PERANCANGAN DAN IMPLEMENTASI APLIKASI PENCARIAN RUTE OPTIMAL UNTUK PEMADAM KEBAKARAN BERBASIS ANDROID MENGGUNAKAN ALGORITMA FLOYD-WARSHALL

(Studi Kasus : Bandung dan Sekitarnya)

Optimal Finding Route Application Design And Implementation For Firefighter Based On Android Using Floyd-Warshall Algorithm

(Case Of Study : Bandung and Surrounding Area)

Aloisius Gonzaga Januar Widi Aquarizky[1], Budhi Irawan, S.Si., M.T.[2], Casi Setianingsih, S.T., M.T.[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom

[1]loiszaga@gmail.com, [2]budhiirawan@telkomuniversity.ac.id, [3]setiacasie@telkomuniversity.ac.id

Abstrak

Kebakaran merupakan suatu bencana yang disebabkan oleh api yang bergerak bebas dan dapat menyebabkan nyawa manusia terancam, kerusakan bangunan dan ekologi sekitarnya. Kebakaran dapat menjadi permasalahan yang serius jika tidak ditangani secara cepat karena akan menimbulkan kerugian yang sangat besar. Oleh karena itu, peran pemadam kebakaran sangat penting. Pemadam Kebakaran harus siap siaga apabila terjadi kebakaran dan dituntut harus memilih rute perjalanan yang optimal untuk datang ke tempat kejadian dengan cepat. Masalah pencarian rute perjalanan ini terkadang menjadi sebuah permasalahan yang cukup kompleks. Hal ini disebabkan kondisi jalan yang harus dilewati mobil pemadam kebakaran memiliki tingkat kepadatan kendaraan yang cukup tinggi. Pada penelitian tugas akhir ini, penulis membuat sebuah aplikasi pencarian rute untuk mobil pemadam kebakaran berbasis android dengan menggunakan algoritma *Floyd-Warshall*. Algoritma ini memberikan solusi pencarian rute yang optimal untuk pemadam kebakaran dengan memperhatikan kondisi jalan seperti kemacetan, kondisi ruas jalan dan sumber air terdekat dari lokasi kebakaran. Proses pencarian didasarkan pada perhitungan jarak tiap simpul dan memilih jarak terkecil antar titik. Diharapkan dengan adanya aplikasi ini dapat membantu memberikan panduan rute perjalanan yang optimal sehingga pemadam kebakaran dapat sampai di lokasi kebakaran dengan cepat dan kerugian yang disebabkan oleh kebakaran dapat ditekan seminimal mungkin.

Kata Kunci : Pencarian Rute Optimal, Pemadam Kebakaran, Algoritma *Floyd-Warshall*, *Android.*

Abstract

Fire is a disaster caused by a freely-moving fire that can cause human lives in danger, damage to buildings and the surrounding ecology. Fires can be a serious problem if not treated quickly because it will cause huge losses. Therefore, the role of firefighters is certainly very important for this case. Firefighters should always be alert in case of fire and choose the optimal travel route to come to the scene quickly. Search problem sometimes can be a fairly complex problem. It is caused by road conditions that must pass by the fire truck has a vehicle density levels are quite high.In this research, the author created a search application for the firetruck using Floyd-Warshall algorithm based on Android. This algorithm will provide the optimal route search solution for firefighters. By paying attention to road conditions such as traffic jams, road conditions and the nearest water source of the fire. Search process is based on the calculation of distance of each node and choose the smallest distance between points.Hope with this application can help provide guidance most optimal travel route, so firefighters can arrive to the scene quickly and losses caused by fire can be minimized.

***Keywords** : Searching the Optimal Route, Fire, Floyd-Warshall algorithm, Android.*

1. Pendahuluan

1.1 Latar Belakang

Kebakaran adalah sebuah bencana yang disebabkan oleh api yang bergerak secara bebas yang dapat membahayakan nyawa manusia, bangunan dan ekologi. Penyebab kemunculan api tersebut biasanya disebabkan oleh beberapa faktor seperti faktor manusia (*human error*), teknis dan alam. Kebakaran menimbulkan kerugian yang sangat besar dan tak terhitung jumlahnya. Untuk menekan tingkat kerugian yang dihasilkan oleh kebakaran, maka diperlukan tindakan yang cepat untuk mengatasinya. Salah satu solusi adalah dengan segera mendatangkan pemadam kebakaran ke tempat kejadian untuk memadamkan api. Pemadam kebakaran harus segera datang ke lokasi kebakaran dan memilih rute yang paling optimal untuk dilalui.

