to determine the Trans Jogja route because it is based on the variable weight used.

Keywords : Android; Ant Colony; Dijkstra.

Abstrak

TransJogja merupakan salah satu media transportasi umum yang digunakan oleh warga Yogyakarta. Saat ini TransJogja sudah melayani banyak penumpang dengan berbagai rute dan menjadi transportasi favorit bagi warga lokal maupun pendatang yang berkunjung ke Yogyakarta. TransJogja terus memberikan layanan terbaik bagi penggunanya agar dapat menarik minat calon penumpang. Kurangnya informasi mengenai rute dan halte bus menimbulkan kesan negatif bagi pengguna TransJogja seperti salah pemilihan rute bus ataupun turun di halte yang bukan tujuan dari pengguna teresebut. Penelitian ini bertujuan untuk melakukan implementasi dan analisis rute TransJogja menggunakan Algoritma Dijkstra serta memanfaatkan Android sebagai medianya sehingga dapat menambah minat pengguna Transjogja dan meningkatkan pelayanan. Algortima Dijkstra merupakan algoritma untuk mencari rute terbaik dengan membandingkan semua bobot berupa jarak atau waktu. Pada penelitian ini bobot yang digunakan adalah jarak dengan satuan kilometer (KM) dan didapatkan hasil rute terbaik dengan waktu eksekusi lebih singkat yaitu < 0,0076 detik dan hasil tersebut lebih cepat dibandingkan algoritma *Ant Colony* yang memiliki hasil tercepatnya adalah 0,0102. Berdasarkan hasil yang diperoleh dapat disimpulkan bahwa algoritma Dijkstra adalah algoritma yang tepat digunakan untuk penentuan rute Trans Jogja karena berdasarkan variabel bobot yang digunakan.

Kata kunci: Android; Ant Colony; Dijkstra.

1. Pendahuluan

Transportasi umum merupakan salah satu pilihan masyarakat untuk bepergian dari satu tempat ke tempat lainnya. Transportasi umum dapat menjadi solusi untuk mengurangi kemacetan akibat jalan umum yang tidak dapat menampung lagi volume kendaraan. Banyak cara yang ditempuh oleh pemerintah dari mulai membuat peraturan ganjil-genap dan menggunakan kendaraan umum pada hari-hari tertentu. Langkah tersebut belum sesuai harapan diinginkan. Salah satu transportasi umum saat ini yang sering digunakan adalah bus, karena dapat menampung penumpang lebih banyak, contohnya TransJogja.

TransJogja merupakan salah satu transportasi umum yang saat ini diterapkan untuk mengurangi kemacetan. Bus TransJogja sudah menjadi pilihan masyarakat karena biayanya yang relatif murah dibandingkan menggunakan transportasi umum

*) Penulis korespondensi: azharkadim@yahoo.com

lainnya seperti taxi, taxi online, ojek pangkalan maupun ojek online (Syukri, 2014). TransJogja memiliki beberapa titik halte, namun kurang informatif mengenai posisi halte dan jalur bus TransJogja. Informasi lengkap mengenai TransJogja hanya ada pada halte-halte tertentu. Hal tersebut dapat mengurangi minat penumpang untuk menggunakan TransJogja. TransJogja harus memikirkan cara agar minat pengguna penumpang dapat bertambah serperti memadukan dengan kemajuan teknologi.

Perkembangan teknologi saat ini berkembang dengan sangat cepat karena kemudahan dalam pengembangan dan terjangkaunya alat-alat yang dibutuhkan. Salah satunya teknologi dalam bidang Mobile (Yudhana, *et al.* 2018) seperti smartphone dapat membantu aktifitas manusia untuk berkomunikasi dan urusan pekerjaan. Kemajuan teknologi tersebut dapat dijadikan solusi untuk menanggulangi masalah yang ada pada TransJogja seperti penumpang dapat mengakses dengan mudah informasi secara detail, cepat dan dimana saja. Selain itu dapat memanfaatkannya untuk mengetahui posisi sehingga dapat diprediksi waktu kedatangan bus.

