

PENCARIAN RUTE PERJALANAN KURIR DENGAN SQUIRREL SEARCH ALGORITHM



Disusun Oleh:

N a m a : Raditya Hafidh Erlangga Riyanto
NIM : 22523086

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENCARIAN RUTE PERJALANAN KURIR DENGAN

SQUIRREL SEARCH ALGORITHM

TUGAS AKHIR



N a m a : Raditya Hafidh Erlangga Riyanto
NIM : 22523086

Yogyakarta, 23 April 2026

Pembimbing,

(Zainudin Zukhri, S.T, M.IT)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENCARIAN RUTE PERJALANAN KURIR DENGAN SQUIRREL SEARCH ALGORITHM

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 23 April 2026

Tim Penguji
Zainudin Zukhri, S.T., M.IT

Anggota 1

Ahmad Fathan Hidayatullah, S.T., M.Cs.,
Ph.D.

Anggota 2

Taufiq Hidayat, S.T., M.C.S., Ph.D.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D..)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Raditya Hafidh Erlangga Riyanto

NIM : 22523086

Tugas akhir dengan judul:

PENCARIAN RUTE PERJALANAN KURIR DENGAN SQUIRREL SEARCH ALGORITHM

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 23 April 2026



(Raditya Hafidh Erlangga Riyanto)

HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan ke hadirat Allah Subhanahu Wa Ta'ala, atas rahmat, izin, dan ridha-Nya, hingga akhirnya langkah ini sampai pada titik yang saya perjuangkan, menyelesaikan tugas akhir sebagai bagian dari perjalanan meraih gelar Sarjana Informatika di Universitas Islam Indonesia. Karya ini saya persembahkan pertama untuk diri saya sendiri. Untuk semua usaha yang mungkin tidak selalu terlihat, untuk rasa lelah yang pernah datang, dan untuk keberanian yang tetap bertahan sampai hari ini. Terima kasih sudah tidak menyerah, sudah terus berjalan, dan sudah percaya bahwa semua ini akan ada hasilnya. Untuk ibu tercinta dan kakak, terima kasih atas doa yang tidak pernah berhenti, dukungan yang selalu ada, dan kasih sayang yang tidak pernah berkurang. Tanpa kalian, perjalanan ini tidak akan sampai sejauh ini. Untuk sahabat dan teman-teman seperjuangan, terima kasih sudah menjadi bagian dari cerita ini. Terima kasih atas tawa, dukungan, dan kebersamaan yang membuat semuanya terasa lebih ringan. Semoga tulisan sederhana ini bisa jadi langkah awal untuk hal-hal baik ke depannya, bisa membanggakan orang-orang yang saya sayangi, dan menjadi pengingat bahwa perjalanan ini belum selesai.

HALAMAN MOTO

“Karena sesungguhnya sesudah kesulitan itu ada kemudahan.

Sesungguhnya sesudah kesulitan itu ada kemudahan.”

-Q.S. Al-Insyirah: 5-6

“The only thing we have to fear is fear itself.”

-Franklin D. Roosevelt

KATA PENGANTAR

Assalamu'alaikum warahmatullahi wabarakatuh.

Alhamdulillahirabbil 'alamin, segala puji dan syukur saya panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya. Shalawat serta salam semoga senantiasa tercurah kepada Nabi Muhammad SAW, beserta keluarga, sahabat, dan seluruh umatnya yang istiqamah dalam ketaatan kepada Allah SWT. Berkat izin dan karunia-Nya, saya dapat menyelesaikan Tugas Akhir yang berjudul :

“Pencarian Rute Perjalanan Kurir Dengan Squirrel Search Algorithm”

