**Journal of Applied Informatics and Computing (JAIC)**
Vol.9, No.1, February 2025, pp. 202~210
e-ISSN: 2548-6861


# Optimization of Urban Waste Collection Routes Using the Held-Karp Algorithm in a Web and Mobile-Based System

**Tiara Juli Arsita [1]*, Nouval Trezandy Lapatta [2]**, Yuri Yudhaswana Joefri [3]*, Dwi Shinta Angreni [4]**, Septiano Anggun Pratama [5]***
[1,2,3,5]Informatics Engineering, Engineering Faculty, Universitas Tadulako, Indonesia
[4]Information Systems, Engineering Faculty, Universitas Tadulako, Indonesia
tiarajuliarsita@gmail.com [1], nouval @untad.ac.id [2], yuri.yudhaswana@untad.ac.id [3], ds.angreni@untad.ac.id [4], septiano@untad.ac.id [5]

**Article Info**

***Article history:***

Received 2024-11-04
Revised 2024-12-04
Accepted 2025-01-18

***Keyword:***

*Held-Karp Algorithm,*
*Shortest Route,*
*Travelling Salesman Problem,*
*Waste Management.*

**ABSTRACT**

In 2023, the Environmental Agency of Palu City recorded a total waste production of 97,492 tons, of which 10.4% was plastic waste. The Palu City Government operates a fleet of garbage trucks on a predetermined collection schedule. However, garbage bins frequently overflow before their scheduled pickup, resulting in extended waste accumulation and inefficiency. This study proposes a web and mobile-based system to enhance waste management by integrating bin condition reporting and shortest route calculation for collecting full bins. The Held-Karp algorithm is utilized to address the Travelling Salesman Problem (TSP) for determining optimal collection routes. The system was developed using Golang, Flutter, ReactJS, and a MySQL database. API functionality was validated using Postman, and overall system functionality was tested using the black-box method. A case study involving 8 test points (1 starting point, 10 waste collection points, and 1 endpoint) demonstrated that the proposed system reduces travel time by up to 21.74%, costs by 22.29%, fuel consumption by 21.16%, and distance traveled by 21.16% compared to conventional methods. These results highlight the potential of the system to significantly optimize waste collection operations and support sustainable urban waste management practices.



## I. PENDAHULUAN

Pengelolaan sampah merupakan langkah penting dalam mengatasi pencemaran lingkungan. Pada tahun 2023, Dinas Lingkungan Hidup (DLH) Kota Palu mencatat volume sampah sebesar 97.492 ton, dengan 10,4% di antaranya berupa sampah plastik [1]. Pemerintah Kota Palu telah menyediakan titik-titik Tempat Penampungan Sementara (TPS) dan armada truk pengangkut sampah yang beroperasi sesuai jadwal untuk membawa sampah ke Tempat Pembuangan Akhir (TPA). Meskipun langkah ini memberikan dampak positif, masalah TPS yang sering penuh sebelum jadwal pengangkutan tetap menjadi tantangan. Akibatnya, terjadi penumpukan sampah yang berkepanjangan, yang berpotensi menyebarkan sampah, menciptakan genangan air, dan menimbulkan dampak negatif, seperti menurunnya estetika kota, bau tidak sedap, serta peningkatan risiko penyakit [2].

Kesadaran masyarakat berperan penting dalam keberhasilan pengelolaan sampah. Untuk itu, diperlukan sistem yang dapat meningkatkan efisiensi pengelolaan, seperti pelaporan kondisi TPS secara real-time. Penelitian ini mengembangkan sistem berbasis web dan mobile yang memungkinkan masyarakat melaporkan TPS penuh melalui aplikasi mobile. Informasi ini membantu armada truk pengangkut menentukan rute optimal berdasarkan prioritas TPS yang telah dilaporkan, sehingga pengangkutan sampah dapat dilakukan lebih cepat tanpa menunggu jadwal reguler.

Penentuan rute optimal sangat penting untuk meningkatkan efisiensi biaya, bahan bakar, dan waktu kerja. Permasalahan ini dapat dianalogikan dengan Travelling Salesman Problem (TSP), di mana armada truk dianggap sebagai "salesman" dan TPS sebagai titik-titik kota yang harus dikunjungi. Penelitian

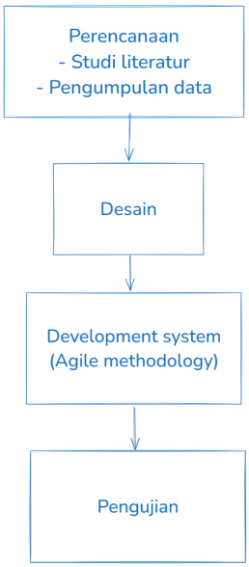
ini mengadopsi pendekatan Asymmetric Traveling Salesman Problem (ATSP) yang lebih sesuai dengan kondisi pengangkutan sampah, di mana jarak antar titik dapat berbeda untuk arah yang berlawanan [3][4][5]. Hal tersebut sesuai dengan fakta lapangan pada proses pengangkutan sampah titik awal dan titik akhir tidak selalu berada pada titik yang sama.

Berbagai algoritma tersedia untuk menyelesaikan TSP, termasuk Brute Force [6], Held-Karp (HK) [7][8] Djikstra [9][10] Genetic Algorithm (GA) [11][12][13][14]. Dari algoritma tersebut, Held-Karp dipilih karena keunggulannya dalam memberikan solusi optimal menggunakan pendekatan dynamic programming. Kompleksitas waktu Held-Karp, yaitu $O(2^n n^2)$ lebih efisien dibandingkan Brute Force $O(n!)$, sehingga lebih cocok untuk skala besar [6][7].

Dalam penelitian ini, pengembangan sistem memanfaatkan teknologi modern. Backend dibangun menggunakan bahasa pemrograman Go untuk API [15], sementara frontend untuk aplikasi web menggunakan ReactJS [16] dan aplikasi mobile menggunakan Flutter [17]. Sistem menggunakan MySQL sebagai basis data relasional [18], serta Google Maps API untuk mendapatkan jarak antar titik dan memvisualisasikan peta [10]. Penelitian ini bertujuan untuk menghasilkan sistem yang mampu memberikan solusi optimal dalam pengelolaan sampah melalui perhitungan rute pengangkutan yang lebih efisien menggunakan algoritma Held-Karp.