Pencarian rute optimal merupakan hal yang sangat penting untuk dilakukan ketika keadaan darurat atau keadaan yang membutuhkan waktu tempuh perjalanan yang sangat cepat. Permasalahan ini sering dialami oleh

Pemadam Kebakaran. Pemadam Kebakaran sering kesulitan untuk memilih rute yang terbaik dan tercepat dikarenakan banyaknya faktor penghambat di jalan seperti tingkat volume kendaraan dijalan, kondisi ruas jalan, dan lainnya. Oleh karena itu, diperlukan pencarian dan penentuan rute yang sangat optimal untuk mobil pemadam kebakaran dengan memperhatikan kondisi jalan seperti kemacetan, kondisi ruas jalan, dan posisi *hydrant* terdekat dengan lokasi kebakaran.

Dari beberapa poin permasalahan diatas, maka dibutuhkan sebuah pemecahan masalah berupa perancangan aplikasi pencarian rute untuk pemadam kebakaran berbasis *android*. Sistem yang tertanam pada aplikasi ini menggunakan algoritma *Floyd-Warshall* dan Metode *Simple Additive Weighting* (SAW). Metode SAW digunakan untuk memperhitungkan pemecahan solusi dari ketiga matriks utama pada aplikasi ini yaitu jarak, kemacetan dan lokasi hydrant. Metode tersebut menghasilkan sebuah solusi tunggal yang akan digunakan sebagai bahan olah Algoritma Floyd-Warshall. Algoritma ini dapat mencari semua jarak tiap simpul yang artinya dapat digunakan untuk menghitung bobot terkecil dari semua jalur yang saling menghubungkan antar titik.[1]

Pencarian dan penentuan rute yang optimal sangat diperlukan. Diharapkan dengan aplikasi ini, dapat membantu kinerja pemadam kebakaran agar semakin baik sehingga dapat menekan tingkat kerugian yang diakibatkan oleh kebakaran.

1.2 Perumusan Masalah

Rumusan masalah yang terdapat dalam Tugas Akhir ini adalah sebagai berikut :

(1) Bagaimana aplikasi memberikan informasi kepada pengemudi mobil pemadam kebakaran tentang pencarian dan penentuan rute optimal ke tempat tujuan ?
(2) Bagaimana merancang dan mengimplementasikan metode *Simple Additive Weighting* (SAW) ke dalam sebuah aplikasi berbasis android untuk memperhitungkan kriteria jarak, nilai kemacetan dan posisi hydrant dalam menentukan bobot suatu jalan ?
(3) Bagaimana merancang dan mengimplementasikan algoritma *Floyd-Warshall* ke dalam aplikasi pencarian rute optimal untuk pemadam kebakaran berbasis *smartphone Android* ?

1.3 Tujuan

Tujuan dari penelitian ini antara lain :

(1) Membuat sebuah aplikasi yang dapat memberikan informasi kepada pengemudi mobil pemadam kebakaran tentang pencarian dan penentuan rute optimal ke tempat tujuan.
(2) Merancang dan mengimplementasikan metode *Simple Additive Weighting* (SAW) ke dalam sebuah aplikasi berbasis android untuk memperhitungkan kriteria jarak, nilai kemacetan dan posisi hydrant dalam menentukan bobot suatu jalan ?
(3) Bagaimana merancang dan mengimplementasikan algoritma *Floyd-Warshall* ke dalam aplikasi pencarian rute optimal untuk pemadam kebakaran berbasis *smartphone Android* ?

1.4 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut :

(1) Aplikasi diimplementasikan pada sistem operasi *Android*.
(2) SDK Minimum yang digunakan adalah API Level 17 (*Jellybean*).
(3) Algoritma pencarian rute terpendek menggunakan algoritma *Floyd-Warshall*.
(4) Aplikasi hanya memberikan informasi lokasi hydrant yang ada di Kota Bandung tanpa memperhatikan keefektifan hydrant tersebut.
(5) Aplikasi tidak dapat melakukan *update* pemberian informasi otomatis ketika ada suatu nama jalan yang berganti atau jalur yang ditutup tiba-tiba oleh beberapa pihak terkait.
(6) Data Kemacetan yang terdapat pada aplikasi adalah hasil survey dan data hasil penelitian yang dilakukan oleh Dinas Perhubungan Kota Bandung.
(7) Jalur yang digunakan sebagai rute terbaik adalah jalur primer, jalur sekunder dan jalur yang dapat dilalui kendaraan roda empat, tidak termasuk jalan-jalan kecil.
(8) Studi Kasus hanya dilakukan di kota Bandung dan sekitarnya dengan batas-batas daerah yang telah ditentukan.
(9) Pembuatan peta secara manual menggunakan JOSM dan menyesuaikan dengan area dinas Pemadam Kebakaran Kota Bandung dapat diperbantukan.
(10) Tidak ada proses *login* pada aplikasi.
(11) Peta yang digunakan adalah peta *OpenStreetMap*.

