Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.2024118054
Vol. 11, No. 6, Desember 2024, hlm.1185-1196 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

PENINGKATAN PERFORMA KOMPUTASI SISTEM NAVIGASI TRANSPORTASI PUBLIK PADA PERANGKAT BERGERAK MELALUI PENERAPAN TEKNIK KOMPRESI DATA DAN PENYEDERHANAAN GRAF

Aryo Pinandito[*1], Agi Putra Kharisma[2], Muhammad Aminul Akbar[3], Mochamad Chandra Saputra[4]

[1]Universitas Brawijaya, Malang
Email: [1]aryo@ub.ac.id, [2]agi@ub.ac.id, [3]muhammad.aminul@ub.ac.id, [4]andra@ub.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 15 November 2023, diterima untuk diterbitkan: 19 November 2024)

Abstrak

Penelitian ini bertujuan untuk menyelesaikan permasalahan di bidang transportasi terkait penentuan rute perjalanan menggunakan transportasi umum dengan mengembangkan sistem dan aplikasi perangkat bergerak yang mampu merekomendasikan rute perjalanan angkutan umum di kota Malang. Rekomendasi rute perjalanan dihasilkan dari penerapan algoritme Dijkstra dari rute transportasi angkutan umum di Kota Malang yang dimodelkan dalam struktur data graf. Bagian yang akan dinilai pada penelitian ini adalah perbandingan efektivitas penggunaan format JSON terutama pada penghematan *bandwidth*. Teknik kompresi yang digunakan pada penelitian ini adalah Zlib yang dikombinasikan dengan encoding Base64. Analisis juga dilakukan terhadap beban tambahan dan penggunaan memori akibat proses kompresi serta analisis waktu komputasi dan penggunaan memori akibat penyederhanaan struktur data graf melalui pendekatan visual dengan algoritme Douglas-Peucker. Penggunaan dokumen dalam format JSON terbukti efektif digunakan sebagai media komunikasi, penggunaan kompresi data dapat menghemat penggunaan *bandwidth* untuk mendistribusikan data di Internet sebesar 64,61%, dan memberikan beban tambahan pada memori yang diperlukan oleh server untuk kompresi data sebesar 0,44%. Selain itu, dengan penyederhanaan graf secara visual, waktu proses yang diperlukan untuk merekomendasikan rute transportasi menjadi lebih singkat 81,24% serta berkurangnya penggunaan memori hingga 72,99%.

Kata kunci: *API, Dijkstra, Douglas-Peucker, kompresi, optimasi, transportasi*

COMPUTATION PERFORMANCE IMPROVEMENT OF PUBLIC TRANSPORT NAVIGATION SYSTEM ON MOBILE DEVICES THROUGH THE APPLICATION OF DATA COMPRESSION AND GRAPH SIMPLIFICATION TECHNIQUES

Abstract

This research aims to solve problems in the field of transportation related to determining travel plans using public transportation by developing systems and mobile applications based on Google Android that recommend public transport plans in Malang City. The plans are generated from the implementation of Dijkstra's algorithm, which is modeled in the graph data structure. This study determines the effectiveness of using JSON format, the bandwidth saved for distributing data on the Internet from Zlib data compression and Base64 encoding, and analyzes additional memory usage due to compression and the impact of data simplification through a visual approach on transport network graphs with the Douglas-Peucker algorithm. This study suggested that the use of JSON format is effective as a communication medium, and the use of data compression allows clients to save bandwidth usage by up to 64.61% with an additional 0.44% of memory load for data compression. Additionally, visually simplifying the graph data improved the system's performance; it reduced the average processing time to recommend transportation routes by 81.24% and reduced memory usage by up to 72.99%.

Keywords: *API, compression, Dijkstra, Douglas-Peucker, optimization, transportation*

1. PENDAHULUAN

Dengan latar belakang budaya dan pendidikan, Kota Malang telah menjadi salah satu kota dan daerah yang memiliki banyak destinasi wisata dan institusi pendidikan, sehingga Kota Malang menjadi salah satu kota yang sangat menarik bagi para pelajar dan wisatawan dari seluruh penjuru Indonesia dan mancanegara. Populasi dan mobilitas penduduk Kota Malang hampir tidak pernah sepi sepanjang tahun, yang didominasi oleh para wisatawan di akhir minggu

dan musim liburan, dan didominasi oleh para pelajar serta mahasiswa di hari-hari normal lainnya.

Dengan banyaknya jumlah penduduk di Kota Malang, hal-hal terkait mobilitas dan transportasi umum menjadi hal penting bagi masyarakat dalam memenuhi kebutuhan dan pekerjaan sehari-hari. Penelitian ini dilatarbelakangi oleh permasalahan di bidang transportasi, khususnya permasalahan yang dialami oleh penduduk dan pengunjung di Kota Malang yang tidak memiliki kendaraan pribadi atau ingin menggunakan alat transportasi umum untuk menunjang pergerakan dan mobilitas sehari-harinya. Hal tersebut utamanya ditujukan bagi para pendatang dan wisatawan yang sedang berkunjung atau berada di Kota Malang untuk mendatangi beberapa tempat dan lokasi tertentu dengan menggunakan moda transportasi angkutan umum. Bagi para pendatang dan wisatawan yang baru berkunjung atau belum lama tinggal di Kota Malang, tidak mengetahui informasi terkait rute transportasi angkutan umum. Ditambah lagi, reliabilitas atau keandalan angkutan umum yang sedang berjalan tidak dapat dipastikan jadwal dan lokasi kedatangannya di tempat-tempat pemberhentian tertentu. Sehingga, masyarakat umum saat ini tidak mendapatkan jaminan keandalan waktu dan biaya terkait transportasi angkutan umum untuk mendukung mobilitas sehari-hari.

Topik penelitian terkait permasalahan umum yang diangkat dan diusulkan untuk diselesaikan dalam penelitian ini terkait pemilihan rute perjalanan moda transportasi angkutan umum yang tersedia di Kota Malang. Solusi yang diusulkan dalam penelitian ini diutamakan pada penciptaan suatu alat dalam bentuk sistem perangkat lunak berbasis web dan perangkat bergerak Google Android yang mampu memberikan rekomendasi dan menampilkan rute perjalanan dari suatu tempat ke tempat lain di Kota Malang dengan moda transportasi umum dengan lebih mudah melalui perangkat bergerak yang dimiliki oleh pengguna.

Dalam bidang ilmu teknik informatika, domain rute perjalanan angkutan umum merupakan permasalahan optimasi yang dapat dimodelkan dengan struktur data graf. Penentuan rute perjalanan dari suatu lokasi menuju lokasi lainnya dapat diselesaikan menggunakan metode rute terpendek dengan algoritme Dijkstra yang diterapkan pada struktur data graf tersebut. Dengan algoritme Dijkstra, rute perjalanan dapat direkomendasikan serta dapat dioptimalkan berdasarkan parameter nilai biaya perpindahan titik dari suatu titik tertentu; dalam konteks ini, jarak tempuh terdekat atau ongkos penggunaan angkutan umum yang terendah. Dalam situasi tertentu, misalnya kemacetan yang terjadi di titik-titik tertentu, rute perjalanan yang paling pendek bisa jadi tidak merepresentasikan waktu tempuh terpendek untuk menuju lokasi yang diinginkan. Selain itu, algoritme Dijkstra merupakan algoritme yang bersifat *exhaustive*, di mana algoritme tersebut mengevaluasi semua kemungkinan yang ada dalam memberikan rekomendasi rute perjalanan terpendek, sehingga jika algoritme tersebut diterapkan pada data graf yang besar akan memerlukan waktu yang lama. Sehingga, rekayasa terhadap implementasi algoritme Dijkstra dalam penelitian ini perlu diteliti lebih dalam guna mendapatkan ketepatan dan kecepatan proses yang optimal dalam sebuah sistem yang melibatkan perangkat bergerak terkait pemberian rekomendasi rute perjalanan dengan angkutan umum di Kota Malang.