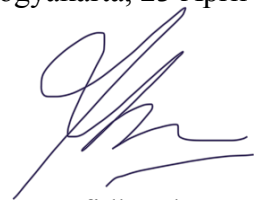
Oleh karena itu, dengan penuh rasa hormat dan ketulusan, saya ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, Sang Maha Pemilik Arah. Terimakasih telah menjadi rumah tempat saya pulang saat dunia terasa begitu kabur.
2. Bapak Zainudin Zukhri, S.T., M.IT., selaku dosen pembimbing. Terima kasih atas kesabaran dalam membimbing, ketulusan dalam mengarahkan, serta kepercayaan yang diberikan di saat saya sendiri sering meragukan langkah saya. Bimbingan Bapak bukan hanya tentang akademik, tetapi juga tentang bagaimana bertahan dalam proses
3. Keluarga besar Informatika UIL. Terima kasih telah menjadi tempat saya bertumbuh, belajar, dan mengenal arti perjuangan bersama. Setiap dinamika yang ada telah membentuk saya menjadi pribadi yang lebih kuat dan lebih siap menghadapi dunia.
4. Ibu saya tersayang. Sosok paling hangat yang selalu menjadi rumah, bahkan ketika dunia terasa asing dan langkah terasa goyah. Ibu adalah tempat pertama saya belajar tentang arti bertahan, tentang bagaimana tetap kuat dalam diam, dan bagaimana mencintai tanpa syarat. Setiap doa yang Ibu panjatkan, meski tidak selalu saya dengar, selalu saya rasakan sebagai kekuatan yang menuntun saya sampai di titik ini. Dalam setiap lelah yang hampir membuat saya berhenti, selalu ada bayangan Ibu yang membuat saya kembali melangkah. Ibu selalu menjadi alasan saya mengapa saya tidak pernah benar-benar menyerah. Karya ini tidak akan pernah cukup untuk membalas semua yang telah Ibu berikan, namun semoga menjadi salah satu hal kecil yang bisa membuat Ibu tersenyum bangga.
5. Kakak saya tercinta, *role model* saya, dan inspirasi saya, Natya Lakshita Ardhananeswari Riyanto. Terima kasih telah menjadi sosok yang selalu ada,

bahkan tanpa perlu banyak bicara. Terima kasih telah menjadi bagian dari perjalanan ini, menjadi penguat ketika saya mulai ragu, dan menjadi pengingat bahwa saya tidak pernah berjalan sendirian.

6. Almarhum ayahanda tersayang, nahkoda yang telah lebih dulu menepi ke keabadian sebelum sempat melihat kapal ini bersandar di pelabuhan. Meski raga tak lahi hadir, namun jejak, nilai, dan harapan Ayah tetap hidup dalam setiap langkah yang saya tempuh hingga hari ini. Saya percaya, di suatu tempat yang tidak terlihat, Ayah tetap menyaksikan perjalanan ini. Karya ini adalah salah satu cara saya menyampaikan bahwa saya terus berusaha menjadi seseorang yang bisa Ayah banggakan, meskipun tanpa kehadiran Ayah di sisi saya. Dan semoga, Tuhan menempatkanmu di tempat terbaik, sebagaimana engkau pernah menjadi tempat terbaik bagiku.
7. Nadzira Zahrawani Alhaq yang selalu membersamai saya. Terima kasih karena telah memilih untuk tetap ada, bahkan di tengah proses yang tidak selalu mudah untuk dijalani. Di saat segalanya terasa rumit dan melelahkan, kehadiranmu menjadi ruang tenang, namun selalu cukup untuk membuat saya bertahan. Terima kasih untuk kesabaran, pengertian, dan cara sederhana dalam menghibur, menguatkan, dan menemani yang mungkin tidak selalu terlihat besar, tetapi sangat berarti. Kamu hadir bukan hanya sebagai teman berbagi cerita, tetapi juga sebagai alasan kecil yang membuat saya terus melangkah, meskipun pelan.
8. Sahabat-sahabat saya. Terima kasih karena telah hadir, bukan hanya di saat segala terasa ringan dan penuh tawa, tetapi juga di momen-momen ketika lelah datang tanpa banyak kata. Dari percakapan kecil hingga dukungan yang tidak terlihat, semuanya menjadi bagian yang ikut mengantarkan saya sampai di titik ini.
9. Dan terakhir, untuk diri saya sendiri. Terima kasih sudah bertahan sejauh ini. Untuk semua hari yang tidak mudah, untuk semua rasa ragu yang berhasil dilewati, dan untuk semua langkah kecil yang akhirnya membawa saya sampai di titik ini. Ini bukan akhir, hanya bukti bahwa saya pernah berjuang.