## II. METODE

Metode yang digunakan adalah studi literatur untuk mendapatkan algoritma, dan metode *agile* untuk pengembangan aplikasi dengan menerapkan *scrum*, yaitu *sprint planning, sprint, dan sprint review* [19]. Metode pengujian menggunakan *tools postman* dan pengujian *blackbox*. Alur pengembangan sistem aplikasi ini dijelaskan dalam Gambar 1.




Gambar 1. Alur Metode Pengembangan

Penelitian ini menggunakan pendekatan studi literatur dan metodologi Agile dengan framework Scrum untuk mendukung pengembangan aplikasi secara efektif dan efisien. Studi literatur dilakukan untuk memperoleh referensi algoritma yang relevan dalam menyelesaikan masalah Travelling Salesman Problem (TSP), dengan algoritma Held-Karp dipilih karena keunggulannya dalam memberikan solusi optimal untuk kasus Asymmetric Traveling Salesman Problem (ATSP). Selain itu, studi literatur juga mencakup penelitian terkait pengelolaan sampah dan teknologi pendukung sistem berbasis web dan mobile.

Pengembangan aplikasi dilakukan dengan metodologi Agile menggunakan Scrum, yang bersifat iteratif dan adaptif terhadap perubahan kebutuhan. Proses Scrum mencakup tiga tahapan utama, yaitu perencanaan sprint (sprint planning) untuk menentukan backlog pengembangan, pelaksanaan sprint sebagai tahap implementasi pengembangan, dan sprint review untuk mengevaluasi hasil pengembangan dan mengidentifikasi kebutuhan perbaikan.

Pengujian sistem dilakukan menggunakan dua metode utama, yaitu pengujian API menggunakan Postman untuk menguji respons, validitas data, dan integrasi antara frontend dan backend, serta pengujian blackbox untuk memastikan setiap fitur pada aplikasi berfungsi sesuai dengan spesifikasi tanpa memeriksa kode program.

Secara keseluruhan, penelitian ini diharapkan dapat menghasilkan aplikasi yang mampu memberikan solusi optimal untuk pengelolaan sampah. Aplikasi ini dirancang melalui proses pengembangan iteratif dengan Scrum dan dilengkapi dengan algoritma Held-Karp untuk menentukan rute pengangkutan sampah yang lebih optimal.

### *A. Perencanaan*

Pada tahap perencanaan ini digunakan untuk menganalisis kebutuhan data yang diperlukan selama pengembangan sistem. Studi literatur dilakukan untuk mengkaji lebih dalam terkait algoritma. Perencanaan lain terkait teknologi yang akan digunakan dalam pengembangan sistem [9]. Pengumpulan data dilakukan dengan observasi dan bekerja sama dengan DLH pemerintah kota Palu.

### *B. Desain*

Proses desain merupakan proses pengolahan data yang telah dikumpulkan pada tahap perencanaan yang akan di tampilkan dalam bentuk *user interface* pada sistem [20].

### *C. Development System*

Pada tahap ini menggunakan metode *agile* yaitu *scrum* dimana melalui tahap-tahap seperti berikut:

*1)* *Sprint Planning:* Tahap *sprint planning* merupakan salah satu dari *Software Development Life Cycle* (SDLC) yang memecah sistem aplikasi yang akan dikembangkan. *Output* dari SDLC berupa *task* dari pengembangan fitur dengan rentang waktu yang sudah ditentukan [21].

*2)* *Sprint:* Tahapan *sprint* merupakan masa atau periode pengerjaan *task* yang telah ditetapkan pada saat *sprint*

*planning*, dimana waktu dalam 1 *sprint* dilaksanakan dalam periode waktu 2 minggu [22].

*3) Sprint Review:* Tahap presentasi *task* yang telah diselesaikan selama periode 1 *sprint* sebelum melanjutkan ke *sprint* berikutnya.

### *D. Pengujian*

Pada tahap ini dilakukan dua pengujian, yaitu pengujian pada API dan pengujian fungsionalitas. Pengujian terhadap API dilakukan dengan menggunakan *tools Postman*, untuk menguji kesesuaian antara permintaan dan respons dengan mengirimkan pemintaan API dengan menggunakan *protocol* HTTP dengan memanfaatkan layanan client yaitu *JavaScript Object Notation* (JSON) [19].

Dalam pengujian fungsionalitas, digunakan metode *blackbox*. Pengujian dengan metode ini diperlukan untuk mengantisipasi kesalahan pada sistem dan memastikan sistem berjalan dengan baik [16]. Pengujian dengan metode *blackbox* dapat dilihat pada tabel 1.

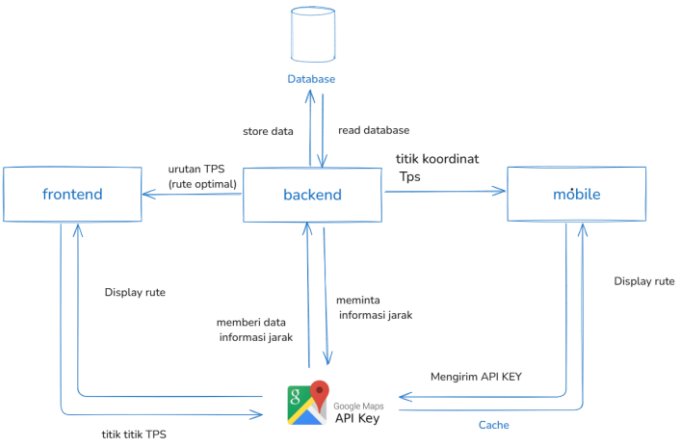
TABEL I
PENGUJIAN BLACKBOX

| Fitur | Fungsi Fitur |
|---|---|
| *User register* | *User register* untuk membuat akun |
| *User login* | *User login* untuk dapat mengakses aplikasi. |
| Admin *login* | Admin *login* untuk mengakses dan mengelola informasi atau konten pada sistem aplikasi |
| Penentuan rute tercepat (pengujian algoritma) | menampilkan rute terpendek untuk armada pengangkut sampah berdasarkan titik awal, titik akhir, dan lokasi tempat sampah. |
| Rute menuju TPS | *User* dapat mengetahui rute atau jalur menuju tempat sampah. |
| Mengelola data laporan | Admin dapat melakukan operasi *create, read, update* dan *delete* (CRUD) pada data laporan. |
| Mengelola data tempat sampah | Admin dapat melakukan operasi *create, read, update* dan *delete* (CRUD) pada data TPS |
| Daftar TPS | User dapat mengetahui TPS yang ada di sekitarnya berserta kondisi TPS tersebut. |
| Report tempat sampah | User dapat melaporkan TPS yang sudah penuh. |
| Riwayat laporan | User dapat mengetahui riwayat laporannya. |
| Pencarian | Admin dan user dapat menggunakan semua fitur pencarian pada sistem. |