2. Dasar Teori

2.1 Android

Android adalah sistem operasi berbasis Linux yang dimodifikasi untuk perangkat bergerak (*mobile device*). Yang terdiri dari sistem operasi, *middleware*, dan aplikasi-aplikasi utama. Awalnya android dikembangkan oleh

Android Inc. Perusahaan ini kemudian dibeli oleh Google pada tahun 2005. Sistem operasi Android kemudian diluncurkan bersamaan dengan dibentuknya organisasi Open Handset Alliance tahun 2007. Selain Google, beberapa nama-nama besar juga ikut serta dalam Open Handset Alliance, antara lain Motorola, Samsung, LG, Sony Erricson, T-Mobile, Vodafone, Toshiba, dan Intel. [3]

2.1 SIG (Sistem Informasi Geografis)

Sistem Informasi Geografis adalah sebuah sistem atau teknologi berbasis komputer yang dibangun dengan tujuan untuk mengumpulkan, menyimpan, mengolah dan menganalisa, serta menyajikan data dan informasi dari suatu obyek atau fenomena yang berkaitan dengan letak atau keberadaannya dipermukaan bumi. Pada dasarnya, SIG dapat dirinci menjadi beberapa sub-sistem yang saling berkaitan yang mencakup input data, manajemen data, pemroresan atau analisis data, pelaporan (*output*) dan hasil analisa. [2]

2.2 GPS

Global Positioning System (GPS) merupakan sebuah sistem navigasi global berbasis satelit yang dikembangkan oleh departemen pertahanan Amerika Serikat. Sebenarnya GPS memiliki nama asli yaitu NAVSTAR (*Navigational Satellite Timing and Ranging*). GPS dapat memberikan informasi berupa posisi, kecepatan, dan informasi secara teliti di segala cuaca.

2.3 OpenStreetMap

OpenStreetMap merupakan sebuah peta dunia digital gratis yang dapat diubah sesuai dengan kebutuhan, dan dirilis dengan lisensi terbuka (*open source*). OpenStreetMap mengizinkan kita untuk menampilkan, merubah, dan menggunakan data geografis berupa kolaborasi jalan dari berbagai belahan bumi.[5]

OpenStreetMap bukan hanya proyek perangkat lunak, para pengembang selalu keluar dari balik layar komputer dan mengamati kota-kota dan desa untuk membuat dan memperbarui data peta. Jika anda memiliki perangkat GPS, anda juga dapat menyumbang catatan/rekaman perjalanan(tracking). OpenStreetMap juga dapat menampilkan data peta dari Citra Aerial Yahoo! Anda dapat mulai mengedit peta ini secara langsung menggunakan editor online "Potlatch", atau mengunduh aplikasi desktop JOSM. [5]

2.4 Algoritma Floyd-Warshall

Algoritma *floyd-warshall* adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu. [4]

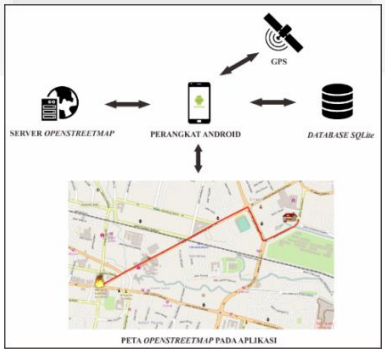
2.5 Metode Simple Additive Weighting (SAW)

Proses pengambilan keputusan adalah memilih suatu alternatife. Metode SAW sering juga dikenal istilah metode penjumlahan terbobot. Konsep dasar merode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat dibandingkan dengan semua rating alternatif yang ada. [7]