Penelitian sebelumnya yang dilakukan oleh Pinandito (2020) terkait deteksi area di sekitar rute perjalanan dengan penyederhanaan rute perjalanan menggunakan algoritme Douglas-Peucker, dapat terlihat bahwa metode *line simplification* Douglas-Peucker tersebut mampu meningkatkan efisiensi pemrosesan data yang sangat signifikan hingga 87 kali lipat lebih cepat dari pemrosesan yang dilakukan secara normal tanpa penyederhanaan dan mampu mempertahankan tingkat kepresisian area yang diperoleh di atas 92,6%. Penelitian ini bertujuan untuk menghasilkan sebuah perangkat lunak aplikasi pada perangkat bergerak Android yang mampu memberikan rekomendasi rute perjalanan angkutan umum yang menerapkan algoritme Dijkstra di mana data Graf yang dioptimasikan dengan mengunakan teknik *line simplification* Douglas-Peucker.

Permasalahan efisiensi dalam proses pemberian rekomendasi melalui penyederhanaan data input menggunakan algoritme Douglas-Peucker tersebut kemudian diangkat menjadi rumusan masalah dan hipotesis yang diselesaikan dalam penelitian ini, yaitu:

- Bagaimana desain arsitektur dari sistem informasi transportasi publik di kota Malang yang dapat digunakan pada aplikasi mobile, berbasis *mobile web* dan Google Android dalam memberikan rekomendasi jalur-jalur transportasi umum yang digunakan?
- Bagaimana karakteristik performa dari sistem informasi yang dirancang dengan menerapkan algoritme *shortest-path* Dijkstra dan algoritme *line simplification* Douglas-Peucker dari sudut pandang waktu pemrosesan dan penggunaan memori untuk aplikasi *mobile* berbasis *mobile web* dan Google Android dalam mengoptimalkan performa komputasi untuk merekomendasikan rute perjalanan pengguna dengan transportasi umum?
- Sejauh mana algoritme penyederhanaan garis Douglas-Peucker mampu meningkatkan efisiensi proses penentuan rute perjalanan pada algoritme Dijkstra?
- Sejauh mana dampak dari penerapan metode Douglas-Peucker terhadap beban kerja sistem yang pada aplikasi perangkat bergerak maupun *server*?

Penelitian ini memberikan kontribusi berupa rancangan sistem informasi berbasis *client-server* berbasis API yang dapat digunakan oleh aplikasi perangkat bergerak berbasis *mobile web* dan *native* Android. Selain itu penelitian ini juga berkontribusi

dalam memberikan gambaran bagaimana algoritme Dijkstra dapat diterapkan dalam merekomendasikan rute perjalanan angkutan umum di Kota Malang serta meningkatkan performa proses komputasinya dalam memberikan rekomendasi perjalanan angkutan umum dengan melibatkan algoritme penyederhanaan garis Douglas-Peucker.

2. TINJAUAN PUSTAKA

2.1 Algoritme Dijkstra dalam Penentuan Rute Terpendek Transportasi Publik

Permasalahan dalam penentuan rute terpendek dapat diselesaikan oleh berbagai algoritme, seperti algoritme Dijkstra dan A*. Penelitian yang dilakukan oleh Rachmawati (2020) mengungkapkan bahwa algoritme Dijkstra dan A* dapat digunakan untuk pencarian rute terpendek dalam lingkup kota dengan performa yang hampir sama saja, namun algoritme A* memiliki performa lebih baik pada skala atau cakupan wilayah yang lebih besar. Algoritme Dijkstra adalah algoritme yang umum digunakan untuk menyelesaikan masalah terkait rute perjalanan dalam sebuah peta (Zhou, 2019) yang dimodelkan dengan struktur data graf. Penelitian yang dilakukan Salem (2022) menerapkan algoritme Dijkstra dalam menentukan jadwal rute penerbangan pesawat termasuk beberapa penelitian serupa terkait sistem navigasi, baik *indoor* maupun *outdoor*, yang menerapkan algoritme Dijkstra juga banyak diteliti oleh beberapa penelitian lainnya (Parulekar, 2013; Ruan, 2014; Xu, 2015; Singh, 2018; Sujatha, 2020; Samah, 2020; Friaswanto, 2021; Nayagam, 2023). Dengan demikian, algoritme Dijkstra ini tepat digunakan sebagai metode yang digunakan untuk menentukan rute perjalanan angkutan umum di Kota Malang yang diteliti dalam penelitian ini. Algoritme Dijkstra ini seringkali dikombinasikan dengan metode A* dalam penentuan titik terdekat untuk mempercepat pemrosesan untuk model peta dalam skala besar (Rachmawati, 2020) dan permainan gim komputer (Foead, 2023). Walaupun demikian, algoritme A* ini memiliki beberapa kelemahan dalam menentukan rute alternatif perjalanan yang memiliki *cost* yang sama namun dengan rute yang sedikit memutar seperti yang ada pada rute perjalanan angkutan umum yang cenderung demikian. Optimasi penentuan rute perjalanan angkutan umum dengan algoritme Dijkstra yang adaptif seperti yang diteliti dalam studi kasus penelitian ini belum pernah diteliti sebelumnya, sehingga menjadi urgensi dan dasar mengapa penelitian ini perlu dilakukan pendalaman dan uji coba secara langsung di lapangan.

2.2 Penerapan Algoritme Douglas-Peucker dalam Penyederhanaan Visual Rute Transportasi

Di bidang grafika komputer dan pemetaan (kartografi), algoritme Ramer Douglas Peucker atau algoritme Ramer-Douglas-Peucker (Douglas, 1973; Visvalingam, 2007; Jilin, 2018; Lakhera, 2020; Wang, 2021; Guedri, 2021) telah lama digunakan untuk mengurangi jumlah verteks (titik-titik) yang diperlukan dalam sebuah gambar berbasis vektor, seperti gambar berformat Scalable Vector Graphics (SVG) dengan mempertahankan bentuk grafik yang disederhanakan guna meningkatkan efisiensi dalam pemrosesan datanya (Bhimanjaya, 2023). Metode Douglas-Peucker ditingkatkan performanya dalam penelitian yang dilakukan oleh Hershberger (1992) dan Liu (2020). Beberapa alternatif algoritme untuk menyederhanakan bentuk dan garis seperti metode *line simplification* lainnya yang telah dikembangkan oleh beberapa peneliti seperti Visvalingam-Whyatt (Visvalingam, 1992), Reumann-Witkam (Reuman, 1974), Opheim (Opheim, 1981), Lang (Shi, 2006), dan Zhao–Saalfeld (Zao, 2008).

Pada prinsipnya, algoritme Douglas-Peucker menyederhanakan representasi garis kurva pada grafika komputer dengan tujuan mengurangi jumlah verteks yang diperlukan (Douglas, 1973). Secara lebih spesifik, metode ini mempertahankan titik-titik verteks awal dan verteks akhir dari sebuah kurva, kemudian mencari sekumpulan verteks yang berada di antara verteks awal dan verteks akhir. Aturan yang digunakan adalah verteks yang memiliki jarak terjauh dari batas tertentu akan dipertahankan dan titik-titik verteks lainnya yang tidak relevan dihapus. Proses ini akan dilakukan secara rekursif hingga seluruh verteks memenuhi batas yang ditentukan.

Penelitian sebelumnya yang dilakukan oleh Pinandito (2020) terkait deteksi area di sekitar rute perjalanan dengan penyederhanaan rute perjalanan menggunakan algoritme Douglas-Peucker, dapat terlihat bahwa metode *line simplification* Douglas-Peucker tersebut mampu meningkatkan efisiensi pemrosesan data yang sangat signifikan hingga 87,8 kali lebih cepat dari pemrosesan yang dilakukan secara normal dan mampu mempertahankan tingkat kepresisian area yang diperoleh melalui proses penyederhanaan di atas 92,6%. Sehingga penelitian ini diharapkan dapat memenuhi tujuannya untuk menghasilkan sebuah perangkat lunak aplikasi pada perangkat bergerak Google Android yang mampu memberikan rekomendasi rute perjalanan angkutan umum yang menerapkan algoritme Dijkstra di mana data Graf yang proses komputasinya dioptimasikan dengan menggunakan teknik *line simplification* Douglas-Peucker. Selain itu, beban kerja pemrosesan terhadap penerapan metode Douglas-Peucker ini dianalisis lebih lanjut untuk mengetahui seberapa besar dampak dari proses penyederhanaan terhadap sistem yang dilakukan dengan pengujian dan analisis secara empiris dalam penelitian ini.

