INOTEK, Vol. 9
ISSN: 2580-3336 (Print) / 2549-7952 (Online)
Url: https://proceeding.unpkediri.ac.id/index.php/inotek/

Perbandingan Algoritma Djikstra dan Floyd-Warshall Untuk Membuat Data Input TSP

[1]M.Farkhan Fahmi Zuhri, [2]Daniel Swanjaya, [3]Julian Sahertian
[1,2,3]Teknik Informatika, Universitas Nusantara PGRI Kediri
E-mail: [1]*farkhanfahmi79@gmail.com*, [2]daniel@unpkediri.ac.id, [3]juliansahertian@unpkediri.ac.id
Penulis Korespondens : M.Farkhan Fahmi Zuhri

Abstrak—Optimasi rute pada pengiriman air galon sangat penting untuk efisiensi waktu dan biaya di daerah pedesaan seperti Desa Sumberejo. Penelitian ini bertujuan membandingkan kinerja waktu komputasi algoritma Dijkstra yang dijalankan secara iteratif dan Floyd-Warshall dalam menghasilkan matriks waktu tempuh terpendek dan rute antar semua pasangan pelanggan dan depot. Matriks ini ditujukan untuk input algoritma optimasi *multi-stop* seperti *Traveling Salesman Problem* (TSP). Studi kasus yang digunakan dalam penelitian ini adalah jaringan jalan riil Desa Sumberejo yang memiliki 62 node jalan dan 125 node target yaitu pelanggan dan depot dengan waktu tempuh sebagai bobot graf yang dihitung berdasarkan kondisi jalan aktual. Hasil pengujian waktu eksekusi menunjukkan bahwa secara signifikan pendekatan Dijkstra iteratif lebih cepat dibandingkan Floyd-Warshall untuk semua skenario jumlah pelanggan yang diuji. Hasil ini menunjukan bahwa Dijkstra lebih efisien untuk pra-pemrosesan data TSP pada skala jaringan serupa.
Kata Kunci— Algoritma Dijkstra, Algoritma Floyd-Warshall, Matriks Jarak, Optimasi Rute, Pengiriman Galon, TSP, Waktu Komputasi

Abstract— *Route optimization for bottled water delivery is crucial for time and operational cost efficiency, especially in rural areas such as Sumberejo Village. This study aims to compare the computational performance of the iterative Dijkstra algorithm and the Floyd-Warshall algorithm in generating the shortest travel time matrix and routes between all pairs of customers and depots. This matrix serves as a vital input for multi-stop optimization algorithms such as the Traveling Salesman Problem (TSP). The method employed is a case study on the real road network of Sumberejo Village, which consists of 62 road nodes and 125 target customer/depot nodes, with graph weights representing travel times based on actual road conditions. The experimental results show that the iterative Dijkstra approach performs significantly faster in execution time compared to Floyd-Warshall across all tested customer scenarios. These findings recommend the iterative Dijkstra algorithm as a more efficient method for TSP data preprocessing in similarly scaled networks.*
Keywords—*Dijkstra's Algorithm, Floyd-Warshall Algorithm, Distance Matrix, Optimization Route, Gallon Delivery, TSP, Computation Time*

I. PENDAHULUAN

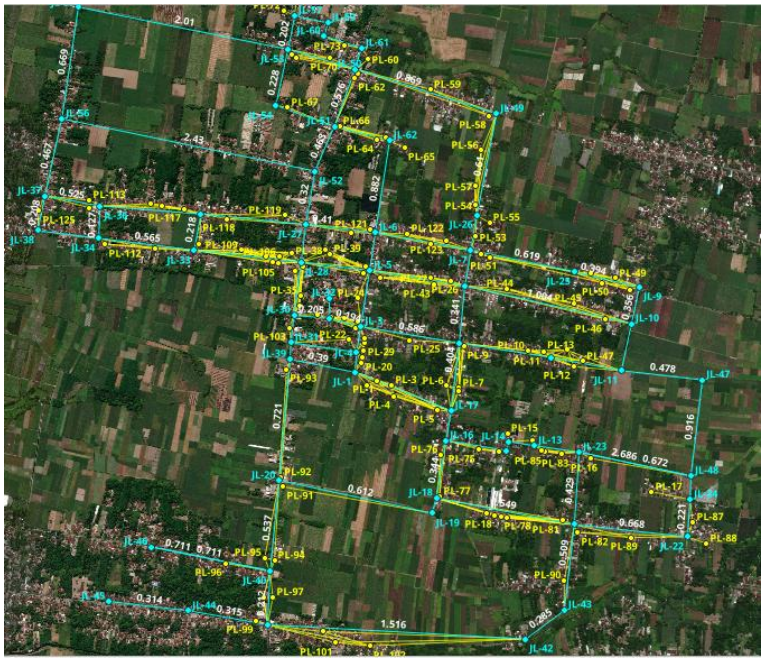
Keberhasilan operasional sangat bergantung pada sistem distribusi barang yang efisien, terutama untuk barang kebutuhan sehari-hari seperti air minum dalam galon. Optimalisasi rute pengantar menurunkan biaya bahan bakar dan waktu perjalanan serta meningkatkan layanan dan kepuasan pelanggan. Saat melayani banyak titik pelanggan dari satu depot, masalah yang sering terjadi adalah menentukan urutan kunjungan dan rute terpendek yang harus diambil. Masalah