Algoritma Dijsktra merupakan algoritma yang digunakan untuk menentukan jalur terpendek dan tercepat dari satu titik ke titik lainnya dengan cara menghitung segala kemungkinan jalur yang akan dilalui (Ratnasari, *et al,* 2013). Adapun penelitian yang menggunakan algoritma Dijkstra untuk menentukan rute adalah sebagai berikut :

1. Penelitian berjudul "Penentuan Rute Terpendek Pengambilan Sampah di Kota Merauke Menggunakan Algortima Dijkstra" melibatkan beberapa pertimbangan utama meliputi rute kendaraan dan meminimalisir biaya distribusi, serta dapat memperluas wilayah pengambilan sampah dengan armada yang terbatas. (Andayani dan Perwitasari, 2014).
2. Studi "Finding the Shortest Paths Among Cities in Java Island Using Node Combination Based on Dijkstra Algorithm" mendapatkan jalur terpendek antara kota-kota yang ada di pulau Jawa. (Amaliah, *et al*. 2016).
3. Penelitian "Algoritma Dijkstra untuk pemetaan dan menentukan jalur terpendek pariwisata yang ada di Timor Leste berbasis Web" mendapatkan nilai keakuratan jarak rata-rata 0,03%. Hasil pengukuran berupa rute dan waktu tempuh dengan kecepatan rata-rata kendaraan yang bervariasi. (Gusmão, *et al.* 2013).
4. Penelitian "Penentuan Jalur Terpendek dan Jarak Terpendek Alternatif Menggunakan Algoritma Dijkstra Serta Estimasi Waktu Tempuh" dengan cara membuat graf dengan node (tempat wisata) sebagai titik awal dan titik akhir. (Ratnasari, *et al.* 2013).

Perbedaan penelitian ini dengan penelitian sebelumnya adalah memanfaatkan algoritma Dijkstra sebagai pencari rute Transjogja dan Android sebagai medianya. Adapun penelitian ini bertujuan memanfaatkan Algoritma Dijkstra untuk penentuan rute TransJogja dan memanfaatkan android sebagai media untuk menampilkan hasil rute yang sudah diproses menggunakan Dijkstra, selain itu memanfaatkan fitur GIS yang dapat memberikan informasi kepada pengguna mengenai posisi dari halte dan bus.

2. Kerangka Teori

2.1. Algoritma Dijkstra

Algoritma Dijkstra adalah algoritma yang dipakai untuk memecahkan permasalahan jarak terpendek untuk sebuah graf yang berarah. Nama Dijkstra diambil dari nama penemunya yaitu Edsger Dijkstra yang merupakan seorang ilmuwan komputer (Sunardi, *et al,* 2017). Algoritma ini menggunakan prinsip greedy dengan membandingkan setiap bobot minimum yang dilewati kemudian disimpan dalam himpunan. Algoritma Dijkstra populer digunakan untuk menentukan jalur terpendek dan tercepat. Algoritma Dijkstra bekerja dengan cara menghitung semua vertex atau titik yang tersedia. Algoritma Dijkstra dapat menemukan jalur terpendek pada graph yang memilik vertex dan jarak antar vertex yang memiliki bobot positif. (Wibowo dan Wicaksono, 2012). Penerapan algoritma Dijkstra membutuhkan waktu eksekusi yang lebih cepat dibandingkan algoritma lain seperti algotima *Ant Colony* sehingga Dijkstra lebih banyak digunakan dalam pencarian jalur optimum (Azizah dan Mahendra, 2017).

Penelitian ini menggunakan algoritma Dijkstra sebagai penentu rute dari Transjogja karena dalam penerapannya lebih akurat dan cepat sehingga proses yang dilakukan tidak membutuhkan waktu lama pada saat diaplikasikan.