2.3 Penggunaan API dalam Distribusi Data di Internet

Integrasi lintas sistem secara definisi adalah hubungan antar dua atau lebih aplikasi, baik itu

berbasis *web*, *desktop*, atau *mobile* yang terhubung melalui API yang disediakan oleh masing-masing aplikasi, sehingga sistem-sistem yang terhubung dengan API tersebut dimungkinkan untuk dapat saling bertukar data (Jones, 2021). Penggunaan API pada domain sistem transportasi publik atau juga dapat disebut paratransit dapat direalisasikan dalam bentuk RESTful API (Kharisma, 2017; Kharisma, 2019). Dengan adanya API, otomasi proses dapat dilakukan antar sistem. Dengan mengintegrasikan sistem yang berbeda melalui API, skalabilitas suatu sistem dapat ditingkatkan dengan lebih baik untuk menangani data dan kebutuhan yang lebih besar. Selain itu, dengan menjamin kualitas sistem untuk tiap entitas yang berinteraksi dengan menggunakan API, kesalahan-kesalahan yang mungkin terjadi dapat diminimalisir, dihindari, dan/atau diisolir (Sarnacki, 2013). Sehingga, kesalahan dan pemulihan ulang layanan suatu API dapat identifikasi dan diselesaikan dalam waktu yang lebih singkat.

Terlepas dari baik dan buruknya penggunaan API sebagai antarmuka komunikasi lintas sistem, pada dasarnya, sebuah API berfungsi sebagai façade yang menyembunyikan kompleksitas sistem (MDN, 2023; Rosner, 2017) dan proses yang bekerja di balik layar. Pengguna API, hanya perlu mengetahui apa yang diperlukan oleh sebuah API, dan apa yang akan dihasilkan oleh API tersebut sehingga pengguna API dapat menggunakan atau mengonsumsinya dengan baik. Format dokumen yang umum digunakan di Internet dalam berkomunikasi dengan menggunakan API adalah Javascript Object Notation (JSON) dan Extensible Markup Language (XML) (Cooksey, 2014). Kedua format dokumen tersebut merupakan dokumen dengan struktur data teks biasa (*plain text*) yang mudah untuk dibaca dan digunakan oleh hampir semua sistem yang ada dan dapat dengan mudah didistribusikan melalui Internet dengan protokol HTTP.

Dari kedua format data XML dan JSON, hanya format XML yang secara *default* mampu menampung data biner yang direpresentasikan (di-*encode*) dengan menggunakan notasi String Base64 (Rein, 1998). Walaupun demikian, data dalam notasi String Base64 dapat dijadikan nilai atribut sebuah elemen dokumen JSON. Sehingga, format XML dan JSON menjadi format yang sangat populer untuk digunakan dalam mendistribusikan data lintas sistem di Internet. XML memiliki kemampuan dan kompatibilitas yang lebih baik dalam membawa data dan didesain untuk kesederhanaan, generalitas, dan usabilitas data yang dapat digunakan di Internet. Di sisi lain, JSON mengusung format data berorientasi pada objek yang mengedepankan kesederhanaan format data dan bersifat independen terhadap bahasa pemrograman yang digunakan. JSON dibuat berdasarkan bahasa pemrograman JavaScript yang sangat mudah untuk digunakan dan dokumen yang diwujudkan dalam format JSON ini sangat mudah diterapkan untuk merepresentasikan data dalam bentuk objek (Kothari, 2023).

2.4 Kompresi Data dengan Zlib

Kompresi data dalam bentuk teks melibatkan perubahan representasi sebuah file, di mana luaran kompresi yang dihasilkan berbentuk data biner yang menggunakan tempat penyimpanan yang lebih sedikit dibandingkan dengan data yang sama sebelum data tersebut dikompresi (Ferragina, 2009). Data dalam bentuk teks pada umumnya mudah untuk dikompresi (Sayood, 2018). Dengan ide sederhana untuk mengompres data yang dihasilkan oleh API yang bertujuan untuk mengurangi *payload* dengan memanfaatkan kemampuan CPU yang semakin baik menjadi alasan utama sebuah API berbasis web mengompres datanya untuk mendapatkan respon dan performa yang lebih baik (Kanjilal, 2017). Jika suatu data terkompresi perlu didekompresi terlebih dahulu sebelum digunakan oleh aplikasi yang sesungguhnya, maka metode kompresi yang harus digunakan adalah metode kompresi yang bersifat *lossless*, seperti Zip atau GZip yang menggunakan format DEFLATE yang merupakan kombinasi pengodean Huffman (Ferragina, 2009) dan LZ77 (Ziv, 1977). Sehingga, walaupun data yang terkirim telah terkompresi, data tersebut dapat dikembalikan seperti semula secara utuh. Pendekatan yang digunakan oleh metode kompresi ini dapat dikategorikan ke dalam kelompok *symbolwise*, *dictionary-based* dan *transform-based* (Ferragina, 2009; Salomon, 2007). Metode kompresi *lossless* dengan format Zip ini sangat populer digunakan dan banyak didukung secara *native* oleh berbagai sistem operasi yang ada hingga saat ini. Sehingga, pengguna tidak lagi disibukkan dengan menginstal program khusus untuk kompresi atau dekompresi data untuk mengurangi ukuran data yang dimilikinya.

Salah satu bahasa pemrograman yang popular digunakan di Internet untuk menyediakan layanan API adalah bahasa pemrograman PHP. Bahasa pemrograman PHP sangat popular digunakan karena kemudahan menuliskan kode program dan banyaknya fungsi yang disediakan oleh PHP untuk membuat kode program *web* menjadi lebih mudah dan lebih cepat (Shiotsu, 2021). PHP sendiri telah menyediakan beberapa fungsi kompresi berbasis Zlib yang tersedia secara langsung dari paket instalasinya, seperti fungsi PHP gzencode(), gzcompress(), dan gzdeflate(). Dengan banyaknya fungsi *built-in* yang telah disediakan dari paket instalasi PHP tersebut menjadi alasan dan daya tarik mengapa *framework* pengembangan aplikasi *web* banyak dikembangkan dengan menggunakan bahasa pemrograman PHP sebagai basisnya pengembangannya. Begitu pula *framework* pengembangan aplikasi yang digunakan untuk menyediakan layanan API sistem navigasi dalam penelitian ini adalah PHP.

Di sisi *client*, aplikasi Android menjadi salah satu *platform* yang mungkin mengonsumsi data dan

layanan yang disediakan oleh API, sistem operasi Android dengan paket SDK-nya telah menyediakan pula *library* yang dapat digunakan untuk melakukan kompresi dan dekompresi data yang berbasis Zlib. Sehingga, ketersediaan *library* untuk mengurangi besarnya data yang didistribusikan melalui Internet menjadi lebih mudah untuk dikonsumsi oleh aplikasi Android. Di sisi lain, jika aplikasi suatu sistem di sisi *client* dibangun dan dikembangkan dengan bahasa pemrograman JavaScript, terdapat library JavaScript bernama Pako (Tupitsin, 2022) yang dapat disertakan secara langsung ke dalam aplikasi *client* berbasis Javascript untuk mengompres dan mendekompres data dalam format Zlib dengan mudah. Walaupun teknik kompresi dapat dilakukan dan ditujukan untuk mengurangi ukuran data yang didistribusikan di Internet, proses kompresi itu sendiri memberikan beban kerja tambahan, baik itu di sisi *server* untuk melakukan proses kompresi dan di sisi *client* untuk melakukan proses dekompresi data.