klasik dalam manajemen logistik dikenal sebagai masalah penjualan perjalanan (TSP) atau masalah rute kendaraan (VRP).

Untuk mengatasi masalah dengan optimasi rute, banyak algoritma pencarian jalur terpendek telah dibuat dan digunakan. Algoritma Djikstra dan Floyd-Warshall adalah dua algoritma yang paling terkenal dan sering dibandingkan. Algoritma Dijkstra umumnya digunakan untuk menemukan jalur terpendek dari satu titik asal ke semua titik lain dalam graf berbobot non-negatif, sedangkan Algoritma Floyd-Warshall dirancang untuk menentukan jalur terpendek antar semua pasangan simpul dengan mengevaluasi semua lintasan yang memungkinkan. Memiliki matriks jarak atau waktu perjalanan yang tepat antara semua pasangan titik pemberhentian adalah langkah awal yang penting dalam konteks TSP atau VRP sebelum menentukan urutan kunjungan yang terbaik.

Sejumlah penelitian terdahulu telah membandingkan dan menerapkan kedua algoritma ini. Hendra dan Riti menemukan Dijkstra lebih efisien dari segi perhitungan dan kompleksitas program untuk rute wisata Surabaya[1]. Sementara Muhlasin, Noviandi, dkk. menyoroti kegunaan Floyd-Warshall untuk rute multi-stop kurir karena kemampuannya menghitung seluruh jarak rute sebagai input TSP[2], didukung oleh Khalisya dan Vikaliana yang mengoptimasi distribusi multi-titik J&T Express menggunakan algoritma yang sama[3]. Darmadi dkk. juga menunjukkan efektivitas Floyd-Warshall untuk pemetaan rumah sakit[4]. Di sisi lain, Marlina dkk. menemukan hasil rute yang sebagian besar identik antara Dijkstra dan Floyd-Warshall untuk rute wisata Batang, dengan perbedaan pada beberapa kasus akibat prinsip kerja yang berbeda[5]. Syahputri dkk. menunjukkan penghematan jarak dan biaya 6,7% melalui Floyd-Warshall dalam optimasi rute pengangkutan sampah[6], dan Risald mengusulkan kolaborasi keduanya untuk rute ambulans guna memanfaatkan keunggulan masing-masing[7].

Studi yang secara khusus memeriksa dan membandingkan efisiensi waktu komputasi antara pendekatan Dijkstra dan Floyd-Warshall yang dijalankan secara iteratif (untuk setiap node target) dengan algoritma Floyd-Warshall dalam konteks menghasilkan matriks jarak lengkap antar semua pasangan pelanggan dan depot masih diperlukan. Harapan penelitian ini adalah untuk menganalisis dan membandingkan kinerja algoritma Djikstra yang digunakan secara iteratif dari setiap node. Selain itu, penggunaan pada jaringan jalan di daerah pedesaan seperti Desa Sumberejo, dengan penggunaan bobot graf berupa waktu tempuh riil yang memperhitungkan variasi kondisi jalan aktual, adalah aspek penting yang belum banyak dipelajari secara komparatif untuk kedua algoritma ini sebagai langkah pra-pemrosesan data TSP.