3. Perancangan dan Analisis Sistem

3.1 Gambaran Umum Sistem

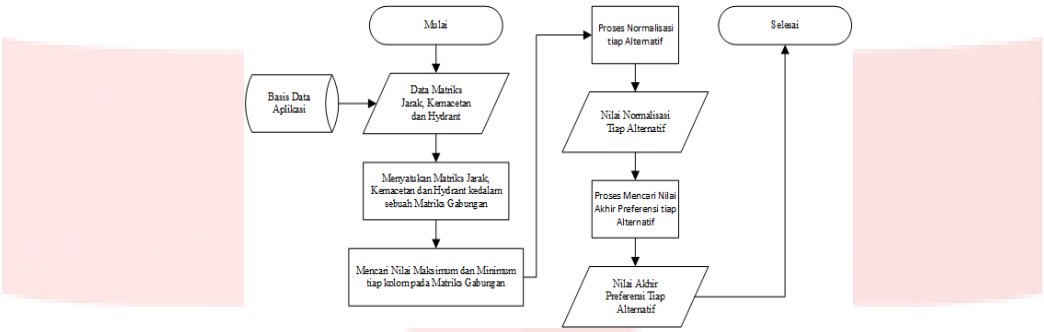
Berikut adalah gambaran umum sistem dari aplikasi pencarian rute untuk mobil pemadam kebakaran berbasis android dengan menggunakan algoritma *Floyd-Warshall* ini :



Gambar 3. 1 Gambaran Umum Sistem

Pada gambar 3.1, perangkat android akan mengunduh peta dari server openstreetmap yang kemudian akan digunakan sebagai media petunjuk informasi mengenai rute terbaik kepada pemadam kebakaran dengan mengimplementasikan suatu algoritma pencarian. Aplikasi akan memanfaatkan penggunaan *Global Positioning System* (GPS) untuk mengetahui lokasi pengemudi mobil pemadam kebakaran. Aplikasi akan mengakses *database* yang berisikan *vertex* dan *edge* jalan-jalan yang ada di kota bandung, data lokasi *hydrant*, dan data kemacetan suatu jalan tertentu. Keluaran aplikasi akan berupa petunjuk arah rute optimal berdasarkan algoritma Floyd-Warshall di aplikasi android.

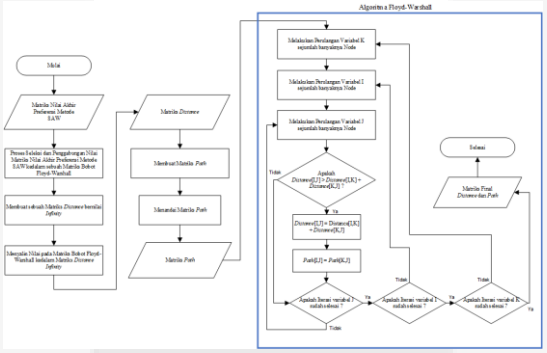
3.2 Perancangan Metode Simple Additive Weighting (SAW)



Gambar 3. 3 Perancangan Metode SAW

Pada gambar 3.4 dijelaskan bagaimana Metode *Simple Additive Weighting* (SAW) berperan sebagai solusi pengambilan keputusan dari banyaknya kriteria pada aplikasi ini seperti Jarak, Kemacetan dan letak *Hydrant* di suatu jalan. Data Matriks Jarak, Kemacetan dan *Hydrant* diperoleh dari basis data SQLite yang tertanam pada aplikasi. Kemudian tiga matriks yang berbeda tersebut disatukan ke dalam sebuah matriks gabungan yang terdiri dari tiga buah kolom. Kolom pertama adalah kriteria Jarak, kedua adalah kemacetan dan yang terakhir adalah kolom *Hydrant*. Kemudian tiap kolom matriks kolom tersebut dicari nilai maksimum dan minimumnya karena nilai tersebut akan dipakai sebagai pembilang dan penyebut pada proses Normalisasi pada metode SAW. Setelah proses Normalisasi, Metode SAW akan mengerjakan proses menghitung Nilai Akhir setiap alternatif. Proses ini mengalikan nilai yang didapat pada proses Normalisasi dengan Bobot Preferensi tiap alternatif. Setelah itu akan didapatkan sebuah matriks akhir hasil dari pengolahan dari tiga buah matriks yang berbeda. Matriks inilah yang kemudian akan menjadi bahan acuan dalam proses pencarian rute dengan menggunakan algoritma Floyd-Warshall.