## III. ANALISIS DAN PERANCANGAN

### *A. Desain Sistem*

Perancangan Design sistem atau arsitektur sistem untuk membangun pencarian jalur tercepat dengan algoritma HK dalam dilihat pada gambar 2 berikut.




Gambar 2. Desain Sistem

Proses analisis sistem dilakukan untuk memahami kebutuhan fungsional dan non-fungsional aplikasi yang dirancang. Analisis ini mencakup identifikasi masalah, kebutuhan pengguna, serta spesifikasi teknis yang diperlukan untuk mendukung sistem. Berdasarkan data dari Dinas Lingkungan Hidup Kota Palu, permasalahan utama yang dihadapi adalah penumpukan sampah di Tempat Pembuangan Sampah Sementara (TPS) sebelum jadwal pengangkutan tiba. Masalah ini berdampak negatif terhadap lingkungan dan kesehatan masyarakat sekitar.

Untuk mengatasi masalah tersebut, dirancang sebuah sistem berbasis web dan mobile yang memiliki fitur utama:

1) Pelaporan kondisi TPS: Pengguna dapat melaporkan kondisi TPS yang sudah penuh melalui aplikasi mobile.
2) Penentuan rute optimal: Sistem menentukan rute terbaik untuk armada pengangkut sampah menggunakan algoritma Held-Karp.
3) Visualisasi rute: Rute optimal divisualisasikan melalui Google Maps API pada aplikasi web dan mobile.

Kebutuhan teknis sistem meliputi pengembangan backend dengan Golang untuk Application Programming Interface (API), frontend dengan ReactJS untuk aplikasi web, Flutter untuk aplikasi mobile, dan MySQL sebagai basis data relasional.

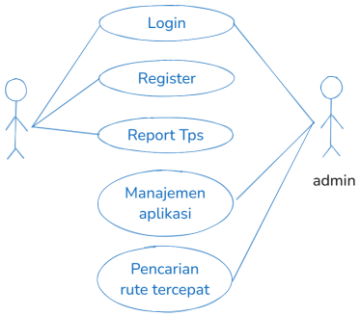
### *B. Use Case Diagram*

Sistem ini dikembangkan dengan berbasis *mobile* yang digunakan *user* dan *website* yang digunakan oleh admin. Perancangan sistem mencakup tiga aspek utama, yaitu perancangan arsitektur sistem, perancangan alur kerja, dan perancangan antarmuka pengguna. Sistem dirancang menggunakan arsitektur berbasis client-server, di mana aplikasi mobile dan web berfungsi sebagai client yang berkomunikasi dengan server melalui API. Arsitektur ini memungkinkan pemisahan yang jelas antara frontend dan backend, sehingga memudahkan pengembangan dan pemeliharaan sistem.

Alur kerja sistem mencakup tiga proses utama yang saling terintegrasi. Pertama, pelaporan kondisi TPS dilakukan oleh pengguna melalui aplikasi mobile, di mana mereka dapat

melaporkan TPS yang penuh. Data laporan ini kemudian disimpan dalam basis data dan diteruskan ke server untuk diproses lebih lanjut. Kedua, pengolahan data rute dilakukan oleh server menggunakan algoritma Held-Karp. Algoritma ini digunakan untuk menghitung rute optimal berdasarkan laporan TPS yang masuk, sehingga efisiensi rute dapat ditingkatkan. Ketiga, hasil perhitungan rute optimal divisualisasikan melalui aplikasi web dan mobile, mempermudah operator armada truk dalam mengambil keputusan terkait pengangkutan sampah.

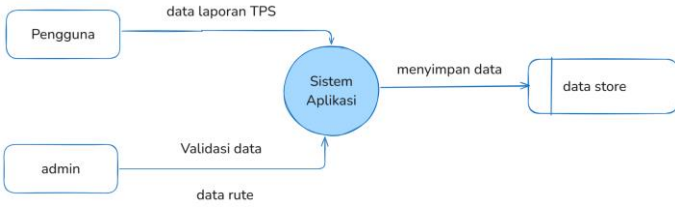
Perancangan antarmuka pengguna pada aplikasi ini dirancang dengan fokus pada kemudahan penggunaan dan aksesibilitas. Tiga komponen antarmuka utama menjadi prioritas dalam perancangan. Pertama, halaman pelaporan pada aplikasi mobile menyediakan form sederhana yang memungkinkan pengguna dengan mudah melaporkan kondisi TPS. Kedua, dashboard pengelolaan pada aplikasi web menyediakan fitur untuk memantau laporan TPS dan rute optimal yang dihasilkan. Ketiga, peta interaktif digunakan untuk menampilkan rute pengangkutan sampah secara visual dengan memanfaatkan Google Maps API, memberikan gambaran yang jelas dan informatif kepada operator. Kombinasi fitur ini dirancang untuk menciptakan pengalaman pengguna yang intuitif dan mendukung efisiensi sistem secara keseluruhan. Berikut ini *use case diagram* pada sistem aplikasi *mobile* dan *website*.