Gambar 1. Jaringan Jalan Desa Sumberejo

Waktu komputasi total yang dibutuhkan oleh masing-masing metode adalah parameter utama untuk perbandingan. Berdasarkan hasil pengujian awal yang dilakukan pada dataset penelitian ini, algoritma Djikstra, yang digunakan secara iteratif untuk setiap pelanggan sebagai titik awal, diperkirakan akan menghabiskan waktu komputasi yang lebih sedikit secara keseluruhan dibandingkan dengan algoritma Floyd-Warshall. Untuk menghasilkan matriks jarak semua-pasangan-pelanggan pada graf jaringan jalan Desa Sumberejo, yang terdiri dari 187 node total dan 125 node target, algoritma Djikstra diperkirakan akan Diharapkan kontribusi penelitian ini akan memberikan pedoman berbasis bukti empiris untuk memilih algoritma yang lebih efisien untuk tugas pra-pemrosesan data yang berkaitan dengan optimasi rute pengiriman multi-titik pada skala dan fitur jaringan yang sebanding.

II. METODE

Penelitian ini menggunakan pendekatan kuantitatif dengan desain studi komparatif eksperimental untuk menganalisis kinerja algoritma Dijkstra dan Floyd-Warshall dalam penentuan rute terpendek pada jaringan jalan pengiriman galon air di Desa Sumberejo. Bobot utama yang digunakan dalam pencarian jalur terpendek adalah estimasi waktu tempuh riil antar titik yang dihitung dari kondisi jalan yang aktual.

A. Sumber Data dan Pengumpulan Data

Data penelitian ini mencakup data spasial (node dan edge jaringan jalan) dan data atribut (bobot edge). Data pelanggan dan depot disimpan dalam file CSV terpisah per skenario pengujian, di mana node jalan merepresentasikan persimpangan penting, dan edge adalah ruas jalan yang menghubungkan semua jenis node. Data yang digunakan dalam penelitian ini disajikan dalam Tabel 1.

.Tabel 1. Sumber Data

Jenis Data	Nama File	Deskripsi Singkat
Skenario 1	10Node.csv	Terdiri atas kolom *Id_Node* dan Nama dengan data 9 node pelanggan dan 1 *node* depot
Skenario 2	50Node.csv	Terdiri atas kolom Id_Node dan Nama dengan data 49 node pelanggan dan 1 node depot
Skenario 3	FullNode.csv	Terdiri atas kolom Id_Node dan Nama dengan data 124 node pelanggan dan 1 node depot
Data Set	NodeJalan.csv	Terdiri atas kolom Id_Node dengan data 62 node jalan
Data Set	Edge.csv	Terdiri atas kolom Node_Asal, Node_Tujuan, Waktu, Jarak dengan data 455 edge

Pengumpulan data spasial node dan edge, serta perhitungan awal jarak fisik dan waktu tempuh, dilakukan menggunakan perangkat lunak QGIS.

B. Representasi Graf

Jaringan jalan Desa Sumberejo direpresentasikan sebagai graf tidak berarah yang dinyatakan dengan notasi :

$$G = (V,E) \quad (1)$$

V adalah himpunan semua node unik dalam jaringan, termasuk node depot, pelanggan, dan node jalan. Total node dalam graf ini berjumlah 187.

E adalah himpunan dari 455 edge yang menghubungkan pasangan node dalam *V*.

Setiap edge $e \in E$ memiliki bobot *wt(e)* yang mewakili waktu tempuh, dan *wd(e)* yang mengindikasikan jarak fisik antar pasangan node yang terhubung oleh edge tersebut.

Dalam implementasi menggunakan Python, graf ini direpresentasikan melalui struktur data adjacency list yang disimpan dalam bentuk dictionary of dictionaries. Dengan demikian, graf *G* dapat diakses dengan menggunakan format *graf[u][v]*, di mana *u* dan *v* adalah dua node yang terhubung oleh sebuah edge, dan nilai yang disimpan pada *graf[u]* adalah dictionary yang berisi dua entri: "waktu" (untuk bobot waktu) dan "jarak" (untuk bobot jarak).

C. Implementasi Algoritma

Implementasi kedua algoritma pencarian rute terpendek, yaitu Floyd-Warshall dan Dijkstra, dilakukan menggunakan bahasa pemrograman Python. Proses implementasi mencakup inisialisasi struktur data, eksekusi logika inti algoritma, dan mekanisme untuk merekonstruksi rute yang dihasilkan.