3.3 Perancangan Algoritma Floyd-Warshall



Gambar 3. 2 Perancangan Algoritma Floyd-Warshall

Setelah proses SAW selesai, maka nilai matriks akhir hasil olahan metode tersebut akan digunakan oleh Algoritma Floyd-Warshall. Pada gambar 3.5 dijelaskan bahwa matriks nilai akhir preferensi akan diseleksi dan digabung kedalam sebuah matriks bobot. Pada intinya, proses ini hanya mengambil nilai-nilai yang ada pada matriks nilai akhir preferensi karena ada kemungkinan selain nilai-nilai tersebut terdapat nilai tak hingga pada matriks nilai akhir preferensi.

Setelah itu, akan dibuat sebuah matriks yang besarannya menyesuaikan dengan matriks bobot dan isi nilai matriks tersebut diisi dengan tak hingga. Nilai pada matriks bobot tersebut akan disalin kedalam matriks *infinity* menyesuaikan dengan posisi baris dan kolomnya. Lalu kita akan mendapatkan sebuah matriks *Distance* yang nantinya akan menjadi bahan olahan utama pada inti algoritma Floyd-Warshall. Kemudian akan dibuat sebuah matriks *path* yang berguna sebagai penentu keputusan rute-rute jalan yang akan diambil untuk sampai ke tujuan.

Setelah sistem mendapatkan nilai-nilai pada matriks *Distance* dan matriks *Path*, maka algoritma Floyd-Warshall siap untuk dijalankan. Di dalam algoritma ini, terdapat tiga buah proses iterasi utama yang saling berkaitan sesuai dengan diagram yang terlihat pada gambar 3.5. Banyaknya iterasi yang akan dikerjakan disesuaikan dengan banyaknya *node* yang terdaftar pada basisdata. Kemudian akan dilakukan proses pengecekan berikut ini :

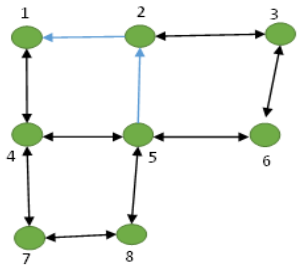
1. ***Distance*[I,J] > *Distance*[I,K] + *Distance*[K,J]** ,
   Apakah nilai *Distance*[I,J] lebih besar dari *Distance*[I,K] + *Distance*[K,J] ?
   (*Note* : Nilai **K** mengikuti iterasi yang sedang berlangung)
   Jika ya, maka nilai *Distance*[I,K] + *Distance*[K,J] berpindah ke variable Distance[I,J].
   Jika tidak, maka tidak ada perubahan pada nilai Distance[I,J].
2. Sistem akan melakukan proses ***Path*[I,J] = *Path*[K,J]**. Nilai *Path*[K,J] akan menggantikan nilai *Path*[I,J].

Proses inti algoritma diatas akan menyesuaikan dengan tiga buah proses iterasi yang sudah di inisialisasi diawal proses. Setelah semua iterasi terpenuhi, maka akan didapat matriks final *Distance* dan *Path*. Kedua matriks inilah yang akan digunakan sebagai penentu akhir pengambilan keputusan rute-rute yang akan diambil hingga sampai ke tempat tujuan.

4. Implementasi dan Pengujian Sistem

4.1 Pengujian Metode dan Algoritma

Berikut ini akan dilakukan pengujian perbandingan perhitungan manual pencarian rute dengan rute yang dihasilkan oleh aplikasi. Pengujian ini bertujuan untuk membuktikan kebenaran dari sistem yang bekerja pada aplikasi. Untuk skenario pengujian, akan diambil potongan dari peta yang terdiri dari 8 *node.*



Gambar 4. 1 Peta Jalan Pengujian

Gambar 4.1 merupakan potongan peta jalan untuk pengujian yang berisi 8 node yang saling terhubung. Terdapat dua jenis jalan pada peta tersebut yaitu jalan satu arah dan jalan dua arah.

Tabel 4. 2 Keterangan Nama Jalan Pengujian

No	Alternatif	Nama Jalan	Jarak (KM)	Kemacetan	*Hydrant*
1	1 → 2	Jl. Belakang Pasar	0,184	0,1	0
2	1 → 4	Jl. Gardu Jati	0,174	0,1	0
3	2 → 3	Jl. Belakang Pasar 1	0,165	0,1	0
4	2 → 5	Jl. Durman	0,18	0,5	0
5	3 → 6	Jl. Babatan	0,224	0,1	0
6	5 → 4	Jl. Ence Ajis	0,15	0,8	0
7	4 → 7	Jl. Astana Anyar	0,154	0,1	1
8	8 → 5	Jl. Durman 1	0,158	0,1	0
9	6 → 5	Jl. Ence Ajis 1	0,163	0,1	1
10	7 → 8	Jl. Jendral Sudirman	0,157	0,1	0

Tabel 4.12 merupakan keterangan jalan dari *node* yang ada pada peta pengujian.