Dari banyaknya *library* kompresi yang tersedia untuk digunakan di sisi *client* dan *server* mendorong penelitian ini untuk dilakukan untuk mengetahui sejauh mana beban tambahan yang dibebankan kepada CPU dan memori dalam mengompres data dengan ukuran data yang berbeda-beda. Jika ukuran data yang didistribusikan di Internet berukuran relatif kecil untuk dikompres, maka tidak selayaknya sebuah layanan API membebankan proses yang tidak memberikan manfaat yang lebih baik dari layanan data yang tidak terkompresi. Sehingga, penelitian ini diharapkan dapat memberikan gambaran terkait suatu titik atau posisi di mana beban kerja tambahan proses kompresi dan dekompresi yang akan dibebankan kepada CPU terhadap besarnya data API yang akan didistribusikan melalui jaringan Internet layak untuk dibebankan. Oleh karena data yang dikompresi bersifat biner dan data yang ditransmisikan oleh API adalah data dalam format teks biasa (*plain text*), maka data yang telah dikompresi tadi perlu diubah ke dalam format String Base64 agar dapat ditransmisikan dalam bentuk *yang* hanya mengandung karakter ASCII (Tschabitscher, 2020) sebelum digunakan oleh *client*. Namun, perubahan data dalam format biner ke dalam bentuk String Base64 yang hanya berisi karakter ASCII akan menjadikan ukuran data yang ditransmisikan menjadi sedikit lebih besar daripada ukuran data biner dalam format terkompresinya.

3. METODOLOGI

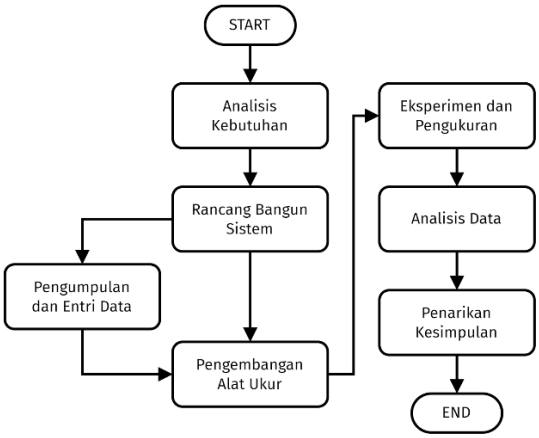
3.1 Data dan Konteks Penelitian

Permasalahan yang diangkat dalam penelitian ini mengangkat topik permasalahan rute perjalanan transportasi umum yang ada di Kota Malang. Data yang digunakan dalam penelitian ini terdiri dari 12 rute transportasi umum yang ada di kota Malang. Data rute perjalanan yang digunakan dalam penelitian ini diperoleh melalui perekaman data di lapangan yang dicatat secara langsung dengan menggunakan perangkat GPS. Rute perjalanan untuk setiap satu jalur transportasi umum tersusun atas urutan sekumpulan titik-titik geospasial sedemikian rupa sehingga membentuk rute perjalanan untuk satu jalur transportasi tersebut dan ditampilkan dalam bentuk peta digital Google Maps. Data rute perjalanan setiap jalur transportasi umum yang diperoleh disimpan ke dalam sistem basis data agar dapat diolah lebih lanjut oleh program atau aplikasi lain yang memerlukannya.

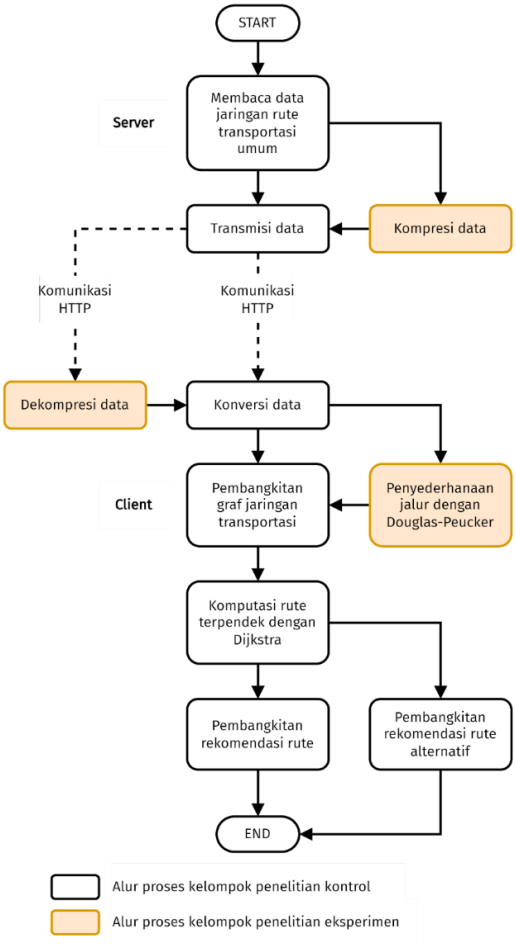
Data yang digunakan dalam sistem ini disimpan di sisi *server* dan dilayani oleh sebuah *web server* Nginx dan *database server* MySQL. Seperti yang telah diuraikan sebelumnya, bahasa pemrograman yang digunakan dalam menyediakan data dan layanan sistem adalah PHP. Sistem yang dikembangkan dalam penelitian ini diuji dalam lingkungan tertutup secara internal di lingkungan pengembangan sistem sebuah laboratorium. Sehingga, pengaruh performa sistem yang disebabkan oleh faktor eksternal seperti kondisi kepadatan lalu lintas data di Internet dapat diabaikan. Secara sederhana, metrik pengukuran terhadap sistem yang dikaji dan diteliti dalam penelitian ini adalah lamanya waktu pemrosesan Dijkstra dan penggunaan memori sistem yang diukur dan dianalisis secara empiris.

3.2 Tahapan Umum Penelitian

Kajian yang dilakukan dalam penelitian ini bersifat implementatif empiris. Kajian-kajian yang dilakukan dalam penelitian memiliki tujuan untuk menyelesaikan permasalahan pengguna di bidang transportasi terkait penentuan rute perjalanan dengan menggunakan moda transportasi umum dengan dirancang dan dibangunnya sistem informasi yang melibatkan aplikasi perangkat bergerak. Selain itu, implementasi yang dilakukan menerapkan beberapa teknik, metode, dan algoritme yang relevan dengan tujuan dan permasalahan, seperti halnya penerapan teknik kompresi Zlib, algoritme Dijkstra, dan algoritme Douglas-Peucker guna memberikan solusi yang optimal dan memberikan pengalaman pengguna yang lebih baik atas performa sistem yang dirancang. Karakteristik dari optimalisasi yang dilakukan dari sistem yang dirancang dikaji dan dianalisis lebih dalam sebagai salah satu bentuk kontribusi penelitian yang diberikan terhadap bidang ilmu rekayasa perangkat lunak, sistem informasi, perangkat bergerak, dan sistem cerdas. Aktivitas-aktivitas yang dilakukan dalam penelitian ini secara garis besar diperlihatkan dalam Gambar 1.



Gambar 1. Tahapan umum aktivitas penelitian



Gambar 2. Implementasi aplikasi perangkat bergerak *mobile web*

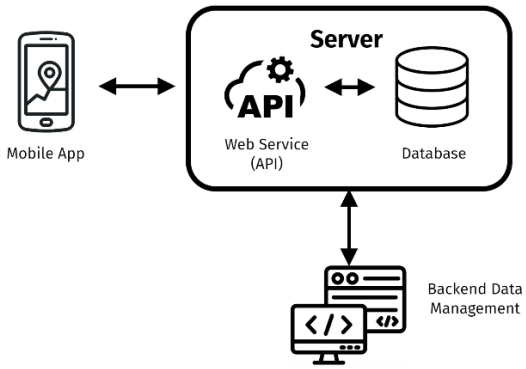
3.3 Alur Kerja Sistem dan Desain Eksperimen

Eksperimen yang dilakukan dalam penelitian ini difokuskan pada implementasi rancangan sistem dan implementasi algoritme Dijkstra dan algoritme penyederhanaan jalur Douglas-Peucker dalam proses komputasi yang dilakukan secara internal dalam merekomendasikan rute perjalanan transportasi umum kepada pengguna serta pengukuran dan analisis data pengukuran dari proses komputasi yang dilakukan di dalam aplikasi perangkat bergerak *mobile web* dan *native* Android. Sistem yang dibangun kemudian disesuaikan pada bagian-bagian tertentu agar proses komputasi yang dilakukan oleh sistem dapat diukur dan memberikan data hasil pengukuran yang tepat. Alur proses kerja sistem yang dikembangkan dalam penelitian ini diperlihatkan dalam Gambar 2.