1. Algoritma *Floyd-Warshall*

Prinsip kerja algoritma Floyd-Warshall adalah membandingkan semua kemungkinan jalur dalam graf secara iteratif untuk setiap pasangan simpul, hingga diperoleh nilai optimal, dan inputnya dapat berupa graf berarah maupun tidak berarah dengan bobot sisi yang bisa positif atau negatif[8]. Fokus utama algoritma ini adalah menghasilkan dua matriks utama, yaitu matriks waktu tempuh terpendek W_{ij} dan matriks total jarak fisik kumulatif D_{ij}. Proses pembaruan jarak dilakukan berdasarkan prinsip relasi berikut :

$$w_{ij} = \min(w_{ij,}\ w_{ik} + w_{kj}) \quad (2)$$

Untuk memungkinkan penelusuran kembali rute yang terbentuk, sebuah matriks tambahan $next_{ij}$ juga dibangun selama proses perhitungan. Matriks ini berfungsi untuk mencatat simpul perantara berikutnya yang harus dilalui pada jalur terpendek dari simpul *i* menuju simpul *j*.

2. Algoritma Djikstra

Algoritma Dijkstra bekerja dengan prinsip *greedy*, di mana pada setiap langkahnya akan dipilih sisi dengan bobot minimum yang menghubungkan simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Meskipun efisien untuk pencarian dari satu titik sumber, pendekatan iteratif diperlukan untuk kasus semua-pasangan-simpul-target[9]. Implementasi ini memanfaatkan struktur data *min-priority queue*, yang diwujudkan menggunakan modul *heapq* pada *python*, untuk efisiensi dalam memilih simpul dengan jarak terpendek yang belum diproses. Setiap eksekusi Dijkstra dari satu simpul sumber menghasilkan *dictionary* waktu tempuh minimum, total jarak fisik kumulatif pada rute terpendek (berdasarkan waktu), dan *dictionary predecessor* untuk rekonstruksi rute ke semua simpul lain yang dapat dijangkau.

Fungsi Dijkstra dijalankan secara interatif pada setiap simpul target dari himpunan node target (pelanggan dan depot) yang telah ditentukan. Hasil dari setiap simpul target ke semua simpul target lainnya dikumpulkan dan digabungkan untuk membentuk matriks jarak antar pasangan target.

3. Rekonstruksi Rute

Setelah algoritma pencarian jalur selesai dieksekusi dan menghasilkan informasi jarak serta rute, langkah selanjutnya adalah merekonstruksi urutan simpul yang membentuk jalur terpendek.

a. Untuk Floyd-Warshall, fungsi rekonstruksi menggunakan matriks $next_{ij}$ untuk menelusuri jalur dari simpul awal *i* ke simpul tujuan *j* hingga mencapai simpul tujuan.
b. Untuk Dijkstra, fungsi rekonstruksi menggunakan dictionary *Predecessor* untuk menelusuri jalur mundur dari simpul tujuan hingga mencapai simpul sumber, kemudian membalik urutan jalur tersebut untuk mendapatkan rute dari sumber ke tujuan.

D. Skenario Pengujian dan Parameter Kinerja

Pengujian dilakukan dengan tiga skenario jumlah node target yang berbeda yaitu 10 node, 50 node, dan 125 node. File data node jalan dan edge tetap sama untuk semua skenario. Parameter utama yang diukur untuk membandingkan kedua algoritma adalah waktu total yang dibutuhkan oleh masing-masing algoritma untuk menghasilkan matriks jarak antar pasangan *node* target. Waktu eksekusi diukur dengan menggunakan fungsi *time.perf_counter()* di *Python*.

III. HASIL DAN PEMBAHASAN

Bagian ini memaparkan hasil pengujian komparatif kinerja algoritma Dijkstra (dijalankan secara iteratif) dan Floyd-Warshall. Fokus utama adalah analisis waktu komputasi dan validasi kebenaran *output* dalam konteks pembentukan matriks jarak waktu tempuh sebagai *input* TSP pada jaringan jalan Desa Sumberejo.

A. Hasil Pengujian Kinerja Algoritma

Pengujian dilakukan pada representasi graf jaringan jalan Desa Sumberejo yang terdiri dari 187 *node* unik (1 depot, 124 pelanggan, dan 62 *node* jalan) serta 455 *edge*. Graf ini dikategorikan sebagai graf jarang (*sparse*) tidak berarah dengan nilai densitas 0.0262, yang dihitung menggunakan rumus berikut:

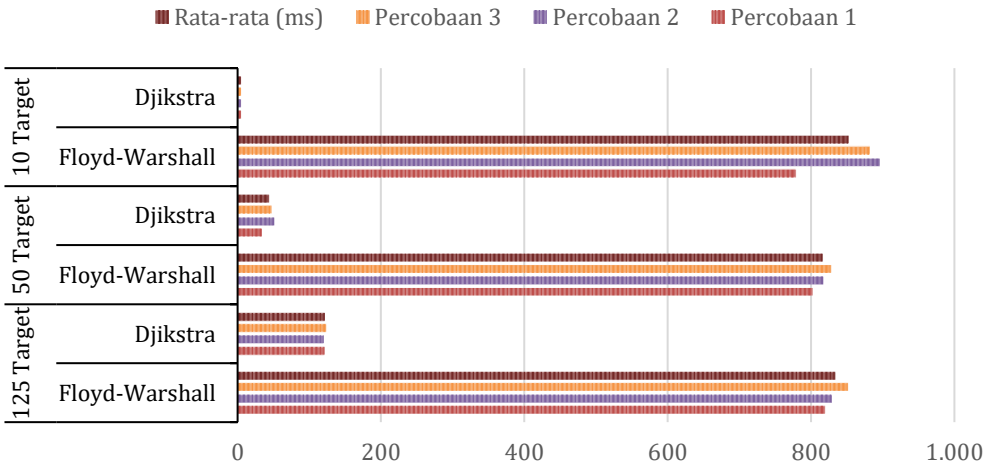
$$D = \frac{\mathrm{E}}{\mathrm{V} \times (\mathrm{V} - 1)} \tag{3}$$

Dengan E adalah jumlah *edge*, dan V adalah jumlah *node*. Analisis kuantitatif dilakukan berdasarkan tiga skenario jumlah *node* target, yaitu 10, 50, dan 125 *node target* dengan waktu sebagai parametern utama yang diukur. Hasil rata-rata waktu komputasi dari tiga kali percobaan untuk setiap algoritma dan skenario disajikan pada Table 2 sebagai berikut.

Tabel 2. Hasil Waktu Komputasi

Skenario	Algoritma	Percobaan 1	Percobaan 2	Percobaan 3	Rata-rata (ms)
10 Target	Floyd-Warshall	778.9363	895.7076	881.2255	851,9565
	Djikstra	4.9832	5.0623	5.0936	5,0464
50 Target	Floyd-Warshall	802.3241	816.9517	827.4165	815,5641
	Djikstra	33.6469	51.0901	47.5365	44,0912
125 Target	Floyd-Warshall	819.4718	829.0706	851.1520	833,2315
	Djikstra	121.2521	120.3386	123.6961	121,7623

Dari Tabel 2 dapat dilihat bahwa algoritma Dijkstra, yang dijalankan secara iteratif menunjukkan waktu komputasi yang jauh lebih cepat dibandingkan dengan algoritma Floyd-Warshall pada setiap skenario pengujian. Sebagai contoh, pada skenario 10 node target, Dijkstra iteratif lebih cepat sekitar 132 kali dibandingkan dengan Floyd-Warshall. Pada skenario dengan 50 node target, Dijkstra iteratif lebih cepat sekitar 21 kali, dan pada 125 node target, Dijkstra iteratif lebih cepat sekitar 6,3 kali.



Gambar 2. Grafik Hasil Waktu Komputasi

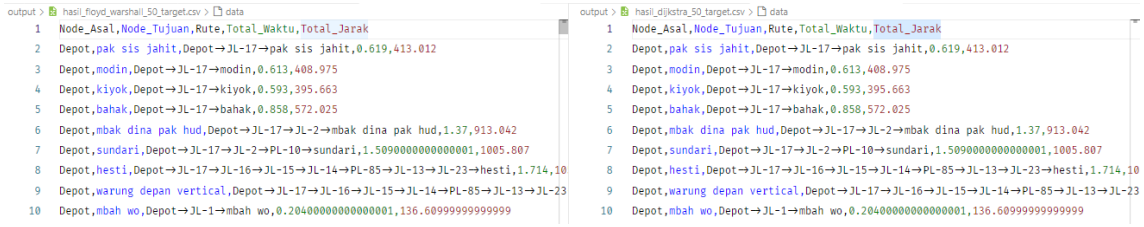
Gambar berikut menunjukkan perbandingan waktu komputasi antara algoritma Dijkstra dan Floyd-Warshall untuk tiga skenario jumlah node target yang berbeda.