2.2. Cara Kerja Algoritma Dijkstra

Algoritma Dijkstra bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke-n, setidaknya ada n node yang sudah diketahui jalur terpendek. Langkah-langkah algoritma Dijkstra dapat dilakukan sebagai berikut:

1. Penentuan titik yang akan menjadi node awal, lalu diberi bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.
2. Pemberian nilai bobot (jarak) untuk setiap titik ke titik lainnya, diberikan nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi).
3. Semua node yang belum dilalui dan node awal sebagai "Node keberangkatan".

4. Dari node keberangkatan, selanjutnya node tetangga yang belum dilalui dan dihitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.
5. Saat selesai mempertimbangkan setiap jarak terhadap node tetangga, node yang telah dilalui diberi label "Node dilewati". Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. "Node belum dilewati" dengan jarak terkecil (dari node keberangkatan) sebagai "Node Keberangkatan" selanjutnya dan ulangi langkah 4-6.

Pengujian dilakukan untuk mendapatkan kesesuaian jarak dari titik awal sampai titik akhir pada peta yang dihasilkan oleh algoritma Dijkstra dengan jarak pengukuran. Untuk mendapatkan seberapa besar persen selisih nilai antara jarak pengukuran dengan jarak yang hasil Dijkstra dapat dihitung dengan persamaan (1). (Azizah dan Mahendra, 2017).

$$S_j = \left[\frac{\sum_{i=1}^{N}(P_1 - P_2)}{N}\right] \times 100\% \quad (1)$$

Dimana :
S_j = Selisih jarak
P_1 = Jarak pengukuran
P_2 = Jarak yang dihasilkan Dijkstra

2.3. Informasi Transjogja Saat ini

Saat ini informasi yang diperoleh oleh pengguna dari TransJogja adalah dengan bertanya kepada pada petugas yang berada di halte atau melihat informasi peta jalur dan rute yang terpasang pada halte tertentu atau dapat dilihat pada *website* resmi Dinas Perhubungan DIY dan aplikasi infojogja. Seperti pada Gambar 1.

Gambar 1. Peta transjogja dan aplikasi infojogja

Dari Gambar 1 dapat dikembangkan lagi untuk menambah fitur untuk mempermudah pengguna TransJogja dengan menyajikan informasi yang lebih detail seperti informasi halte yang akan dilewati dari titik asal ke titik tujuan, informasi waktu dan tempat transit.

Pada penelitian ini akan menambahkan fitur yang sudah disebutkan sebelumnya serta memanfaatkan *Geographic Information System* (GIS) untuk *track location* dari bus TransJogja sehingga pengguna dapat memprediksikan waktu tiba dan berangkat dari bus.. GIS adalah sistem informasi berbasis komputer, yang digunakan untuk memproses data spasial yang ber-georeferensi (berupa detail, fakta, kondisi, dsb) yang disimpan dalam suatu basis data dan berhubungan dengan persoalan serta keadaan dunia nyata *(real world)*. Manfaat SIG secara umum memberikan informasi yang mendekati kondisi dunia nyata, memprediksi suatu hasil dan perencanaan strategis. (Fauzan, 2014). Studi lain mendefinisikan SIG adalah informasi mengenai permukaan bumi dan semua objek yang berada di atasnya, yang menjadi kerangka bagi pengaturan dan pengorganisasian bagi semua tindakan selanjutnya. Teknologi Sistem Informasi Geografis mengintegrasikan operasi umum database, seperti query dan analisa statistik, dengan kemampuan visualisasi dan analisa yang unik yang dimiliki oleh pemetaan. (Soyusiawaty, *et al.* 2007).