Berdasarkan Gambar 2, alur kerja pemrosesan data di dalam sistem dilakukan di kedua sisi, *client* dan *server*. Penerapan proses dan metode komputasi data yang dikaji dalam penelitian ini diperlihatkan dalam alur pemrosesan kompresi dan dekompresi data, serta penyederhanaan data jalur transportasi umum sebelum graf jaringan transportasi umum dibangkitkan dan diproses dengan algoritme Dijkstra. Aliran data yang tidak melalui proses kompresi-dekompresi dan tidak dilakukan penyederhanaan rute jalur transportasi dijadikan kelompok kontrol dalam penelitian ini. Sedangkan aliran data yang melalui proses kompresi-dekompresi dan penyederhanaan rute jalur transportasi menjadi kelompok eksperimen penelitian ini. Data yang dihasilkan dari kelompok kontrol menjadi referensi hasil penerapan optimasi proses rekomendasi rute penggunaan transportasi umum.

4. HASIL DAN PEMBAHASAN

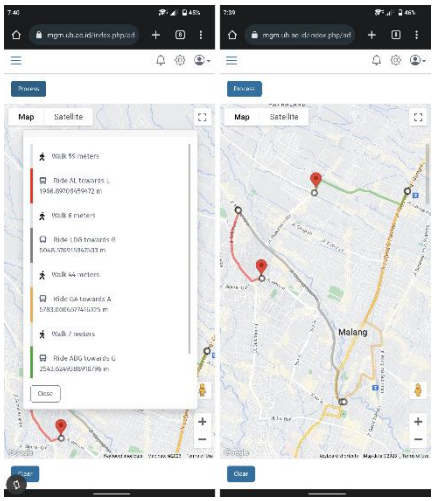
4.1 Rancangan dan Hasil Implementasi Sistem



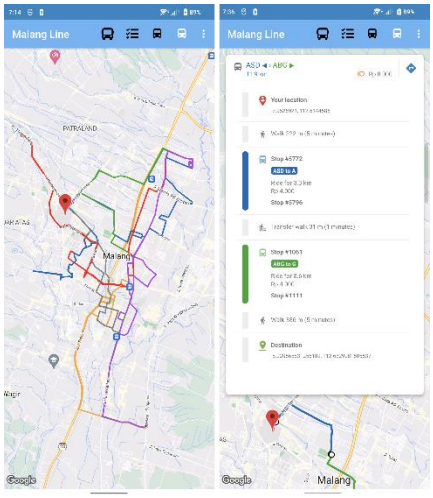
Gambar 3. Arsitektur *client-server* sistem transportasi publik

Aplikasi perangkat bergerak *mobile web* yang dibangun dan dikembangkan dalam penelitian ini diimplementasikan dengan menggunakan teknologi *web*, yakni HTML, CSS, dan JavaScript. Model pengembangan aplikasi *native* Android modern yang banyak direkomendasikan oleh developer-developer di dunia diimplementasikan dengan pola perancangan Model-View-View-Model (MVVM) atau Model-View-Controller (MVC) dengan menggunakan bahasa pemrograman Kotlin dinilai lebih efisien dari bahasa pemrograman Java. Namun, dalam menjaga kesesuaian pengembangan system yang sama dengan pola pengembangan aplikasi *mobile web*, maka aplikasi *native* Android diimplementasikan secara *native* dengan menggunakan bahasa pemrograman Java dan menerapkan pola perancangan Model-View-Controller (MVC) standar dengan fitur yang

serupa dengan aplikasi *mobile web*. Arsitektur *client-server* sistem yang diimplementasikan diperlihatkan dalam Gambar 3. Tampilan tangkapan layar hasil implementasi aplikasi perangkat bergerak *mobile web* dan *native* Google Android yang digunakan di dalam eksperimen untuk dianalisis secara berturut-turut diperlihatkan dalam Gambar 4 dan Gambar 5.



Gambar 4. Implementasi aplikasi perangkat bergerak *mobile web*



Gambar 5. Implementasi aplikasi perangkat bergerak *native* Android

Sistem yang telah dibangun diuji kesesuaian fungsionalitasnya dengan kebutuhan umum yang diperlukan, yakni mampu memberikan rekomendasi jalur transportasi angkutan umum yang dapat digunakan untuk melakukan perjalanan dari satu lokasi ke lokasi lainnya di kota Malang. Pengujian dilakukan dengan menentukan 10 pasang titik-titik lokasi awal dan tujuan perjalanan yang memenuhi kriteria berikut ini:

- Perjalanan jarak dekat yang memungkinkan penggunaan hanya satu jalur transportasi umum;
- Perjalanan jarak dekat yang memungkinkan penggunaan beberapa jalur transportasi umum;
- Perjalanan jarak jauh yang memungkinkan perjalanan dilakukan dengan menggunakan beberapa jalur transportasi umum yang berbeda;
- Perjalanan jarak jauh yang memungkinkan penggunaan satu jalur transportasi umum yang tersedia;
- Perjalanan jarak menengah dari pinggir ke pusat kota dan sebaliknya.

Hasil implementasi dari penerapan algoritme Dijkstra terhadap sistem yang dibangun mampu memberikan rekomendasi jalur transportasi angkutan umum yang dapat digunakan untuk melakukan perjalanan dari satu lokasi ke lokasi lainnya di kota Malang dengan baik dan benar yang diverifikasi secara oleh pelaku pengguna transportasi umum di kota Malang. Tangkapan layar antarmuka pengguna aplikasi yang dikembangkan dalam penelitian ini diperlihatkan dalam Gambar 4 untuk jenis mobile web dan Gambar 5 untuk aplikasi perangkat bergerak *native* Google Android. Tangkapan layar tersebut memperlihatkan rekomendasi dan detail rencana perjalanan atas rekomendasi yang dipilih oleh pengguna.

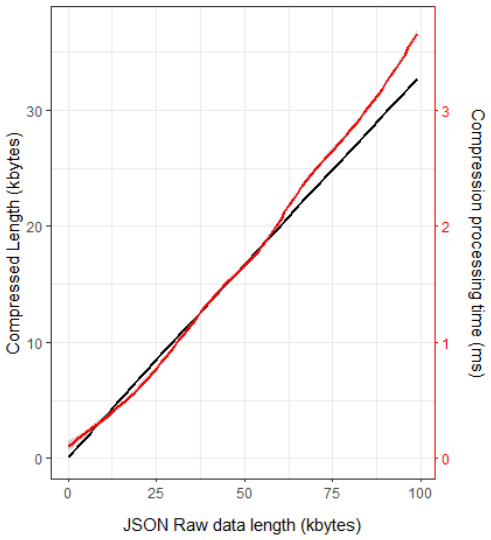
4.2 Kompresi Data API dengan Zlib

Contoh hasil pengukuran kompresi dan dekompresi data API yang diberikan oleh web server terkait data jaringan jalur transportasi umum diperlihatkan pada Tabel 1. Tabel 1 memperlihatkan lamanya proses pembangkitan data oleh server (t_{json}), waktu yang diperlukan untuk melakukan proses kompresi (t_c) dan dekompresi (t_d) data, serta panjang data (l_{json}) dan panjang data yang terkompresi (l_c). Selain itu, Tabel 1 juga memperlihatkan penggunaan memori untuk membangkitkan data dalam format JSON (m_{json}), besarnya memori yang diperlukan untuk melakukan kompresi data (m_c) dan memori yang diperlukan untuk melakukan dekompresi data (m_d).