1. Analisis Waktu Komputasi

Waktu eksekusi Floyd-Warshall cenderung konstan karena memproses seluruh graf secara simultan, sementara waktu Dijkstra meningkat dengan bertambahnya jumlah *node* target yang diiterasi, meski tetap jauh lebih efisien. Hal ini sesuai dengan kompleksitas waktu masing-masing: Floyd-Warshall memiliki kompleksitas $O(V^3)$ (V adalah total *node*), sehingga waktu eksekusinya konstan terhadap jumlah *node* target namun dipengaruhi ukuran graf keseluruhan. Dijkstra dengan *min-priority queue* berkompleksitas O(E log V) (E adalah jumlah *edge*, V jumlah *node*); ia lebih cepat karena hanya memproses *node* target secara iteratif, bukan seluruh graf, dan sangat efisien pada jaringan jarang (*sparse graph*).

B. Validasi Hasil Rute dan Jarak

Untuk memvalidasi hasil yang diperoleh, kami membandingkan output matriks jarak waktu tempuh terpendek dan total jarak fisik antar pasangan node target yang dihasilkan oleh kedua algoritma. Hasil validasi menunjukkan bahwa meskipun kedua algoritma memiliki perbedaan waktu komputasi, hasil akhir dari jarak waktu tempuh terpendek dan total jarak fisik adalah identik untuk semua pasangan node target yang diuji.



output > hasil_floyd_warshall_50_target.csv > data

```
Node_Asal,Node_Tujuan,Rute,Total_Waktu,Total_Jarak
Depot,pak sis jahit,Depot→JL-17→pak sis jahit,0.619,413.012
Depot,modin,Depot→JL-17→modin,0.613,408.975
Depot,kiyok,Depot→JL-17→kiyok,0.593,395.663
Depot,bahak,Depot→JL-17→bahak,0.858,572.025
Depot,mbak dina pak hud,Depot→JL-17→JL-2→mbak dina pak hud,1.37,913.042
Depot,sundari,Depot→JL-17→JL-2→PL-10→sundari,1.5090000000000001,1005.807
Depot,hesti,Depot→JL-17→JL-16→JL-15→JL-14→PL-85→JL-13→JL-23→hesti,1.714,10
Depot,warung depan vertical,Depot→JL-17→JL-16→JL-15→JL-14→PL-85→JL-13→JL-23
Depot,mbah wo,Depot→JL-1→mbah wo,0.20400000000000001,136.60999999999999
```

output > hasil_dijkstra_50_target.csv > data

```
Node_Asal,Node_Tujuan,Rute,Total_Waktu,Total_Jarak
Depot,pak sis jahit,Depot→JL-17→pak sis jahit,0.619,413.012
Depot,modin,Depot→JL-17→modin,0.613,408.975
Depot,kiyok,Depot→JL-17→kiyok,0.593,395.663
Depot,bahak,Depot→JL-17→bahak,0.858,572.025
Depot,mbak dina pak hud,Depot→JL-17→JL-2→mbak dina pak hud,1.37,913.042
Depot,sundari,Depot→JL-17→JL-2→PL-10→sundari,1.5090000000000001,1005.807
Depot,hesti,Depot→JL-17→JL-16→JL-15→JL-14→PL-85→JL-13→JL-23→hesti,1.714,10
Depot,warung depan vertical,Depot→JL-17→JL-16→JL-15→JL-14→PL-85→JL-13→JL-23
Depot,mbah wo,Depot→JL-1→mbah wo,0.20400000000000001,136.60999999999999
```

Gambar 3. Contoh Hasil Rute Yang Ditemuukan

Tabel 3. Total Rute Yang Ditemukan

File Output	Total Rute Yang Ditemukan	
	Floyd-Warshall	Djikstra
10 Node	73	73
50 Node	2353	2353
125 Node	15253	15253

Dari Gambar 3 dan Tabel 3, dapat dilihat bahwa jumlah rute yang ditemukan oleh kedua algoritma adalah sama, yang mengindikasikan bahwa kedua algoritma berhasil menghasilkan jalur yang identik untuk semua pasangan node target. Hal ini menunjukkan bahwa meskipun ada perbedaan dalam waktu komputasi, kedua algoritma memberikan hasil yang benar dan konsisten.