Penelitian ini juga menggunakan sistem operasi Android untuk penerapan dan menampilkan hasil *output* dari perhitungan Dijkstra. Android merupakan sistem operasi yang berbasis *open source* dikembangkan oleh Google. Android memberikan kebebasan bagi *developer* untuk mengembangkan sistem operasi dan aplikasi yang dibuat. Sifat *open source* Android mendorong pengembang untuk membuat aplikasi dan mengunggahnya ke PlayStore sehingga dapat digunakan oleh pengguna Android lainnya. Aplikasi ini dapat digunakan oleh pengguna dengan mengunduhnya dari Android Market, lalu menginstalnya di ponsel cerdas mereka (Riadi, *et al.* 2017). Sistem operasi Android merupakan teknologi yang pengguna cukup besar karena banyaknya vendor yang menggunakan sistem operasi ini. (Anwar dan Riadi, 2017).

3. Metode

3.1. Prosedur Penelitian

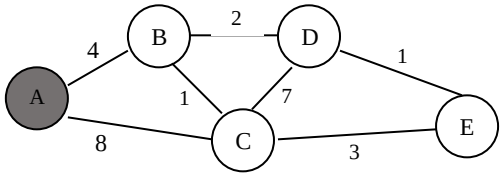
Penelitian ini menggabungkan teknologi Mobile untuk membantu menunjang kualitas pelayanan TransJogja. Penilitian akan membuat aplikasi Mobile berbasis Android yang dapat menampung semua informasi mengenai TransJogja seperti posisi halte, bus dan rute. Adapun untuk ketepatan penentuan rute dari TransJogja menggunakan Algoritma Dijkstra.

Bahan penelitian yang dibutuhkan adalah informasi mengenai TransJogja seperti titik-titik halte yang ada serta rute yang dilewati tiap jalur bus. Alat

yang digunakan adalah Laptop Toshiba L745 dengan spesifikasi RAM 8GB, *Processor* Core i5 dan Windows 10 Pro 64 bit. Kebutuhan perangkat lunak *(Software)* seperti Android Studio untuk pengembangan aplikasi android, SQLite *Destkop* untuk menganalisa database internal pada android, Server untuk menampung data-data yang dibutuhkan pada penelitian.

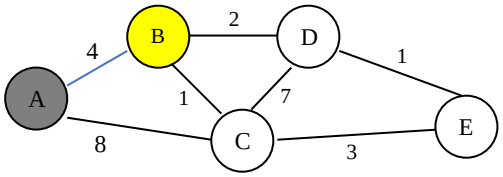
3.2. Proses Perhitungan Dijkstra

Dalam penerapannya algoritma *Dijkstra* membutuhkan parameter asal dan akhir. Untuk lebih detail dapat dilihat skema algoritma *Dijkstra* seperti Gambar 2.



Gambar 2. Menentukan titik awal

Gambar 2 menunjukkan titik A adalah titik awal dan titik tujuan adalah E. Algoritma Dijkstra akan mencari titik terdekat yang terhubung dengan titik A dan memiliki bobot terkecil.

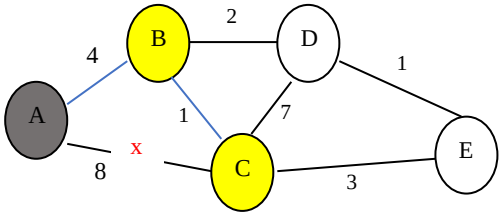


Gambar 3. Bobot terendah terhubung dengan A

Gambar 3 menunjukkan bahwa titik yang memiliki bobot minimum adalah B. Selanjutnya, Algoritma *Dijkstra* akan membandingkan bobot dari titik yang terhubung dengan B dengan cara sebagai berikut:

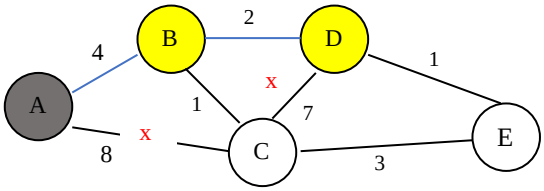
- Titik A-B-D = 6
- Titik A-B-C = 5

Dari perbandingan diatas maka didapatkan titik A-B-C memiliki bobot yang terkecil, sehingga algoritma *Dijkstra* akan melanjutkan perhitungan bobot yang terhubung dengan C. Pada tahap ini titik D tidak menjadi pilihan karena bobot yang dimiliki lebih besar dari bobot A-B-D dan A-B-C seperti gambar 4.