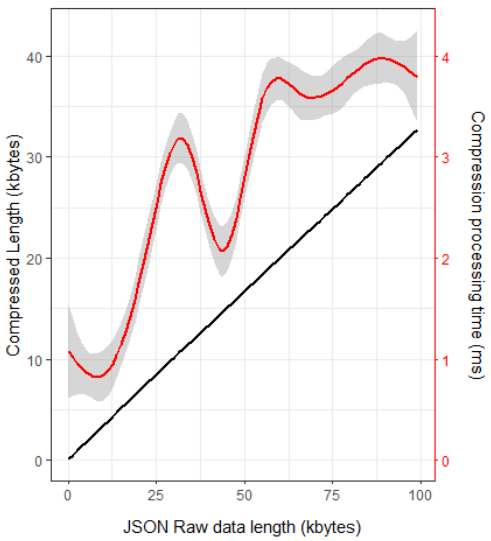
Tabel 1. Sampel data pengukuran performa kompresi dan dekompresi

t_{json} (ms)	t_c (ms)	t_d (ms)	l_{json} (bytes)	l_c (bytes)	m_{json} (kbytes)	m_c (kbytes)	m_d (kbytes)
2,58	0,13	0,40	135	144	417,47	0,18	133,30
3,50	0,10	0,20	1003	424	425,57	0,50	86,55
3,04	0,17	1,40	2066	768	435,48	0,87	95,52
2,16	0,14	0,29	3065	1108	445,44	1,25	107,23
2,50	0,16	0,40	4002	1432	452,76	1,50	107,06
3,63	0,18	2,09	5004	1772	465,85	2,00	141,77
2,06	0,18	0,50	6005	2108	473,69	2,50	122,08
2,94	0,22	1,69	7008	2448	481,54	2,50	126,97
2,66	0,23	2,09	8009	2788	489,39	3,00	151,69
2,36	0,32	2,80	9014	3128	506,74	4,00	136,85
3,24	0,31	0,70	10014	3460	sss514,59	4,00	161,72

Data yang diperoleh dari hasil penggunaan teknik kompresi data dapat diketahui bahwa ukuran data terkait jaringan transportasi umum yang telah dikompresi dengan Zlib dan dinormalisasi dengan *encoding* Base64 mampu menghemat *bandwidth* transmisi data di Internet rata-rata sebesar 64.61%. Adapun waktu proses tambahan yang diperlukan sistem untuk melakukan kompresi data rata-rata sebesar 0,197 ms atau sebesar 7,07% dari waktu yang dibutuhkan untuk membangkitkan data dalam format JSON itu sendiri (2,792 ms). Sedangkan waktu yang dibutuhkan untuk melakukan dekompresi data secara rata-rata selama 1,145 ms atau sebesar 41,01%, sebagaimana yang diperlihatkan dalam Gambar 6 dan diperlihatkan dalam Gambar 7. Walaupun terdapat penambahan waktu pemrosesan yang cukup besar, namun besarnya waktu yang dibutuhkan tersebut cukup singkat untuk dapat dirasakan perbedaannya oleh pengguna.



Gambar 6. Panjang data dan waktu kompresi



Gambar 7. Panjang data dan waktu dekompresi

Dari sisi penggunaan memori, memori rata-rata yang diperlukan untuk membangkitkan data dalam format JSON adalah sebesar 464,415 ms. Memori yang diperlukan untuk melakukan kompresi dan dekompresi data secara berturut-turut sebesar 2,03 kbytes dan 123,62 kbytes atau terdapat penambahan penggunaan memori sebesar 0,437% di sisi server. Penambahan penggunaan memori tersebut dinilai cukup kecil untuk jumlah pengguna sistem yang sedikit dalam waktu bersamaan.

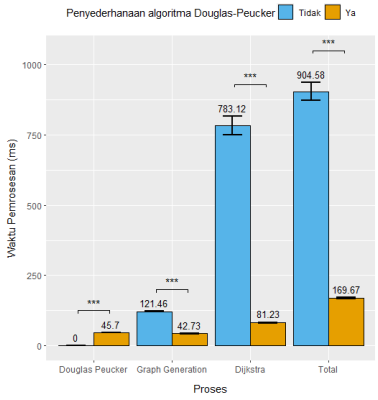
4.3 Penentuan Rekomendasi Rute dengan Algoritme Dijkstra dan Penyederhanaan Jalur Menggunakan Algoritme Douglas-Peucker

Data jaringan transportasi yang diperoleh dari *server* dalam format JSON kemudian diformat ke dalam struktur data graf untuk merepresentasikan struktur jaringan transportasi umum yang dapat diproses oleh algoritme Dijkstra. Algoritme Dijkstra termasuk ke dalam kategori algoritme *greedy*, di mana algoritme ini memeriksa seluruh kemungkinan yang ada untuk memberikan solusi jalur terpendek yang terbaik. Oleh karena itu, jika jumlah titik data dan jalur yang digunakan untuk merepresentasikan jaringan rute transportasi umum semakin banyak, maka waktu yang dibutuhkan oleh algoritme Dijkstra akan meningkat secara drastis.

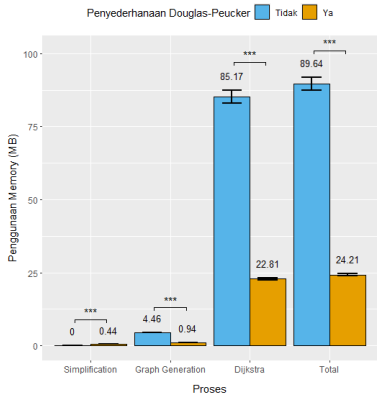
Penelitian ini mencoba untuk mengurangi jumlah titik data yang merepresentasikan rute perjalanan dan struktur jaringan transportasi umum secara visual dengan menggunakan algoritme *path simplification* Douglas-Peucker. Algoritme ini bekerja untuk menyederhanakan bentuk suatu jalur angkutan umum sedemikian rupa sehingga jumlah titik yang diperlukan untuk membentuk jalur tersebut dapat dikurangi dengan tetap mempertahankan bentuk yang disederhanakan serupa dengan bentuk aslinya. Hal ini diperlukan untuk mempertahankan representasi visual terkait jalur transportasi pada aplikasi yang digunakannya. Sehingga pengguna tetap dapat mengetahui tempat-tempat yang akan dilalui oleh jalur transportasi umum.

Tabel 2. Waktu pemrosesan komputasi *shortest-path*

	min	max	mean	median	sd	se
Tanpa penyederhanaan Douglas-Peucker						
t_s	0	0	0	0	0	0
t_g	96	159	121	120	11,3	0,798
t_d	99	1538	783	780	458	32,4
t_t	200	1660	905	914	460	32,5
Dengan penyederhanaan Douglas-Peucker						
t_s	39	77	45,7	43	6,5	0,46
t_g	37	70	42,7	41	6,44	0,455
t_d	38	159	81,2	82,5	27,3	1,93
t_t	121	237	170	171	27,7	1,96



Gambar 8. Waktu pemrosesan proses rekomendasi dengan algoritma Dijkstra



Gambar 9. Penggunaan memori proses rekomendasi algoritma Dijkstra

Tabel 2 memperlihatkan hasil pengukuran waktu proses simplifikasi jalur (t_s), waktu proses pembangkitan graf (t_g), waktu pemrosesan algoritme Dijkstra (t_d), dan waktu proses secara keseluruhan (t_t). Dari hasil pengukuran performa di laboratorium yang diperlihatkan pada Tabel 2, dapat diketahui bahwa proses penyederhanaan rute jalur transportasi umum memerlukan waktu rata-rata sebesar 45,7 ms. Dengan adanya penyederhanaan rute, waktu yang dibutuhkan untuk membangkitkan graf jaringan transportasi umum berkurang sebesar 64,82% dan waktu yang diperlukan oleh algoritme Dijkstra untuk menghitung rute perjalanan terpendek berkurang secara signifikan sebesar 89,63%. Nilai persentase ini didapatkan dari selisih perhitungan rata-rata waktu komputasi proses yang menggunakan teknik penyederhanaan dengan yang tidak. Walaupun teknik penyederhanaan rute ini berkontribusi dalam memberikan beban pemrosesan tambahan, secara keseluruhan waktu yang diperlukan untuk merekomendasikan rute transportasi secara umum berkurang sebesar 81,24%. Analisis secara statistik dengan uji Mann-Whitney U membuktikan secara statistik bahwa pengurangan jumlah titik dalam suatu graf terhubung mampu meningkatkan performa berjalannya algoritme yang menggunakan graf tersebut secara signifikan. Perbandingan waktu yang dibutuhkan untuk memberikan rekomendasi rute transportasi umum antara kedua kelompok pemrosesan diperlihatkan dalam Gambar 8.