Gambar 3. Use Case Diagram

### C. *Data Flow Diagram*

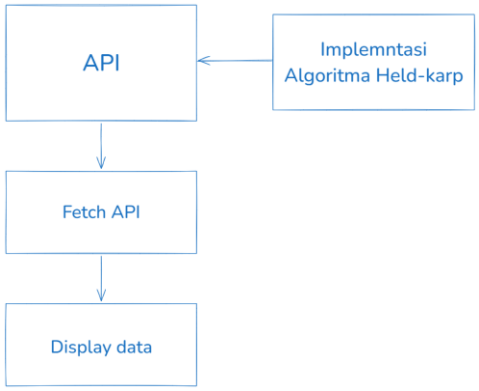
Berikut ini *data flow diagram* pada sistem aplikasi.




Gambar 4. Data Flow Diagaram

### D. *Penentuan Rute Pengangkutan Sampah*

Berikut adalah perhitungan algoritma HK yang diimplementasikan pada API.




Gambar 5. Alur Implementasi Algoritma

Data yang digunakan dalam penelitian ini berasal dari wilayah Kecamatan Mantikulore. Lokasi TPS diperoleh dari Dinas Lingkungan Hidup (DLH) Pemerintah Kota Palu, yang mencakup satu titik awal, sembilan lokasi TPS, dan satu tempat pembuangan akhir (TPA), sehingga total terdapat sebelas titik pengujian. Informasi jarak antara titik-titik tersebut diperoleh dengan memanfaatkan layanan Google Maps API, yang menyediakan data akurat dalam satuan kilometer (km). Setiap titik lokasi memiliki peran tertentu dalam sistem pengangkutan sampah, sebagaimana dirangkum pada Tabel II berikut.

TABEL II
TITIK KOORDINAT PENGUJIAN

| Kode | Keterangan |
|---|---|
| A | Titik Awal |
| B | Kantor Wali Kota |
| C | Rumah Sakit Tkt. III Dr. Shindu Trisno |
| D | SD Al Azhar |
| E | Layana Dupa |
| F | Jl. Una-una |
| G | Huntap Tondo Perdos |
| H | Hutan kota |
| I | UNISMUH |
| J | TPA Kawatuna |

Setiap node (titik) memiliki koordinat geografis yang menentukan posisi lokasinya dalam peta. Berdasarkan node-node tersebut, data jarak antara satu node dengan node lainnya dihitung menggunakan Google Maps API untuk memastikan akurasi pengukuran. Data jarak ini menjadi dasar dalam menentukan rute optimal pengangkutan sampah. Jarak antar node tersebut disajikan dalam Tabel III berikut untuk memberikan gambaran lebih rinci tentang koneksi antara titik-titik pengujian. Informasi ini menjadi elemen penting dalam proses analisis dan pengolahan algoritma Held-Karp untuk menyelesaikan permasalahan Travelling Salesman Problem (TSP) pada penelitian ini.

TABEL III
JARAK SETIAP TITIK LOKASI

| x, y | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 0 | 0.5 | 1.6 | 1.8 | 11.9 | 3.6 | 10.4 | 5.6 | 5.2 | 6.8 |
| **B** | 0.5 | 0 | 1 | 1.3 | 11.4 | 3 | 9.9 | 5.1 | 4.6 | 6.9 |
| **C** | 1.0 | 0.9 | 0 | 1.5 | 10.9 | 2.6 | 10 | 4.6 | 4.2 | 7.5 |
| **D** | 1.8 | 1.3 | 1.3 | 0 | 11.2 | 3.6 | 9.7 | 4.9 | 4.5 | 7.8 |
| **E** | 10.8 | 10.8 | 9.8 | 10.6 | 0 | 11.4 | 4.1 | 8.3 | 7.9 | 16.5 |
| **F** | 3.1 | 3 | 2.8 | 3.6 | 11.3 | 0 | 11.5 | 4.2 | 4.2 | 9.6 |
| **G** | 10.1 | 9.5 | 9.1 | 9.3 | 6.3 | 11 | 0 | 7.6 | 7.2 | 13.6 |
| **H** | 5.2 | 5.2 | 4.3 | 5 | 7.8 | 4.5 | 6.9 | 0 | 0.7 | 10.9 |
| **I** | 4.5 | 4.4 | 3.6 | 4.3 | 8.1 | 4 | 8.1 | 1 | 0 | 12.4 |
| **J** | 6.8 | 7 | 7.6 | 7.8 | 16.9 | 9.6 | 14.0 | 11.1 | 10.6 | 0 |

Notasi yang digunakan dalam algoritma Held-Karp dapat dinyatakan dengan fungsi dp(S, i), di mana:

- S merupakan himpunan bagian (subset) dari semua titik yang telah dikunjungi, dan
- i adalah titik terakhir dalam urutan kunjungan.

Dalam penelitian ini, algoritma Held-Karp dimodifikasi untuk tidak kembali ke titik awal setelah menyelesaikan kunjungan ke semua titik. Modifikasi ini dilakukan untuk menyesuaikan dengan skenario pengangkutan sampah, di mana truk pengangkut tidak perlu kembali ke titik awal setelah mencapai tempat pembuangan akhir (TPA). Dengan demikian, perumusan fungsi dinamisnya menjadi:

$$dp[Semua\ kota][Titik\ akhir] \quad (1)$$

Selain itu, solusi optimal tidak mencakup jalur kembali ke titik awal, sehingga perhitungan akhir cukup mempertimbangkan nilai minimum dari dp(S, i) untuk semua titik akhir yang memungkinkan, tanpa kembali ke titik awal. Pendekatan ini memastikan bahwa rute yang dihasilkan lebih efisien sesuai dengan kebutuhan spesifik skenario pengangkutan sampah pada penelitian ini.

Pada algoritma Held-Karp, inisialisasi dimulai dengan memilih node A sebagai titik awal, dengan nilai biaya mencapai node A dari node A itu sendiri adalah 0. Dalam hal ini, kita menggunakan notasi.

$$dp[\{A\}][A] = 0$$

Menunjukkan bahwa biaya untuk berada di node A pada titik awal adalah nol, karena tidak ada jarak yang perlu ditempuh. Selanjutnya, algoritma melakukan iterasi untuk menghitung biaya ke setiap node yang akan dikunjungi berikutnya dari node A. Setiap kemungkinan subset titik yang akan dikunjungi dihitung berdasarkan jarak antar node. Berikut adalah langkah-langkah perhitungannya untuk setiap subset yang melibatkan node A:

Subset S = {A, B}, terakhir di B:

$S = \{A, B\}$ S = {A, B}, terakhir di $B$ B:

$$dp[\{A, B\}][B] = dp[\{A\}][A] + \text{dist}[A][B] = 0 + 0.5 = 0.5$$

Artinya, biaya untuk mencapai node B setelah A adalah 0.5 km. Subset S = {A, C}, terakhir di C:

$S = \{A, C\}$ S = {A, C}, terakhir di $C$ C:

$$dp[\{A, C\}][C] = dp[\{A\}][A] + \text{dist}[A][C] = 0 + 1.6 = 1.6$$

Biaya untuk mencapai node C setelah A adalah 1.6 km.