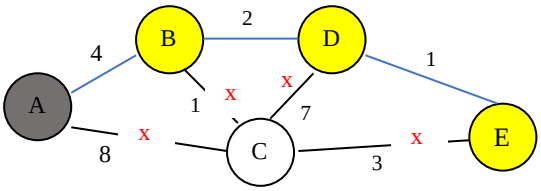
Gambar 4. Bobot terendah terhubung dengan C

Gambar 4 menunjukkan bahwa titik A-B-C-D = 12, sedangkan A-B-C-E = 8 dan sudah mencapai titik tujuan yaitu titik E, akan tetapi A-B-C-E belum bisa menjadi pilihan karena masih ada jalur lain yang memungkinkan memiliki bobot lebih rendah yaitu A-B-D = 6 dan belum dibandingkan oleh algoritma.



Gambar 5. Bobot terendah terhubung dengan D

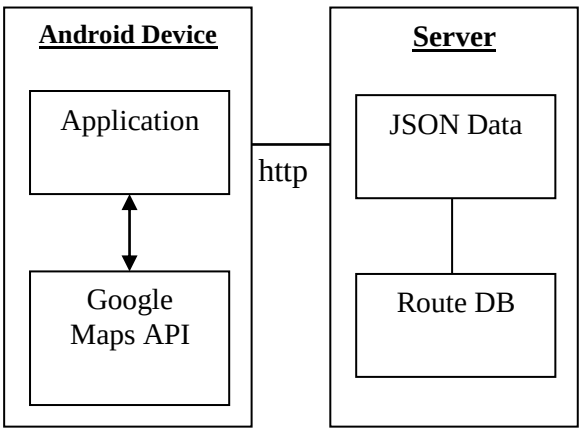
Gambar 5 menunjukkan bahwa titik D terhubung dengan C dan E, akan tetapi titik C tidak menjadi pilihan karena bobot yang dimiliki A-B-D-C melebihi bobot minimum A-B-C-E, sehingga algoritma *Dijkstra* akan langsung menghitung bobot A-B-D-E. Pada tahap ini, didapatkan nilai minimum A-B-D-E = 7 sedangkan A-B-C-E = 8, maka hasil jalur yang memiliki bobot terendah adalah A-B-D-E dapat dilihat pada Gambar 6.



Gambar 6. Hasil akhir jalur terpendek

3.3. Proses Pengiriman Hasil Ke Android

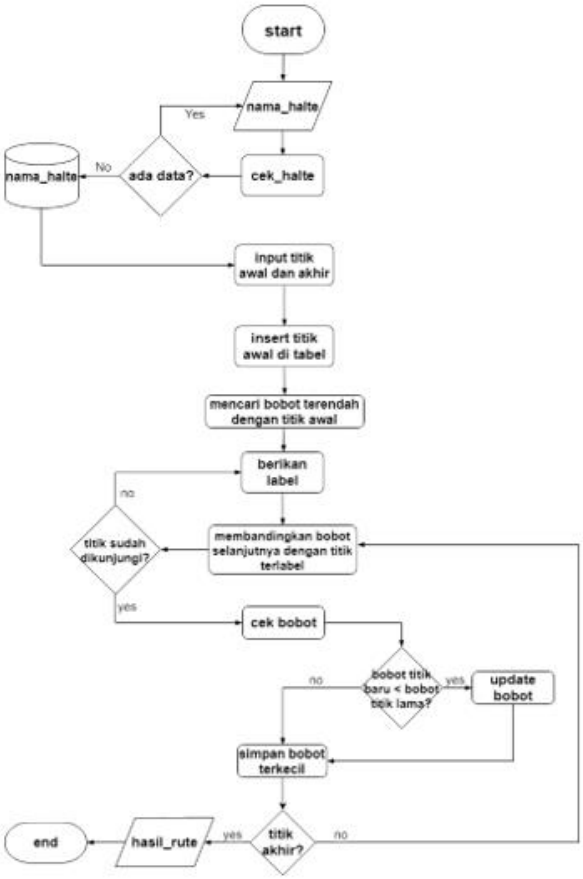
Aplikasi yang dibuat melakukan *request* untuk pencarian rute menggunakan Algoritma *Dijkstra*. Perhitungan Dijkstra dilakukan di *server* dan hasil dikirimkan kembali ke aplikasi dalam bentuk rute yang akan dilewati. Ilustrasi dapat dilihat seperti pada Gambar 7.