4.1.1 Pengujian Metode SAW

Metode Simple Additive Weighting (SAW) merupakan metode yang akan digunakan sebagai alat membentuk sebuah matriks yang terbentuk dari tiga buah matriks utama yaitu jarak, kemacetan dan *hydrant*. Proses metode SAW meliputi penggabungan matriks, mencari nilai *max* dan *min*, normalisasi, dan mencari nilai akhir preferensi tiap alternatif.

Proses SAW dimulai dengan menggabungkan tiga matriks utama yang telah diinisialisasi ke dalam sebuah matriks SAW gabungan. Selanjutnya di setiap kriteria yang ada, dicari nilai maksimum dan minimumnya. Karena nilai maksimum atau minimum tersebut akan menjadi bahan dalam proses normalisasi. Metode SAW juga memerlukan sebuah bobot preferensi masing-masing kriteria yang ada.

Setelah itu, dilakukan proses normalisasi SAW yang mana masing-masing nilai alternatif tersebut mengalami proses pembagian oleh nilai maksimum dan minimum disesuaikan dengan jenis alternatif. Jarak dan

kemacetan termasuk jenis *cost*, maka dilakukan pembagian dengan nilai minimum jarak dan kemacetan sedangkan *hydrant* termasuk *benefit*, maka dilakukan pembagian dengan nilai maksimum *hydrant*.

Selanjutnya, dilakukan sebuah proses untuk mendapatkan nilai akhir preferensi masing-masing alternatif yang mana nilai alternatif masing-masing kriteria dikalikan dengan bobot preferensinya masing-masing. Maka didapatkan sebuah matriks akhir metode SAW seperti yang terlihat pada tabel 4.6. Matriks tersebut yang akan menjadi bahan dasar melanjutkan proses perhitungan dengan algoritma Floyd-Warshall.

Tabel 4. 6 Nilai Akhir Preferensi Metode SAW

No	Alternatif	Nilai Akhir Preferensi
1	1 → 4	0,7076
2	2 → 1	0,7310
3	3 → 2	0,7545
4	3 → 6	0,6348
5	4 → 1	0,7076
6	4 → 5	0,5375
7	4 → 7	0,9870
8	5 → 2	0,4767
9	5 → 4	0,5375
10	5 → 6	0,9601
11	5 → 8	0,7747
12	6 → 3	0,6348
13	6 → 5	0,9601
14	7 → 4	0,9870
15	7 → 8	0,7777
16	8 → 5	0,7747
17	8 → 7	0,7777

4.1.2 Pengujian Algoritma Floyd-Warshall

Setelah proses perhitungan dengan metode SAW selesai, Algoritma Floyd-Warshall akan memulai prosesnya dengan matriks nilai akhir preferensi yang telah didapatkan.

Tabel 4. 7 Matriks *Distance*

Node	1	2	3	4	5	6	7	8
1	∞	∞	∞	0,7076	∞	∞	∞	∞
2	0,731	∞	0,7545	∞	∞	∞	∞	∞
3	∞	0,7545	∞	∞	∞	0,6348	∞	∞
4	0,7076	∞	∞	∞	0,5375	∞	0,987	∞
5	∞	0,4767	∞	0,5375	∞	0,9601	∞	0,7747
6	∞	∞	0,6348	∞	0,9601	∞	∞	∞
7	∞	∞	∞	0,987	∞	∞	∞	0.7777
8	∞	∞	∞	∞	0,7747	∞	0,7777	∞

Pada tabel 4.7 terdapat sebuah tabel matriks *distance* yang berisi kumpulan nilai akhir preferensi hasil olahan metode SAW yang telah disesuaikan dengan letak dan posisi seperti yang terdapat pada tabel 4.6.

Nilai yang terdapat pada matriks *Distance*, **bukan** jarak total dari *node* asal ke *node* tujuan karena nilai tersebut adalah hasil perhitungan gabungan antara matriks jarak, kemacetan dan *hydrant*. Matriks *Distance* hanya digunakan sebagai media perantara perubahan pada matriks *path*. Jika terdapat perubahan pada matriks *Distance*, maka matriks *path* juga akan mengalami perubahan yang sama dengan matriks *Distance*.