$S = \{A, D\}$ S = {A, D}, terakhir di $D$ D:

$$dp[[\{A, D\}][D] = dp[\{A\}][A] + \text{dist}[A][D] = 0 + 1.8 = 1.8$$

Biaya untuk mencapai node D setelah A adalah 1.8 km.

$S = \{A, E\}$ S = {A, E}, terakhir di $E$ E:

$$dp[\{A, E\}] [E] = dp\ [\{A\}] [A] + \text{dist}[A][E] = 0 + 11.9 = 11.9$$

Biaya untuk mencapai node E setelah A adalah 11.9 km.

$S = \{A, F\}$ S = {A, F}, terakhir di $F$ F:

$$dp[\{A, F\}][F] = dp[\{A\}][A] + \text{dist}[A][F] = 0 + 3.6 = 3.6$$

Biaya untuk mencapai node F setelah A adalah 3.6 km.

$S = \{A, G\}$ S = {A, G}, terakhir di $G$ G:

$$dp[\{A, G\}][G] = dp[\{A\}][A] + \text{dist}[A][G] = 0 + 10.4 = 10.4$$

Biaya untuk mencapai node G setelah A adalah 10.4 km.

$S = \{A, H\}$ S = {A, H}, terakhir di $H$ H:

$$dp[\{A, H\}][H] = dp[\{A\}][A] + \text{dist}[A][H] = 0 + 5.6 = 5.6$$

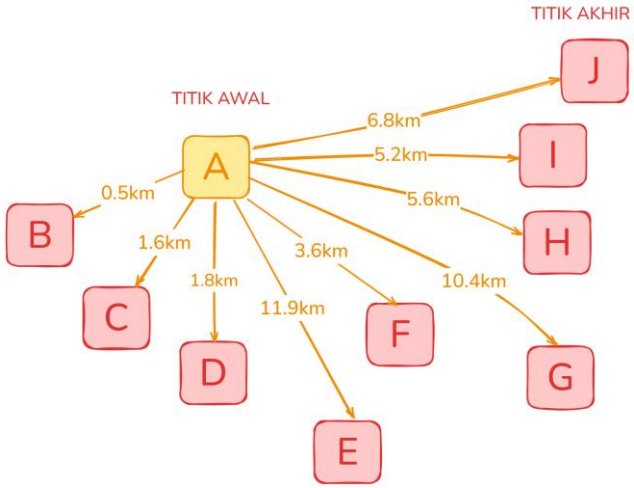
Biaya untuk mencapai node H setelah A adalah 5.6 km.

$S = \{A, I\}$ S = {A, I}, terakhir di $I$ I:

$dp$[{A, I}] [I] = $dp$ [{A}] [A] + dist[A][I] = 0+5.2 = 5.2
Biaya untuk mencapai node I setelah A adalah 5.2 km.

$S$ = {$A$, $J$} S = {A, J}, terakhir di $J$ J:
$dp$[{A, J}] [J] = $dp$ [{A}] [A] + dist[A][J] = 0 + 6.8 = 6.8
Biaya untuk mencapai node J setelah A adalah 6.8 km.

Dengan mengikuti langkah-langkah ini, kita memperoleh biaya perjalanan dari node A ke setiap titik lain yang ada dalam subset, dan dapat melanjutkan perhitungan untuk langkah-langkah berikutnya dalam algoritma Held-Karp. Grafik yang dihasilkan akan menunjukkan representasi visual dari jarak antar titik-titik tersebut dan membantu dalam menentukan rute optimal yang harus diambil oleh sistem. Dapat dilihat pada graf berikut.




Gambar 6. Subset dengan dua kota

Proses iterasi dalam menghitung biaya minimum untuk setiap subset kota yang dikunjungi menggunakan algoritma Held-Karp. Proses ini dilakukan secara bertahap untuk setiap subset yang berisi kombinasi titik yang berbeda, dengan tujuan untuk menemukan jalur terpendek dari titik awal ke titik tujuan.

Subset dengan 3 kota:
Untuk subset S = {A, B, D}, titik terakhir yang dikunjungi adalah D. Biaya minimum dihitung sebagai kombinasi biaya untuk mengunjungi kota-kota sebelumnya ditambah jarak dari titik B ke D:
Untuk S = {A, B, D}, j=D:
$C(\{A, B, D\}, D) = min[C(\{A, B\}, B) + d(B, D)] = 0.5 + 1.3 = 1.8$
Iterasi ini akan terus berlanjut untuk kombinasi lainnya.

Pada subset S = {A, B, D, C}, dengan titik terakhir C, biaya dihitung dengan menambahkan jarak dari D ke C setelah menghitung biaya minimum sebelumnya.

Subset dengan 4 kota
Untuk S = {A, B, D, C}, j=C:
$C(\{A, B, D, C\}, C) = min[C(\{A, B, D\}, D) + d(D, C)] = 1.8 + 1.3 = 3.1$
Lanjutkan ke iterasi yang lain.

Subset dengan 5 kota:
Untuk S = {A, B, D, C, F}, j=F:
$C(\{A, B, D, C, F\}, F) = min[C(\{A, B, D, C\}, C) + d(C, F)] = 3.1 + 2.6 = 5.7$
Lanjutkan ke iterasi yang lain.

Subset dengan 6 kota:
Untuk S = {A, B, D, C, F, I}, j=I:
$C(\{A, B, D, C, F, I\}, I) = min[C(\{A, B, D, C, F\}, F) + d(F, I)] = 5.7+4.2=9.9$km
Lanjutkan ke iterasi yang lain.