Gambar 7. Komunikasi antara aplikasi dan server

3.3. Flowchart

Pada Gambar 8 dibawah ini merupakan *flowchart* sistem ketika dijalankan. Pada saat aplikasi dijalankan, data nama halte otomatis disinkronasikan dengan SQLite pada aplikasi sesuai yang ada di server.



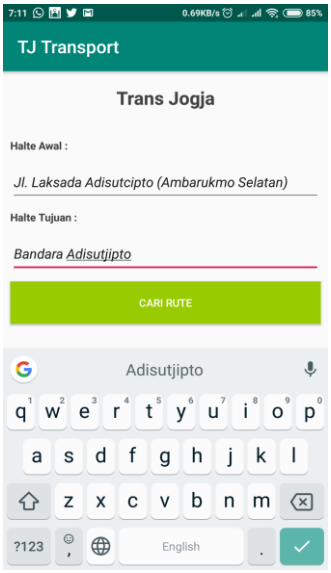
Gambar 8. Flowchart sistem

Pengguna menginputkan nama halte sesuai yang ada pada SQLite seperti pada Gambar 1. Setelah pengguna menginputkan titik awal dan akhir maka aplikasi mencari rute yang akan dilalui. Prinsip algoritma *Dijkstra* yaitu membandingkan semua bobot yang ada dan terhubung dengan titik awal dan akhir. *Dijkstra* menyimpan bobot yang terkecil sehingga data yang ditampilkan sesuai dengan rute TransJogja yang digunakan.

4. Hasil dan Pembahasan

4.1. Proses Input Data

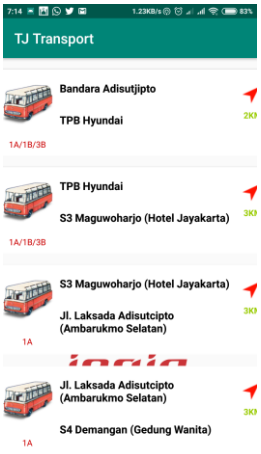
Pada proses ini pengguna transportasi umum Transjogja akan menginputkan data yang diperlukan yaitu titik awal dan titik akhir penumpang. Semua titik awal dan akhir sudah tersimpan kedalam database server dan SQLite. Adapun tujuan disimpan di database server agar jika terjadi perubahan data seperti penambahan rute baru maka SQLite pada aplikasi akan otomatis *syncronize* dengan database yang ada pada server sehingga titik yang diinputkan sesuai dengan data yang diperlukan seperti pada Gambar 9.



Gambar 9. Tampilan input data

4.2. Proses Tampilan Hasil Rute

Pada Gambar 10 merupakan tampilan hasil rute yang dihasilkan dengan Algoritma Dijkstra. Pada tampilan tersebut terdapat informasi halte yang akan dilewati dan bus yang akan digunakan.