Yogyakarta, 23 April 2026



(Raditya Hafidh Erlangga Riyanto)

SARI

Permasalahan penentuan rute pengiriman secara formal dimodelkan sebagai Vehicle Routing Problem (VRP), yang tergolong dalam kelas masalah *NP-Hard* sehingga tidak dapat diselesaikan secara deterministik dalam waktu polinomial. Penelitian ini mengusulkan penerapan Squirrel Search Algorithm (SSA), sebuah algoritma metaheuristik berbasis *swarm intelligence* yang terinspirasi dari perilaku adaptif tupai terbang dalam mencari makanan, untuk mengoptimasi rute perjalanan kurir.

SSA diimplementasikan dalam prototipe aplikasi berbasis web menggunakan Python Flask sebagai backend dan HTML/CSS/JavaScript sebagai frontend, dengan visualisasi rute pada peta interaktif berbasis Leaflet.js. Pengujian dilakukan melalui 50 eksekusi pada lima skenario berbeda ($n = 5, 10, 15, 20, 25$ titik pengiriman) dengan masing-masing 10 replikasi menggunakan koordinat simulasi area perkotaan Yogyakarta.

Hasil eksperimen menunjukkan bahwa SSA menghasilkan total jarak tempuh sebesar 10,7489 km ($n=5$) hingga 75,7020 km ($n=25$), dengan waktu komputasi 69,10 ms hingga 226,05 ms pada 50 eksekusi. Pada skenario kecil ($n=5$), SSA menemukan solusi optimal secara konsisten dengan standar deviasi 0,0000 km. Perilaku konvergensi bersifat monoton non-increasing pada semua skenario, dengan persentase penurunan fitness mencapai 34,5% pada Skenario D. Penelitian ini membuktikan bahwa SSA dapat diimplementasikan pada sistem pengiriman kurir berbasis web dan menghasilkan solusi yang valid pada seluruh eksekusi, namun variabilitas solusi meningkat pada skala besar yang menjadi peluang pengembangan lanjutan.

Kata Kunci: *Vehicle Routing Problem, Squirrel Search Algorithm, Metaheuristik, Optimasi Rute, Python Flask, Haversine.*

GLOSARIUM

Ant Colony Optimization	algoritma metaheuristik berbasis perilaku koloni semut dalam menemukan jalur terpendek menuju sumber makanan.
Algoritma	sekumpulan langkah yang terdefinisi dengan jelas untuk menyelesaikan suatu permasalahan atau melakukan komputasi.
Backend	bagian sistem perangkat lunak yang berjalan di sisi server; dalam penelitian ini diimplementasikan menggunakan Python Flask.
Benchmark	dataset atau permasalahan standar yang digunakan sebagai acuan untuk membandingkan performa algoritma secara objektif dan dapat direplikasi, contohnya TSPLIB dan Solomon VRP instances.
Client-server	arsitektur sistem dua lapis digunakan, dengan klien (browser) mengirimkan permintaan kepada server yang memproses dan mengembalikan respons, yang digunakan sebagai arsitektur dasar prototype dalam penelitian ini.
Depot	titik awal dan akhir perjalanan kendaraan dalam permasalahan VRP.
Diversifikasi	strategi dalam algoritma metaheuristik untuk menjelajahi bagian-bagian baru dari ruang solusi yang belum pernah dikunjungi, guna menghindari konvergensi dini ke solusi lokal yang tidak optimal.
Endpoint	URL spesifik pada REST API yang menerima permintaan HTTP dari klien.
Fitness	nilai numerik yang mengukur kualitas suatu solusi dalam algoritma optimasi.
Flask	micro-framework berbasis Python yang digunakan untuk membangun aplikasi web dan REST API secara ringan dan fleksibel.
Frontend	bagian sistem perangkat lunak yang berjalan di sisi klien (browser), menangani tampilan antarmuka pengguna.
Genetic Algorithm	algoritma metaheuristik berbasis evolusi yang meniru mekanisme seleksi alam, persilangan (crossover), dan mutasi genetik untuk mencari solusi optimal.
Gliding	mekanisme pergerakan tupai dalam SSA yang meniru perilaku meluncur tupai terbang dari satu pohon ke pohon lain, direpresentasikan secara matematis melalui konstanta gc (gliding constant) bernilai 1,9.