Subset dengan 7 kota:
Untuk S = {A, B, D, C, F, I, H}, j=H:

$C(\{A, B, D, C, F, I, H\}, H) = min[C(\{A, B, D, C, F, I\}, I) + d(I, H)] = 9.9+1=10.9$
Lanjutkan ke iterasi yang lain.

Subset dengan 8 kota:
Untuk S = {A, B, D, C, F, I, H, E}, j=E:
$C(\{A, B, D, C, F, I, H, E\}, E) = min[C(\{A, B, D, C, F, I, H\}, H) + d(H, E)] = 10.9 +7.8 =18.7$
Lanjutkan ke iterasi yang lain.

Subset dengan 9 kota:
Untuk S = {A, B, D, C, F, I, H, E, G}, j=G:
$C(\{A, B, D, C, F, I, H, E, G\}, G) = min[C(\{A, B, D, C, F, I, H, E\}, E) + d(E, G)] = 18.7+4.1 = 22.8$
Lanjutkan ke iterasi yang lain.

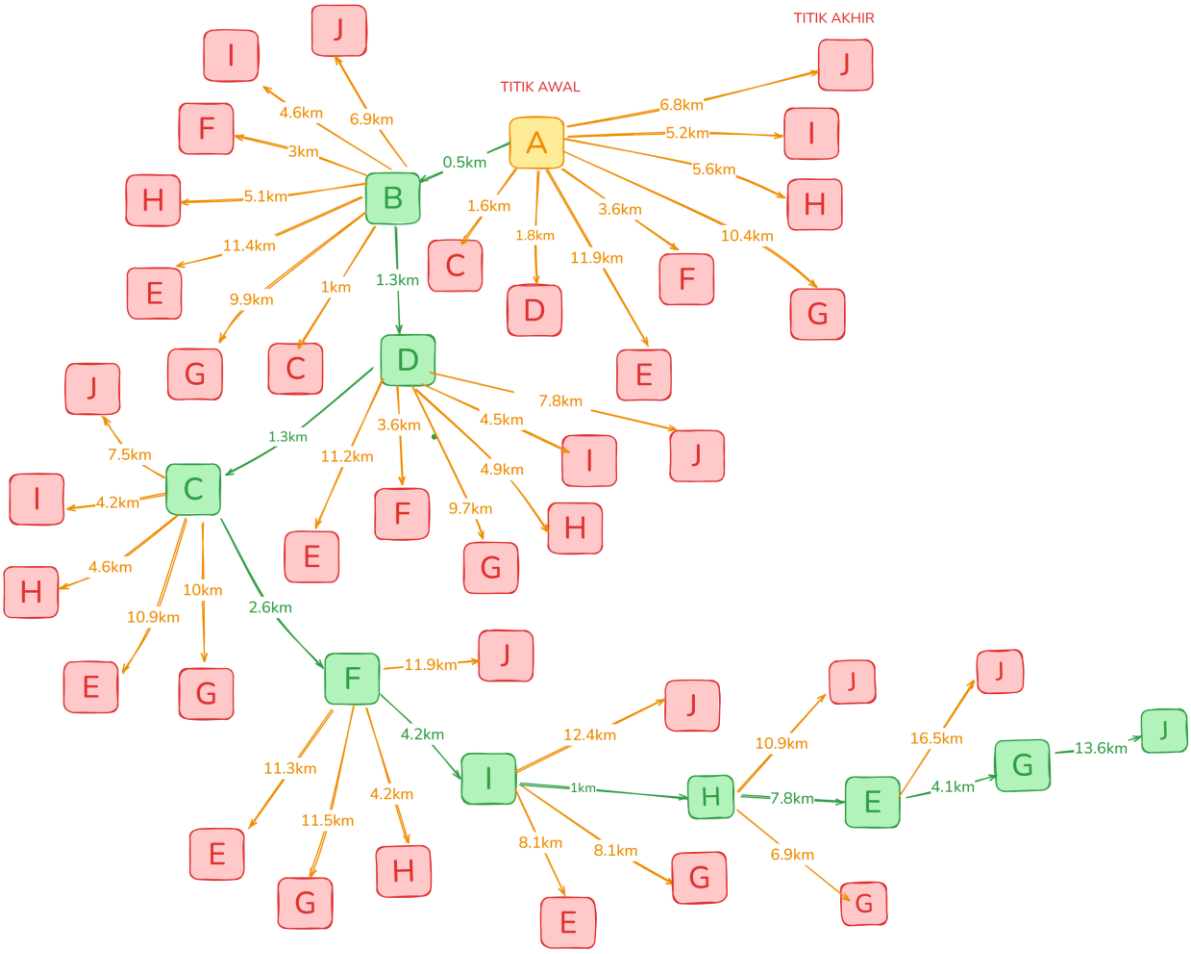
Subset dengan 10 kota:
$C(\{A, B, D, C, F, I, H, E, G, J\}, J) = \min[C(\{A, B, D, C, F, I, H, E, G\}, G) + d(G, J)] = 22.8+13.6 = 36.4$ (Biaya minimum)
Lanjutkan ke iterasi yang lain.

Rute terpendek: A-> B -> D -> C -> F -> I -> H -> E -> G -> J = 36.4

Setelah menghitung biaya minimum untuk semua subset, rute terpendek yang ditemukan adalah:

$A \rightarrow B \rightarrow D \rightarrow C \rightarrow F \rightarrow I \rightarrow H \rightarrow E \rightarrow G \rightarrow J$ dengan total biaya minimum sebesar 36.4 km. Proses ini menunjukkan bagaimana algoritma Held-Karp secara iteratif menghitung biaya minimum untuk setiap subset titik yang dikunjungi, sehingga pada akhirnya dapat ditemukan rute

yang paling efisien dalam hal jarak. Rute terpendek dapat dilihat pada graf berikut.