Tabel 3. Penggunaan memori proses komputasi *shortest-path*

	min	**max**	**mean**	**median**	**sd**	**se**
Tanpa penyederhanaan Douglas-Peucker						
m_s	0	0	0	0	0	0
m_g	4,303	4,541	4,461	4,491	0,077	0,005
m_d	39,802	202	85,203	81,702	30,501	2,161
m_t	44,304	207	89,604	86,304	30,501	2,161
Dengan penyederhanaan Douglas-Peucker						
m_s	0,422	0,453	0,447	0,453	0,009	0,0006
m_g	0,922	1,152	0,948	0,938	0,043	0,003
m_d	16	45,303	22,8	21,801	5,291	0,374
m_t	17,403	46,704	24,201	23,202	5,291	0,374

Tabel 3 juga memperlihatkan hasil pengukuran penggunaan memori untuk proses simplifikasi (m_s), penggunaan memori untuk pembangkitan graf (m_g), penggunaan memori untuk proses penentuan rute terpendek dengan algoritme Dijkstra (m_d), dan waktu pemrosesan secara keseluruhan (m_t). Berdasarkan pengukuran penggunaan memori perangkat bergerak yang diperlihatkan pada Tabel 3, dapat disimpulkan bahwa penyederhanaan graf mampu mengurangi penggunaan sumber daya memori yang menjalankan proses pencarian rute terpendek dengan algoritme Dijkstra. Dari hasil pengukuran diketahui bahwa rata-rata penggunaan memori untuk menyederhanakan rute transportasi umum adalah sebesar 440 kbytes. Besarnya memori yang diperlukan untuk proses penyederhanaan rute tersebut adalah minimal jika dibandingkan dengan kapasitas memori (RAM) yang umumnya tersedia pada perangkat bergerak dengan sistem operasi Android modern yang pada umumnya dibekali RAM dengan ukuran kapasitas lebih dari 4GB. Dengan menyederhanakan graf menggunakan algoritme Douglas-Peucker, jumlah memori yang diperlukan untuk membangkitkan graf rata-rata berkurang sebesar 78,92%. Memori yang digunakan untuk menjalankan algoritme Dijkstra berkurang hingga 73,22%. Secara keseluruhan, penggunaan memori untuk menghasilkan rekomendasi rute terpendek berkurang sebesar 72,99%. Analisis secara statistik dengan menggunakan teknik uji beda Mann-Whitney U Test memperlihatkan bahwa pemrosesan rute terpendek dengan penyederhanaan jalur dan tanpa penyederhanaan jalur menggunakan algoritme Douglas-Peucker secara statistik berbeda dari analisis dengan nilai *p-value* lebih kecil dari nilai signifikansi 0,05. Perbandingan penggunaan memori antara kedua kelompok analisis diperlihatkan dalam Gambar 9.

5. BATASAN

Penelitian ini dibatasi pada konteks penelitian rute transportasi umum yang ada di Kota Malang. Oleh karena data yang digunakan dalam penelitian ini hanya terdiri dari 12 rute transportasi umum dari total rute transportasi umum keseluruhan sebanyak 30 rute perjalanan, maka penggunaan sistem pada kondisi

aktual di lapangan akan memerlukan kapasitas memori yang lebih tinggi dan waktu yang diperlukan oleh sistem untuk melakukan proses komputasi rekomendasi dinilai akan memakan waktu yang lebih lama. Selain itu, penggunaan server dan perangkat bergerak dengan karakteristik sistem operasi dan spesifikasi yang berbeda dari apa yang digunakan dalam penelitian ini akan memberikan pengalaman pengguna yang berbeda pula dari apa yang terukur dalam penelitian ini.

6. KESIMPULAN

Penggunaan dokumen dalam format JSON efektif digunakan sebagai media komunikasi client-server aplikasi perangkat bergerak berbasis HTTP. Berdasarkan hasil pengukuran dan analisis, penggunaan kompresi data Zlib yang telah dinormalisasikan dengan format *encoding* Base64 mampu menghemat penggunaan *bandwidth* untuk mendistribusikan data di Internet sebesar 64.61%. Beban tambahan terhadap penggunaan memori yang diperlukan oleh server untuk melakukan kompresi data sebesar 0.44% dari memori yang digunakan untuk membangkitkan data itu sendiri. Dengan penyederhanaan graf jaringan transportasi dengan algoritme Douglas-Peucker, waktu yang dibutuhkan proses komputasi untuk merekomendasikan rute transportasi secara rata-rata lebih singkat 81.24% dan penggunaan memori untuk melakukan proses komputasi tersebut berkurang sangat signifikan hingga sebesar 72,99%. Dengan demikian, hasil penelitian ini memperlihatkan bahwa penggunaan teknik kompresi data dan penyederhanaan struktur data graf melalui pendekatan visual mampu meningkatkan performa komputasi algoritma Dijkstra dengan baik.

APRESIASI

Penelitian ini didanai dari Program Hibah Penelitian Doktor Non-Lektor Kepala Fakultas Ilmu Komputer Universitas Brawijaya Tahun Anggaran 2023 No. 2122/UN10.F15/PN/2023.

DAFTAR PUSTAKA

COOKSEY, BRIAN. 2014. Chapter 3: Data formats. Zapier. https://zapier.com/learn/apis/chapter-3-data-formats/

DOUGLAS, DAVID; PEUCKER, THOMAS 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer. 10 (2): 112–122. doi:10.3138/FM57-6770-U75U-7727.

FERRAGINA, P.I., NITTO, I. 2009. Text Compression. In: LIU, L., ÖZSU, M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_1151

FOEAD, D., GHIFARI, A., KUSUMA, M.B., HANAFIAH, N., & GUNAWAN, E., 2021. A Systematic Literature Review of A* Pathfinding. Procedia Computer Science, Vol. 179, pp. 507-514. https://doi.org/10.1016/j.procs.2021.01.034.

FRIASWANTO, MELKI & LISANGAN, ERICK & SUMARTA, SEAN. 2021. The Simulation of Traffic Signal Preemption using GPS and Dijkstra Algorithm for Emergency Fire Handling at Makassar City Fire Service. International Journal of Applied Sciences and Smart Technologies. 3. 185-202. doi:10.24071/ijasst.v3i2.3821.

GUEDRI, HICHEM., BAJAHZAR, ABDULLAH., BELMABROUK, HAFEDH. 2021. ECG compression with Douglas- Peucker algorithm and fractal interpolation, [J]. Mathematical Biosciences and Engineering, 18(4): 3502-3520. doi: 10.3934/mbe.2021176

JILIN, CHEN., MIN, ZHAO., ZHONGHUA, GUO., WEIJIANG, QIU., YONG, CHEN., AND WEIXI, WANG. 2018. The Application of Douglas-Peucker Algorithm in Collaborative System for Power Grid Operation Mode Calculation. MATEC Web of Conferences 175.