Gambar 10. Tampilan hasil pencarian rute

4.3. Proses Tampilan GIS

Pada Gambar 11 merupakan tampilan Maps ketika *user* memilih tombol navigasi ( ). *User* dapat melihat posisi halte yang akan dilalui, bus-bus yang ada disekitar lengkap dengan kode jalur ( ), dan posisi *user* sendiri ( ). Menu ini menampilkan semua posisi secara *real-time tracking* yang berarti posisi akan berubah sesuai dengan koordinat dari perangkat yang digunakan baik *user* maupun operator bus.

Gambar 11. Tampilan input data

4.4. Proses Pengujian Dijkstra

Kemampuan Algoritma *Dijkstra* dalam menentukan jalur terpendek dan jumlah titik yang dilalui, dilakukan uji sebanyak enam kali dengan titik awal dan tujuan berbeda. Pengujian yang dilakukan didapatkan hasil pada Tabel 1.

Tabel 1. Hasil uji algoritma dijkstra

No	Input		Output		Waktu Eksekusi (Detik)
	Titik Awal (Halte)	Titik Tujuan (Halte)	Jarak (KM)	Jumlah titik	
1	Terminal Prambanan	Malioboro	16,0	13	0,0076
2	Malioboro	Terminal Jombor	6,2	11	0,0045
3	Terminal Jombor	Bandara Adisutjipto	13,0	8	0,0068
4	Malioboro	Bandara Adisutjipto	10,0	12	0,0065
5	Ambarukmo	Giwangan	5,8	7	0,0037
6	Bandara Adisutjipto	Tugu	8,0	9	0,0060

Hasil yang didapatkan memiliki rata-rata waktu eksekusi tercepat 0,0037 detik dan terbesar 0,0076 detik. Cepat atau lambat waktu eksekusi tergantung dari jarak yang akan dilalui bukan dari jumlah titik karena karakteristik dari Dijkstra adalah menghitung bobot (KM) yang ada. Sehingga semakin jauh bobot (KM) yang dilewati maka akan semakin banyak perbandingan titik *(Node)* sehingga waktu eksekusi akan semakin lama. Jumlah titik hanya menjadi hasil rute yang akan dilalui.

4.5. Perbandingan Dijkstra dengan Ant Colony

Pengujian kemampuan dari algoritma Dijkstra dalam menentukan jalur terpendek dan jumlah titik yang dilalui, penulis membandingkan waktu eksekusi algoritma Dijkstra dengan *Ant Colony*. Perbandingan dilakukan dengan menggunakan inputan titik awal dan tujuan yang sama. Adapun perbandingan waktu eksekusi dari kedua algoritma tersebut disetiap percobaan dapat dilihat pada Tabel 2.

Tabel 1. Hasil uji algoritma dijkstra

No	Input		Waktu Eksekusi (Detik)	
	Titik Awal (Halte)	Titik Tujuan (Halte)	Dijkstra	*Ant Colony*
1	Terminal Prambanan	Malioboro	0,0076	0,0246
2	Malioboro	Terminal Jombor	0,0045	0,0126
3	Terminal Jombor	Bandara Adisutjipto	0,0068	0,0205
4	Malioboro	Bandara Adisutjipto	0,0065	0,0105
5	Ambarukmo	Giwangan	0,0037	0,0149
6	Bandara Adisutjipto	Tugu	0,0060	0,0102

Hasil yang diperoleh adalah waktu eksekusi algoritma *Ant Colony* membutuhkan waktu yang lebih besar dibandingkan Dijkstra hal ini disebabkan karena algoritma *Ant Colony* prinsip kerjanya menyebar semut sesuai dengan jumlah titik yang ada pada *graph*, sehingga data yang diproses semakin banyak, berbeda dengan algoritma Dijkstra yang prinsip kerjanya langsung membandingkan bobot tiap jalur yang ada pada *graph*.