Tabel 4. 8 Matriks *Path*

Node	1	2	3	4	5	6	7	8
1	0	0	0	1	0	0	0	0
2	2	0	2	0	0	0	0	0
3	0	3	0	0	0	3	0	0
4	4	0	0	0	4	0	4	0
5	0	5	0	5	0	5	0	5
6	0	0	6	0	6	0	0	0
7	0	0	0	7	0	0	0	7
8	0	0	0	0	8	0	8	0

Algoritma Floyd-Warshall juga membutuhkan sebuah matriks Path sebagai media penentuan jalur routing. Matriks Path dapat dilihat pada tabel 4.8. Nilai matriks path menyesuaikan dengan nilai dan kondisi pada matriks nilai akhir preferensi. Apabila suatu node memiliki relasi dengan node yang lainnya, maka matriks memiliki nilai sedangkan nilai 0 untuk node yang tidak memiliki relasi dengan node lainnya.

Algoritma Floyd-Warshall memiliki tiga buah iterasi dan setiap iterasi menyesuaikan dengan banyaknya *node* yang ada. Jika disesuaikan pada skenario uji kali ini, maka iterasi akan berjalan selama $8^3 = 512$ kali iterasi.

Selama proses iterasi berlangsung, akan ada perubahan yang terjadi yang disebabkan oleh algoritma ini. Kotak berwarna kuning pada matriks *Distance* atau matriks path yang terlampir pada bagian lampiran menunjukkan bahwa adanya perubahan nilai yang diakibatkan oleh proses algoritma yang berlangsung.

Proses iterasi akan berhenti setelah proses iterasi variabel K telah dilakukan sebanyak 8 kali menyesuaikan dengan banyaknya *node* pada peta. Matriks *Distance* dan Matriks *Path* iterasi terakhir yang akan digunakan sebagai panduan sistem melakukan proses *routing*.

Tabel 4. 9 Matriks *Final Distance*

Node	1	2	3	4	5	6	7	8
1	1,4152	1,7218	2,4763	0,7076	1,2451	2,2052	1,6946	2,0198
2	0,731	1,509	0,7545	1,4386	1,9761	1,3893	2,4256	2,7508
3	1,4855	0,7545	1,2696	2,1324	1,5949	0,6348	3,1194	2,3696
4	0,7076	1,0142	1,7687	1,075	0,5375	1,4976	0,987	1,3122
5	1,2077	0,4767	1,2312	0,5375	1,075	0,9601	1,5245	0,7747
6	2,1203	1,3893	0,6348	1,4976	0,9601	1,2696	2,4846	1,7348
7	1,6946	2,0012	2,7557	0,987	1,5245	2,4846	1,5554	0,7777
8	1,9824	1,2514	2,0059	1,3122	0,7747	1,7348	0,7777	1,5494

Tabel 4.9 merupakan matriks *final Distance* yang didapatkan dari hasil iterasi terakhir dari total 8 kali proses iterasi yang ada. Matriks ini adalah alasan terbentuknya matriks *path.* Jika matriks *Final Distance* mengalami perubahan, maka matriks *Final Path* juga ikut berubah datanya. Nilai yang terdapat pada matriks *Final Distance*, **bukan** jarak total dari node asal ke node tujuan karena nilai tersebut adalah hasil perhitungan gabungan antara matriks jarak, kemacetan dan hydrant. Matriks *Final Distance*, hanya sebagai media perantara perubahan pada matriks path.

Tabel 4. 10 Matriks *Final Path*

Node	1	2	3	4	5	6	7	8
1	4	5	2	1	4	5	4	5
2	2	3	2	1	4	3	4	5
3	2	3	6	5	6	3	4	5
4	4	5	2	5	4	5	4	5
5	2	5	2	5	4	5	4	5
6	2	3	6	5	6	3	4	5
7	4	5	2	7	4	5	8	7
8	2	5	2	5	8	5	8	5

Tabel 4.10 merupakan matriks *final Path* yang didapatkan dari hasil iterasi terakhir dari total 8 kali proses iterasi yang ada. Nilai-nilai pada matriks ini menyesuaikan dengan perubahan yang terjadi pada Matriks *Final Distance.* Matriks *Final Path* yang akan menjadi acuan dalam melakukan proses *routing*.