C. Analisis Kinerja Berdasarkan Karakteristik Algoritma dan Graf

Kecepatan superior Dijkstra iteratif dapat dijelaskan oleh perbedaan kompleksitas waktu fundamental kedua algoritma dan sifat graf yang diuji. Kompleksitas waktu Floyd-Warshall adalah $O(V^3)$, dengan V adalah total node (187 node), sehingga komputasi V^3 signifikan meskipun menghasilkan semua pasangan jarak sekaligus. Sebaliknya, Dijkstra

dengan *min-priority queue* memiliki kompleksitas O(ElogV) atau O((E+V)logV) per eksekusi dari satu sumber (E=455 edge), dan untuk matriks semua-pasangan-antar-target, dijalankan NT kali (maksimal 125). Efisiensi Dijkstra dengan *priority queue* sangat terasa pada graf jarang, seperti pada graf jaringan jalan Desa Sumberejo dengan densitas 0.0262. Pada kasus ini, karena NT<V dan graf bersifat jarang, pendekatan iteratif Dijkstra menjadi lebih efisien secara komputasi. Temuan ini sejalan dengan penelitian oleh Hendra dan Riti yang juga menemukan Dijkstra lebih efisien dari segi perhitungan pada kasus mereka[1].

D. Perbandingan dengan Studi Terdahulu dan Implikasi Praktis

Beberapa penelitian sebelumnya telah mengeksplorasi penggunaan kedua algoritma ini. Muhlasin, Noviandi, dkk. dan Nawagustia menyoroti kemampuan Floyd-Warshall dalam menangani semua pasangan lintasan yang penting untuk input optimasi *multi-stop* atau ketika semua kemungkinan jalur perlu dipertimbangkan[2][10]. Risald bahkan menggabungkan keduanya, menggunakan Dijkstra untuk waktu tercepat dan Floyd-Warshall untuk jarak terdekat, menunjukkan bahwa pemilihan algoritma dapat bergantung pada metrik optimasi spesifik[7]. Namun, untuk tugas menghasilkan matriks jarak sebagai input dengan waktu tempuh sebagai bobot utama pada skala jaringan pedesaan seperti Desa Sumberejo, hasil penelitian ini menunjukkan bahwa overhead komputasi Floyd-Warshall pada keseluruhan graf menjadi penghalang kinerja dibandingkan efisiensi Dijkstra yang dijalankan secara terfokus untuk setiap node target.

Sementara Syahputri dkk. menunjukkan manfaat Floyd-Warshall dalam optimasi rute pengangkutan sampah dengan penghematan biaya dan jarak, perlu dicatat bahwa jumlah node atau titik pemberhentian dalam kasus tersebut mungkin memiliki karakteristik atau skala yang berbeda[6]. Marlina dkk. membandingkan kedua algoritma untuk rute wisata, menemukan hasil rute yang sebagian besar sama, namun penelitian ini lebih menekankan pada efisiensi waktu komputasi dalam pembentukan matriks jaraknya[5]. Temuan dari penelitian Ramadhan dkk. yang membandingkan Floyd-Warshall dengan Prim juga menarik, dimana Floyd-Warshall unggul dalam biaya rute optimum namun Prim lebih cepat dalam membangkitkan aktif verteks[11]. ini kembali mengindikasikan bahwa parameter evaluasi sangat menentukan kesimpulan.

E. Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan. Pertama, Analisis mendalam mengenai penggunaan memori belum dilakukan dan dapat menjadi area penelitian selanjutnya. Kedua, penelitian ini hanya tahap pra-pemrosesan data input, tidak mencakup implementasi algoritma TSP/VRP. Ketiga, data kondisi jalan dan estimasi waktu tempuh bersifat statis dan tidak memperhitungkan faktor dinamis. Keempat, pengujian dilakukan pada satu dataset spesifik jaringan jalan Desa Sumberejo. generalisasi hasil ke jaringan dengan skala atau topologi yang sangat berbeda memerlukan penelitian lebih lanjut.

IV. KESIMPULAN

Menjawab tujuan penelitian untuk membandingkan kinerja waktu komputasi, penelitian ini menyimpulkan bahwa algoritma Dijkstra yang dijalankan secara iteratif merupakan metode yang secara signifikan lebih unggul dan efisien dibandingkan algoritma Floyd-Warshall untuk

pembuatan matriks jarak sebagai *input* data *Traveling Salesman Problem (TSP)* pada konteks jaringan jalan pedesaan yang jarang (*sparse*).

Kesimpulan ini didasarkan pada fakta bahwa keunggulan kinerja Dijkstra berakar dari perbedaan fundamental kompleksitas waktu dan sifat graf yang diuji. Untuk graf jarang seperti jaringan jalan Desa Sumberejo (densitas 0.0262) , kompleksitas Floyd-Warshall $O(V^3)$ yang menghitung seluruh 187 *node* menjadi tidak efisien. Sebaliknya, pendekatan iteratif Dijkstra dengan kompleksitas O(E log V) per sumber, yang hanya diaplikasikan pada *node* target (maksimal 125), terbukti jauh lebih cepat dan skalabel. Meskipun kedua algoritma menghasilkan matriks jarak dan rute yang identik, perbedaan efisiensi waktu komputasi menjadi faktor penentu.