5. Kesimpulan

Berdasarakan penelitian yang dilakukan didapatkan hasil yang didapatkan memiliki rata-rata waktu eksekusi tercepat 0,0037 detik dan terbesar 0,0076 detik. Cepat atau lambat waktu eksekusi tergantung dari jarak yang akan dilalui bukan dari jumlah titik karena karakteristik dari Dijkstra adalah menghitung bobot (KM) yang ada. Algoritma *Dijkstra* membutuhkan bobot antar titik *(node)* untuk proses perhitungan, tanpa bobot *Dijkstra* tidak dapat dijalankan. Implementasi ke media Android dapat berjalan sesuai dengan tujuan penilitian berdasarkan *screenshoot* aplikasi dan hasil output yang ditampilkan pada perangkat Android sesuai dengan jalur dari titik awal ke titik tujuan. Implementasi GIS berhasil diterapkan pada android dan dapat memberikan informasi mengenai *Track Location* dari bus TransJogja.

Daftar Pustaka

Amaliah, B., Fatichah, C., & Riptianingdyah, O., 2016. Finding The shortest paths among cities in java island using node combination based on dijkstra algorithm. International Journal on Smart Sensing and Intelligent Systems, 9 (4), 2219–2236.

Andayani, S., Perwitasari, E.W., 2014. Penentuan rute terpendek pengambilan sampah di kota merauke menggunakan algoritma Dijkstra. *Prosiding Seminar Nasional Teknologi Informasi & Komunikasi Terapan*, November 15, 164–170.

Anwar, N., Riadi, I., 2017. Analisis investigasi forensik whatsapp messenger smartphone terhadap whatsapp berbasis we*b*. Jurnal Ilmu Teknik Elektro Komputer Dan Informatika, 3 (1), 1–10.

Azizah, N., Mahendra, D., 2017. Geolocation dengan Metode Djikstra untuk menentukan jalur terpendek lokasi peribadata*n*. Jurnal Sistem Informasi Bisnis, 7 (2), 96–103.

Fauzan, M., 2014. Implementasi sistem informasi geografis menggunakan google maps api dalam pemetaan asal mahasiswa. Simetris, 5(2), 181–186.

Gusmão, A., Pramono, S. H., Sunaryo., 2013. Sistem Informasi geografis pariwisata berbasis web dan pencarian jalur terpendek dengan algoritma Dijkstra. Jurnal EECCIS, 7(2), 125–130.

Ratnasari, A., Ardiani, F., Nurvita, F., 2013. Penentuan jarak terpendek dan jarak terpendek alternatif menggunakan algoritma dijkstra serta estimasi waktu tempuh. *Prosiding Seminar Nasional Teknologi Informasi & Komunikasi Terapan, Novembe*r 16, 29–34.

Riadi, I., Umar, R., Firdonsyah, A., 2017. Identification of digital evidence on android's blackberry messenger using NIST mobile forensic method. International Journal of Computer Science and Information Security, 15(5), 3–8.

Soyusiawaty, D., Umar, R., & Mantofani, R. 2007. Sistem Informasi geografis objek wisata propinsi kepulauan bangka belitung berbasis web. *Prosiding Seminar Nasional Aplikasi Teknologi Informasi*, Juni 16, 17–22.

Sunardi, Yudhana, A., Kadim, A. A., 2017. Implementasi algoritma dijkstra dalam penentuan jalur dan pemesanan online transportasi umum berbasis android. *Prosiding Seminar Nasional Teknlogi Infromasi Dan Komunikasi*, Desember 18, 1–7.

Syukri, S., 2014. Penerapan Customer Satisfaction Index (CSI) dan Analisis GAP pada kualitas pelayanan trans jogja. Jurnal Ilmiah Teknologi Industri, 13(1), 103–111.

Wibowo, A. G., Wicaksono, A. P., 2012. Rancang Bangun aplikasi untuk menentukan jalur terpendek rumah sakit di Purbalingga dengan metode Algoritma Dijkstra. Juita, 2(1), 21–35.

Yudhana, A., Dwi, M., Putra, D., 2018. Rancang Bangun sistem pemantauan infus berbasis Android. Transmisi, 20 (2), 91–95.