Gambar 6. Graf Rute Optimal

## IV. HASIL DAN PEMBAHASAN

Dari penelitian yang telah dilakukan, algoritma HK yang diimplementasikan pada pencarian rute optimal menunjukkan hasil yang efektif, dimana dari pengujian menunjukkan hasil bahwa semua fitur dapat berfungsi dengan baik dan kode program yang ditulis sudah terstruktur dengan baik. Algoritma ini berhasil menentukan rute optimal dengan pemilihan jalur yang tepat. Rute yang dihasilkan berdasarkan hasil uji coba adalah titik awal - Kantor Wali Kota - Rumah Sakit Tkt. III Dr. Shindu Trisno - SD Al Azhar - Jl. Una una - Unismuh - Hutan Kota - Layana dupa - Huntap Tondo - Titik Akhir.

Berikut perbandingan menggunakan rute konvensional dan rute yang di hasilkan oleh HK.

TABEL III
PERBANDINGAN METODE KONVENSIONAL DAN ALGORITMA HK

| Jarak (km) | Waktu (40km/jam) | Metode | Biaya (Rupiah) | Bahan Bakar (liter) | Rute |
|---|---|---|---|---|---|
| 36.4 | 54 menit | HK | 61.000 | 9.1 | A, B, D, C, F, I, H, E, G, J |
| 42.3 | 63 menit | konvensional | 72.000 | 10.5 | A, B, I, F, C, D, H, E, G, J |
| 50 | 75 menit | konvensional | 85.000 | 12.5 | A, D, C, H, F, I, B, E, G, J |

Dari perbandingan yang dilakukan, dapat disimpulkan bahwa penggunaan algoritma Held-Karp (HK) berhasil mengefisienkan waktu, biaya, dan penggunaan bahan bakar. Penurunan yang signifikan terlihat pada berbagai aspek sebagai berikut.

*Penurunan jarak*:

$$\frac{((42{,}3\ km + 50\ km) : 2) - 36.4\ \text{km}}{((10.5\ \text{liter} + 12.5\ \text{liter}) \div 2)}\ x\ 100 = 21.16\ \%$$

Perhitungan ini menunjukkan penurunan jarak sebesar 21.16% dibandingkan dengan rute sebelumnya.

*Penurunan waktu*:

$$\frac{((63\ \text{menit} + 75\ \text{menit}) \div 2) - 54\ \text{menit}}{((63\ \text{menit} + 75\ \text{menit}) : 2)}\ x\ 100 = 21.74\%$$

Algoritma HK berhasil mengurangi waktu tempuh hingga 21.74%.

*Penurunan bahan bakar*:

$$\frac{((10.5\ \text{liter}\ +\ 12.5\ \text{liter})\ \div\ 2) - 9.1}{(10.5\ \text{liter}\ +\ 12.5\ \text{liter})\ \div\ 2}\ x\ 100 = 20.16\ \%$$

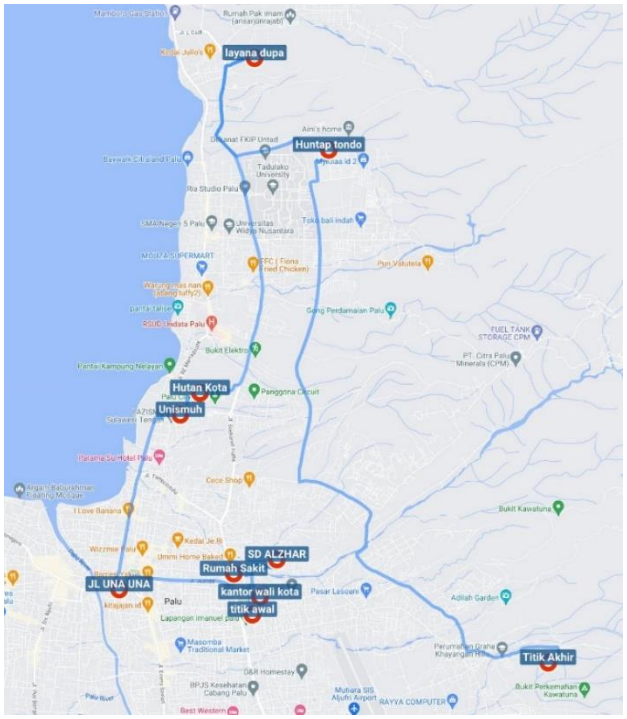
Penggunaan bahan bakar juga mengalami penurunan sebesar 20.16%.

*Penurunan biaya*:

$$\frac{((Rp\ 72.000\ +\ Rp\ 85.000)\ \div\ 2\ ) - 61.000}{(Rp\ 72.000\ +\ Rp\ 85.000)\ \div\ 2} x100 = 22.29\%$$

Biaya total mengalami penurunan sebesar 22.29%, yang merupakan penghematan signifikan.

Hasil uji coba ini menunjukkan bahwa penggunaan algoritma Held-Karp dengan solusi optimal pada sistem mampu memberikan penurunan yang substansial dalam berbagai aspek. Secara keseluruhan, algoritma HK berhasil mengurangi biaya sebesar 22.29%, jarak sebesar 21.16%, waktu sebesar 21.74%, dan bahan bakar sebesar 20.16%. Dengan jumlah titik sebesar n = 10, algoritma HK terbukti efektif dalam menemukan rute optimal dengan kompleksitas waktu yang masih dapat diterima. Hal ini menjadikan algoritma HK sangat cocok untuk diterapkan pada kasus dengan jumlah titik yang terbatas atau dalam skala kecil hingga menengah.




Gambar 7. Visualisasi Rute pada map

Hasil yang ditampilkan pada website dapat dilihat pada Gambar 7. Gambar tersebut menunjukkan urutan titik-titik yang harus dikunjungi oleh armada truk pengangkut sampah, beserta rute optimal yang sebaiknya dilalui. Pada tampilan website, terlihat bahwa semua titik dikunjungi tepat satu kali, mengikuti rute yang telah dihitung menggunakan algoritma untuk memastikan efisiensi dalam waktu dan jarak tempuh. Dengan visualisasi ini, operator armada dapat dengan mudah mengidentifikasi jalur terbaik untuk pengangkutan sampah, meningkatkan efektivitas dan efisiensi operasional.