Haversine	formula matematis untuk menghitung jarak terpendek (great-circle distance) antara dua titik di permukaan bumi berdasarkan koordinat geografis (latitude dan longitude) dengan memperhitungkan kelengkungan bumi; digunakan sebagai fungsi jarak dalam penelitian ini.
HTTP (HyperText Transfer Protocol)	protokol komunikasi yang digunakan untuk pertukaran data antara klien (browser) dan server dalam arsitektur web; menjadi dasar komunikasi antara frontend dan backend dalam prototype penelitian ini.
JSON (JavaScript Object Notation)	format pertukaran data berbasis teks yang ringan dan mudah dibaca oleh manusia maupun mesin; digunakan sebagai format respons API dalam prototype penelitian ini untuk menyampaikan hasil optimasi rute.
Konvergensi	kondisi ini terjadi ketika nilai fitness terbaik dalam populasi tidak lagi mengalami perbaikan yang signifikan dari iterasi ke iterasi, menandakan bahwa algoritma mendekati atau telah menemukan solusi optimal.
Latitude	koordinat geografis yang menyatakan posisi suatu titik di permukaan bumi dalam arah utara–selatan, dinyatakan dalam derajat dengan rentang -90° (kutub selatan) hingga $+90^{\circ}$ (kutub utara)..
Leaflet.js	pustaka JavaScript sumber terbuka (<i>open-source</i>) yang digunakan untuk menampilkan peta interaktif berbasis web; digunakan dalam prototype untuk memvisualisasikan rute pengiriman hasil optimasi SSA di atas peta.
Longitude	koordinat geografis yang menyatakan posisi suatu titik di permukaan bumi dalam arah timur–barat, dinyatakan dalam derajat dengan rentang -180° hingga $+180^{\circ}$..
Metaheuristik	pendekatan algoritmik tingkat tinggi yang digunakan untuk mencari solusi mendekati optimal pada permasalahan kompleks (<i>NP-Hard</i>) secara efisien, tanpa menjamin ditemukannya solusi terbaik secara absolut; contohnya SSA, GA, dan PSO.
NP-Hard	kelas permasalahan komputasi yang tidak memiliki algoritma deterministik dengan waktu penyelesaian polinomial; VRP termasuk

dalam kelas ini, sehingga solusi optimalnya tidak dapat dihitung secara eksak untuk skala besar dalam waktu yang wajar.

Pdp (Predator Presence Probability) parameter SSA yang merepresentasikan probabilitas kehadiran predator; ketika bilangan acak lebih kecil dari Pdp (ditetapkan 0,1), tupai melakukan reinisialisasi acak posisinya sebagai mekanisme diversifikasi.

Premature Convergence kondisi algoritma berhenti berkembang terlalu dini dan terjebak pada solusi lokal yang bukan merupakan solusi terbaik global; diatasi dalam SSA melalui mekanisme SMC dan Pdp.

Proof-of-concept proses pengujian skala kecil untuk memvalidasi apakah sebuah ide atau konsep bisnis dapat diwujudkan dan berfungsi di dunia nyata.

PSO (Particle Swarm Optimization) algoritma metaheuristik berbasis *swarm intelligence* yang terinspirasi dari perilaku kawanan burung atau ikan, setiap partikel bergerak berdasarkan kecepatan dan pengalaman terbaik individu maupun kelompok.

REST API (Representational State Transfer Application Programming Interface) antarmuka pemrograman berbasis arsitektur web yang menggunakan protokol HTTP untuk komunikasi data antar system, diimplementasikan menggunakan Flask sebagai jembatan antara frontend dan modul SSA.