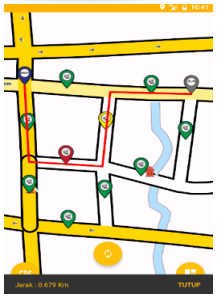
4.1.3 Pengujian Pencarian Rute

Jl. Belakang Pasar -> Jl. Babatan (*Node* 1 -> *Node* 3)

Tabel 4. 11 Hasil Pengujian Pencarian Rute 1

No	Rute Hasil Manual	Rute Hasil Aplikasi	Jarak Tempuh Manual (Km)	Jarak Tempuh Aplikasi (Km)
1	1 → 4 → 5 → 2 → 3	1 → 4 → 5 → 2 → 3	0,679	0,679

Pada tabel 4.11 berdasarkan pengujian yang telah dilakukan untuk kasus pencarian rute pertama dengan metode perhitungan manual menghasilkan rute dan jarak tempuh yang sama dengan rute yang dihasilkan oleh aplikasi.



Gambar 4. 2 Rute 1 Hasil Keluaran Aplikasi

Gambar 4.2 adalah hasil pencarian rute pertama yang dihasilkan oleh aplikasi. Rute yang dihasilkan adalah node 1→ node 4 → node 5 → node 2 → node 3.

Gambar 4. 3 Rute 1 Berdasarkan Pemahaman Supir Pemadam Kebakaran

Gambar 4.3 adalah Rute berdasarkan pemahaman supir pemadam kebakaran. Apabila dibandingkan dengan rute yang disesuaikan dengan "kebiasaan" supir pemadam kabakaran dalam mengambil keputusan pengambilan rute perjalanan menuju TKP, maka akan jauh berbeda dengan rute yang disarankan oleh aplikasi. Rute yang akan diambil oleh supir pemadam kebakaran adalah 1 -> 2 -> 3 (Jl. Belakang Pasar, Jl. Belakang Pasar 1, dan Jl. Babatan). Jl. Belakang Pasar -> Jl. Ence Ajis (*Node* 1 -> *Node* 5)

5. Kesimpulan

Dari hasil pengujian dan analisis yang telah dilakukan maka dapat disimpulkan menjadi beberapa hal sebagai berikut :

1. Metode *Simple Additive Weighting* (SAW) dan Algoritma Floyd-Warshall dapat dirancang dan diimplementasikan ke dalam sebuah aplikasi berbasis android untuk pencarian rute optimal.
2. Berdasarkan hasil pengujian metode dan algoritma terhadap 3 rute pada skenario uji, dihasilkan hasil yang sama antara perhitungan manual dan hasil keluaran yang dihasilkan oleh aplikasi.
3. Algoritma Floyd-Warshall dapat diterapkan untuk proses pencarian rute, akan tetapi tidak akan cukup efektif apabila diterapkan untuk proses pencarian rute dengan cakupan luas daerah yang cukup besar, karena hasil yang disarankan tidak akan cukup optimal.

Beberapa hal yang dapat menjadi masukan untuk penelitian selanjutnya :

1. Aplikasi dapat dikembangkan dengan data kemacetan yang *realtime*.
2. *Database* dilengkapi dengan jumlah node yang lebih banyak.
3. Penambahan data jalan yang dapat dilalui oleh pemadam kebakaran.

Daftar Pustaka

[1] Adipranata, Rudy. Desiree, Fauzi Josephine, Fauzi. Handojo, Andreas. 2008. *Aplikasi Pencarian Rute Optimal Menggunakan Metode Transitive Closure.* Surabaya : Jurnal Universitas Kristen Petra.

[2] Ekadinata, Andree dkk. 2008. *Sistem Informasi Geografis Untuk Pengelolaan Bentang Lahan Berbasis Sumber Daya Alam*. Bogor : ICRAF South East Asia.

[3] Juhara, Zamrony P. 2016 *Panduan Lengkap Pemrograman Android*. Yogyakarta : Penerbit Andi.

[4] Kriswanto, Y. Rudi. Bendi, R. Kristoforus Jawa. Aliyanto, Arif. 2014. *Penentuan Jarak Terpendek Rute Transmusi dengan Algoritma Floyd-Warshall.* Semarang : Jurnal SEMANTIK 2014.

[5] Riyanto. 2010. *Membuat Sendiri Aplikasi Mobile GIS Platform Java ME, Blackberry & Android*. Yogyakarta : Penerbit Andi.

[6] Tanoe, Andre. 2009. *GPS bagi pemula, dasar-dasar penggunaan sehari-hari.*

[7] Utomo, Meriano Setya Dwi. *Penerapan Metode SAW (Simple Additive Weight) Pada Sistem Pendukung Keputusan Untuk Pemberian Beasiswa Pada SMA Negeri 1 Cepu Jawa Tengah.* Semarang : Jurnal Universitas Dian Nuswantoro.