JONES, THOMAS. 2021. What is an API Integration? (A guide for non-technical people). https://www.gend.co/blog/what-is-api-integration-a-guide-for-non-technical-people

KANJILAL, JODIP. 2017. Compressing Web API responses to reduce payload. InfoWorld. https://www.infoworld.com/article/3174597/compressing-web-api-responses-to-reduce-payload.html

KHARISMA, A. P., & PINANDITO, A. 2017. Design of REST API for Local Public Transportation Information Services in Malang City. Journal of Information Technology and Computer Science, 2(2). https://doi.org/10.25126/jitecs.20172226

KHARISMA, A. P., JONEMARO, E. M. A., & ARWANI, I. 2019. Paratransit Trip Data Collection System with Smartphone GPS and REST Web Service in Malang, Indonesia. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 9(1). http://dx.doi.org/10.52549/ijeei.v9i1.2507

KHOTARI, VIVEK. 2023. Difference between JSON and XML. Geeks for Geeks. https://www.geeksforgeeks.org/difference-between-json-and-xml/

LAKHERA, SAKSHAM, AND PRAVEENA T. 2020. Visual Analysis using Modified Ramer-Douglas–Peucker Algorithm on Time Series Data. International Research Journal of Engineering and Technology (IRJET), Vol. 07 (05), pp. 4225-4229

LIU, BO, XUECHAO LIU, DAJUN LI, YU SHI, GABRIELA FERNANDEZ, AND YANDONG WANG. 2020. A Vector Line Simplification Algorithm Based on the Douglas–Peucker Algorithm, Monotonic Chains and Dichotomy. ISPRS International Journal of Geo-Information, 9(4): 251. https://doi.org/10.3390/ijgi9040251

MDN. 2023. Introduction to web APIs. Mozilla Developer Network. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

NAYAGAM, R. DEIVA., SELVATHI, D., GEETA, R., GOPINATH, D., AND SIVAKUMAR, G. 2023. Mobile Application based Indoor Positioning and Navigational System using Dijkstra's Algorithm. Journal of Physics: Conference Series, Volume 2466, 4th National Conference on Communication Systems (NCOCS 2022), DOI 10.1088/1742-6596/2466/1/012007

OPHEIM, H. 1981. Smoothing a digitized curve by data reduction methods. Eurographics' 81. Pp.127- 135.

PARULEKAR, M., PADTE, V., SHAH, T., SHROFF, K., AND SHETTY, R., 2013. "Automatic vehicle navigation using Dijkstra's Algorithm," 2013 International Conference on Advances in Technology and Engineering (ICATE), Mumbai, India, pp. 1-5, doi: 10.1109/ICAdTE.2013.6524721.

PINANDITO, ARYO. 2017. Framework Design for Modular Web-based Application Using Model-CollectionService-Controller-Presenter (MCCP) Pattern. Journal of Information Technology and Computer Science 2(1). DOI: 10.25126/jitecs.20172120

PINANDITO, ARYO AND WULANDARI, CHANDRAWATI PUTRI. 2021. Integrating douglas-peucker line simplification into routeboxer algorithm on a map-based Android application. In Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology (SIET '20). Association for Computing Machinery, New York, NY, USA, 213–219. https://doi.org/10.1145/3427423.3427446

RACHMAWATI, D. & GUSTIN, L. 2020. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem. J. Phys.: Conf. Ser. 1566 012061. doi:10.1088/1742-6596/1566/1/012061

REIN, LISA. 1998. Handling Binary Data in XML Documents. XML.com. https://www.xml.com/pub/a/98/07/binary/binary.html

REUMANN, K., AND WITKAM, A.P.M. 1974. Optimizing Curve Segmentation in Computer Graphics. International Computing Symposium. Amsterdam, North Holland, pp. 467-472.

ROSNER, FRANK. 2017. Hiding Complexity Does Not Make It Go Away, Or Does It?. Dev.to. https://dev.to/frosnerd/hiding-complexity-does-not-make-it-go-away-or-does-it-51i

RUAN, CHUANG., LUO, JIANPING., AND WU, YU 2014. Map navigation system based on optimal Dijkstra algorithm," 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, Shenzhen, pp. 559-564, doi: 10.1109/CCIS.2014.7175798.

SALEM, I. E., MIJWIL, M. M., ABDULQADER, A. W., ISMAEEL, M. M. 2022. Flight-schedule using Dijkstra's algorithm with comparison of routes findings. Int J Elec & Comp Eng, Vol. 12(2), pp. 1675-1682. http://doi.org/10.11591/ijece.v12i2.pp1675-1682

SALOMON D., 2007. Data Compression: The Complete Reference, 4th edn., Springer, London.

SAMAH, ABU., FARIZA, KHYRINA AIRIN., & SHARIP, A. & MUSIRIN, PROFESSOR DR. ISMAIL & SABRI, N. & SALLEH, M. 2020. Reliability study on the adaptation of Dijkstra's algorithm for gateway KLIA2 indoor navigation. Bulletin of Electrical Engineering and Informatics. 9. 10.11591/eei.v9i2.2081.

SARNACKI, PIOTR. 2013. Client and API isolation. Travis CI. https://blog.travis-ci.com/2013-03-13- client-and-api-isolation

SAYOOD, KHALID. 2018. Huffman Coding. Introduction to Data Compression (Fifth ed.).

SHI, WENZHONG & CHEUNG, CHUIKWAN. 2006. Performance Evaluation of Line Simplification Algorithms for Vector Generalization. Cartographic Journal, The. 43. pp. 27-44. doi:10.1179/000870406X93490.

SHIOTSU, YOSHITAKA. 2021. What Is PHP and Why Should You Use It?. Upwork Resource Center. https://www.upwork.com/resources/why-use-php

SINGH, Y., SHARMA, S., SUTTON, R., HATTON, D. 2018. Towards use of Dijkstra Algorithm for Optimal Navigation of an Unmanned Surface Vehicle in a Real-Time Marine Environment with results from Artificial Potential Field. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 12, No. 1, doi:10.12716/1001.12.01.14, pp. 125-131,

SUGIHARJA, DANUR DAN PAHLEVI, OMAR. 2019. The RPTRA Geographic Information System Application in Central Jakarta City

Using the Dijkstra Algorithm Based on Android. Journal Publications & Informatics Engineering Research. Vol. 3, No. 2. pp. 56-60. https://doi.org/10.33395/sinkron.v3i2.10043

SUJATHA RAJKUMAR, SHREYAS LOYA, DHRUV HARIBHAKTI, SIVAKUMAR SUBRAMANIAM. 2020. Pollution based Intelligent Navigation System using Dijkstra's Algorithm. International Journal of Advanced Science and Technology, 29(3), 11242 - 11250. Retrieved from: http://sersc.org/journals/index.php/IJAST/article/view/28023

TUPITSIN, ANDREY; PUZRIN, VITALY. 2022. Pako. http://nodeca.github.io/pako/

TSCHABITSCHER, HEINZ. 2020. How Base64 Encoding Works. Lifewire. https://www.lifewire.com/base64-encoding-overview-1166412

Visvalingam, M., & Whyatt, J. D. (992. Line generalisation by repeated elimination of the smallest area. Cartographic Information Systems Research Group, University of Hull

VISVALINGAM, M. & WHYATT, DUNCAN. 2007. The Douglas-Peucker Algorithm for Line Simplification: Re-evaluation through Visualization. Computer Graphics Forum. 9. 213 - 225. 10.1111/j.1467-8659.1990.tb00398.x.

WANG, XIAOFEI., ZHANG, JIE., YOU, LEI., 2021. A Douglas-Peucker Algorithm Combining Node Importance and Radial Distance Constraints. AIAM2021: 2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture, pp. 265–269. https://doi.org/10.1145/3495018.3495063

XU, YICHENG., WEN, ZHIGANG., ZHANG, XIAOYING. 2015. Indoor optimal path planning based on Dijkstra Algorithm. Proceedings of the 2015 International Conference on Materials Engineering and Information Technology Applications. pp. 309-313. https://doi.org/10.2991/meita-15.2015.57

ZHAO, Z., & SAALFELD, A. 2008. Linear-time sleeve-fitting polyline simplification algorithms. Cartogis. pp. 214-223.

ZHOU, M, & GAO, N. 2019. Research on Optimal Path based on Dijkstra Algorithms. Proceedings of the 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019). https://doi.org/10.2991/icmeit-19.2019.141

ZIV, JACOB; LEMPEL, ABRAHAM. 1977. A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory. 23 (3): 337–343. CiteSeerX 10.1.1.118.8921. doi:10.1109/TIT.1977.1055714.