## V. KESIMPULAN

Dari hasil penelitian yang telah dilakukan, implementasi algoritma Held-Karp (HK) dalam menyelesaikan masalah Asymmetric Traveling Salesman Problem (ATSP) menunjukkan kinerja yang sangat baik dan mampu memberikan solusi optimal. Algoritma ini berhasil menemukan rute optimal untuk n = 10 titik dengan kompleksitas waktu yang masih dapat diterima, menjadikannya cocok untuk diterapkan pada masalah dengan jumlah titik yang terbatas. Meskipun algoritma HK memiliki kompleksitas eksponensial, pada skala kecil hingga menengah, algoritma ini dapat memberikan solusi yang efisien dan efektif, sehingga dapat digunakan sebagai alternatif dalam penyelesaian ATSP dengan batasan jumlah titik yang spesifik.

## DAFTAR PUSTAKA


[1] M. Ridwan, "DLH Palu: Presentase sampah plastik di Kota Palu 10,4 persen." [Online]. Available: https://sulteng.antaranews.com/berita/277314/dlh-palu-presentase-sampah-plastik-di-kota-palu-104-persen

[2] E. Utari, M. Fatimatuzzahra, M. Pramaisyella, S. Jaedah, and T. Triana, "Analisis Pengelolaan Sampah Akibat Pertumbuhanpenduduk Dan Perkembangan Pembangunandi Kelurahan Cipare Kota Serang," vol. 10, no. 1, pp. 556–562, 2022.

[3] Z. Ursani and A. A. Ursani, "Augmented tour construction heuristics for the travelling salesman problem," vol. 4, no. 2, pp. 131–144, 2023.

[4] J. Nalepa, *Smart Delivery System Solving Complex Vehicle Routing Problems*. Elsevier, 2020.

[5] A. Izzah, B. A. Nugroho, W. F. Mahmudy, and F. A. Bachtiar, "Optimasi Asymmetric City Tour di Kota Kediri Menggunakan Ant Colony System," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 9, no. 1, 2020.

[6] M. Kurniawan, Farida, and A. Siti, "Rute Terpendek Algoritma Particle Swarm Optimization Dan Brute Force Untuk Optimasi Travelling Salesman Problem Muchamad," *J. Tek. Inform.*, vol. 14, no. 2, pp. 191–200, 2021.

[7] A. Held-karp, H. Alfin, K. Diah, and K. Wardhani, "Aplikasi Android Untuk Mencari Jalur Tercepat Pada Pengiriman Barang Dengan Algortma Held-Karp," *J. Appl. Informatics Comput.*, vol. 4, no. 2, 2020.

[8] G. Righini and G. Righini, "Efficient optimization of the Held–Karp lower bound," *Open J. Math. Optim.*, vol. 2, pp. 0–17, 2021.

[9] S. Lestari, A. P. Giovani, and D. Dwijayanti, "Dijkstra Algorithm Implementation In Determining Shortest Route To Mosque In Residential Citra Indah City," *J. PILAR Nusa Mandiri*, vol. 16, no. 1, pp. 65–70, 2020, doi: 10.33480/pilar.v16i1.1199.

[10] Suardinata, R. Rusmi, and M. A. Lubis, "Determining Travel Time and Fastest Route Using Dijkstra Algorithm and Google Map," *Sist. J. Sist. Inf.*, vol. 11, no. 1, pp. 496–505, 2022.

[11] D. F. Gimnastian, R. Yasirandi, and D. Oktaria, "Analysis and Design of a Route Recommendation System and Bicycle Rental Fees at Tourist Destinations with Genetic Algorithms," *J. Inform. BUDIDARMA*, vol. 6, no. April, pp. 837–847, 2022, doi: 10.30865/mib.v6i2.3749.

[12] M. Arifin, "Genetic Algorithm Approach to Logistics Transportation and Distribution Problems : A Case Study of Parcel Delivery Services," *J. Optimasi Sist. Ind.*, vol. 14, no. 2, pp. 122–128, 2021.

[13] H. M. Asih, S. A. Rahman, K. Usandi, Q. A. E. Saputra, and A. Della, "Enhancing logistics e fficiency : A case s tudy of genetic algorithm- based route optimization in distribution problem," *J. Optimasi Sist. Ind.*, vol. 16, no. 2, 2023.

[14] S. Cokrowibowo and A. Irianti, "Model Penentuan Rute Terpendek Penjemputan Sampah Menggunakan Metode MTSP dan Algoritma Genetika," *J. Appl. Comput. Sci. Technol. ( JACOST )*, vol. 2, no. 1, pp. 43–48, 2021.

[15] R. Sistem, "Jurnal RESTI Implementasi Golang dan New Simple Queue pada Sistem Sandbox Pihak Ketiga Berbasis REST API," vol. 1, no. 10, pp. 7–8, 2021.

[16] M. Jindan, H. R. Ngemba, S. Hendra, and R. Laila, "Integrated Web-based Palu City Blood Donor Service Application Model Using ReactJS and ExpressJS," vol. 5, no. 3, pp. 1–8, 2023.

[17] J. O. Notohamidjojo, K. Blotongan, and K. Sidorejo, "Aplikasi Penjualan Tiket Kelas Pelatihan Berbasis Mobile menggunakan Flutter," vol. 5, pp. 281–295, 2019.

[18] A. R. Dzikrillah, "Database-Based Gui System To Increase The Effectiveness Of Student Data Management In The Fkip Uhamka Dormitory," vol. 5, no. 4, pp. 21–31, 2024.

[19] K. Nisa, C. Hapsari, Y. Y. Joefrie, and S. Hendra, "MERN Implementation in Online Quiz Applications to Recognize and Avoid Social Media Hoaxes," vol. 6, no. 2, pp. 1–8, 2024.

[20] P. D. Atika, A. Yunizar, P. Yusuf, and F. Nidaul, "Android-Based Shortest Path Finding Using A-Star ( A *) Algorithm in Bekasi City," vol. 9, no. 28, pp. 197–210, 2021.

[21] M. H. Muslim, F. Harvianto, and S. Utama, "Penerapan Metode SCRUM dalam Pengembangan Sistem Informasi Layanan Kawasan," vol. 6, pp. 365–378, 2020.

[22] R. Sistem, "Pengembangan Aplikasi Tiga-Tingkat Menggunakan Metode Scrum pada Aplikasi Presensi Karyawan Glints Academy," vol. 5, pp. 169–176, 2022.