Dengan demikian, untuk kasus pra-pemrosesan data TSP pada jaringan dengan skala dan topologi serupa (graf jarang dimana jumlah *edge* jauh lebih sedikit dari kuadrat jumlah *node*), algoritma Dijkstra iteratif direkomendasikan sebagai metode pilihan untuk mencapai kinerja komputasi yang optimal dan responsif.

DAFTAR PUSTAKA

[1] H. Hendra and Y. F. Riti, "PERBANDINGAN ALGORITMA DIJKSTRA DAN FLOYD-WARSHALL DALAM MENENTUKAN RUTE TERPENDEK STASIUN GUBENG MENUJU WISATA SURABAYA," *JIKA (Jurnal Inform.*, vol. 6, no. 3, p. 297, Oct. 2022, doi: 10.31000/jika.v6i3.6528.

[2] M. Muhlasin, N. Noviandi, R. N. Romadhon, and S. Wijaya, "ALGORITMA FLOYD-WARSHALL DALAM MENENTUKAN RUTE MULTI-STOP UNTUK EFISIENSI PENGIRIMAN BARANG," 2022. doi: 10.47007/komp.v7i02.6136.

[3] K. C. Lintang and R. Vikaliana, "Implementasi Floyd Warshall Algorithm Untuk Optimasi Distribusi J&T Express: Studi Kasus Pickup Distribution Center J&T Express Pasar Minggu," *J. Ilm. Ilmu Terap. Univ. Jambi P-ISSN*, vol. 5, pp. 93–109, 2021, doi: 10.22437/jiituj.v5i1.15416.

[4] T. M. Diansyah, D. Handoko, and corespondent author, "Penerapan Algoritma Floyd Warshall dengan Menggunakan Euclidean Distance dalam Menentukan Rute Terbaik Application Of The Floyd Warshall Algorithm Using Euclidean Distance In Determining The Best Route," 2023. doi: 10.70340/jirsi.v4i1.

[5] L. Marlina, A. Suyitno, and Mashuri, "Penerapan Algoritma Dijkstra dan Floyd-Warshall untuk Menentukan Rute Terpendek Tempat Wisata di Batang," *Unnes J. Math.*, vol. 6, no. 1, pp. 36–47, 2017, doi: 10.15294/ujm.v6i1.13544.

[6] K. Syahputri, M. S. Rahmi, R. Indah, M. T. Mangara, and Josua, "Determination of trash hauling routes using floyd warshall algorithm in medan barat district," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Sep. 2019. doi: 10.1088/1742-6596/1230/1/012051.

[7] Risald, A. Mirino, and Suyoto, *Best Routes Relection Using Dijkstra and Floyd-Warshall Algorithm*. IEEE, 2017. doi: 10.1109/ICTS.2017.8265662.

[8] I. M. A. Bhaskara, I. M. S. Kumara, I. G. W. Darma, and I. K. A. W. Raharja, "Perbandingan Algoritma Dijkstra dan Floyd-Warshall Menggunakan Software Defined Network untuk Rute Terpendek," *J. Resist.*, vol. 7, no. 1, pp. 1–9, 2024, doi: 10.31598/jurnalresistor.v7i2.1623.

[9] C. Prianto and M. Kusnadi, "Penerapan Algoritma Dijkstra Untuk Menentukan Rute Terbaik Pada Mobile E-Parking Berbasis Sistem Informasi Geografis," *J. Inform. J. Pengemb. IT*, vol. 3, no. 3, pp. 329–335, Oct. 2018, doi: 10.30591/jpit.v3i3.941.

[10] V. A. Nawagusti, "Penerapan Algoritma Floyd Warshall dalam Aplikasi Penentuan Rute TerpendekMencari Lokasi BTS (Base Tower Station) pada PT.GCI Palembang," *J. TEKNOSI*, vol. 4, pp. 81–88, 2018, doi: 10.25077/TEKNOSI.v4i2.2018.081-088.

[11] Z. Ramadhan, M. Zarlis, S. Efendi, A. Putera, and U. Siahaan, "Perbandingan Algoritma Prim Dengan Algoritma Floyd-Warshall Dalam Menentukan Rute Terpendek (Shortest Path Problem)," 2018. doi: 10.30865/jurikom.v5i2.625.