SMC (Seasonal Monitoring Condition) mekanisme dalam SSA untuk mendeteksi kondisi stagnasi.

SPV (Smallest Position Value) metode dekoding yang mengubah vektor bilangan riil kontinu (representasi individu SSA) menjadi urutan permutasi diskret.

SSA (Squirrel Search Algorithm) algoritma metaheuristik berbasis *swarm intelligence* yang dikembangkan oleh Mohit Jain (2019), terinspirasi dari perilaku adaptif tupai terbang dalam mencari makanan melalui mekanisme *gliding* antar pohon.

Standar Deviasi ukuran statistik yang menggambarkan seberapa besar penyebaran atau variabilitas nilai hasil eksekusi dari nilai rata-ratanya.

Swarm Intelligence paradigma komputasi yang terinspirasi dari perilaku kolektif makhluk hidup (kawanan, koloni, atau kelompok) untuk memecahkan masalah optimasi.

TSP (Travelling Salesman Problem) permasalahan optimasi klasik untuk mencari rute terpendek yang mengunjungi sejumlah kota tepat satu kali dan kembali ke titik awal.

VRP (Vehicle Routing Problem) permasalahan optimasi kombinatorial yang bertujuan menentukan rute optimal bagi satu atau beberapa kendaraan untuk melayani sejumlah pelanggan dari satu depot, dengan meminimalkan total jarak atau biaya.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	ix
GLOSARIUM	x
DAFTAR ISI	xiv
DAFTAR TABEL	xvi
DAFTAR GAMBAR	xvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan Penelitian	3
1.4 Manfaat Penelitian	4
1.5 Batasan Penelitian	4
1.6 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA	6
2.1 Landasan Teori	6
2.1.1 Permasalahan Vehicle Routing Problem	6
2.1.2 Algoritma <i>Meta-research</i> gaptik dan Posisi SSA	6
2.1.3 Squirrel Search Algorithm (SSA)	7
2.1.4 Representasi VRP dengan SSA	10
2.1.5 Formula <i>Haversine</i>	11
2.2 Penelitian Terdahulu	12
2.3 Kerangka Pemikiran Penelitian	13
BAB III METODOLOGI PENELITIAN	15
3.1 Jenis Penelitian	15
3.2 Tahapan Penelitian	15
3.2.1 Identifikasi Masalah dan Studi Literatur	16
3.2.2 Perancangan Sistem	16
3.2.3 Implementasi Algoritma	16
3.2.4 Pengumpulan Data	16
3.2.5 Eksperimen dan Pengujian	17
3.2.6 Analisis Hasil dan Pelaporan	17
3.3 Variabel Penelitian	17
3.4 Desain Sistem Prototipe	19
3.4.1 Arsitektur Sistem	19
3.4.2 Flowchart Sistem Keseluruhan	23
3.5 Pseudocode Algoritma SSA	24
3.6 Data Koordinat Lokasi Per Skenario	25
3.7 Parameter Algoritma SSA	31
3.8 Data Penelitian	33
3.8.1 Sumber dan Metode Pengumpulan Data	33
3.8.2 Skenario Pengujian	34

3.9	Teknik Analisis Data.....	35
	BAB IV HASIL DAN PEMBAHASAN	36
4.1	Deskripsi Lingkungan Pengujian	36
4.2	Hasil Pengujian	37
	4.2.1 Statistik Total Jarak Tempuh.....	37
	4.2.2 Analisis Perilaku Konvergensi	38
	4.2.3 Detail Rute Terbaik Per Skenario.....	40
4.3	Pembahasan.....	48
	4.3.1 Analisis Pemenuhan Aspek Evaluasi Penelitian	48
	4.3.2 Analisis Skalabilitas SSA terhadap Pertambahan Titik Pengiriman	48
	4.3.3 Analisis Perilaku Konvergensi	49
	4.3.4 Keterbatasan Implementasi dan Rekomendasi Pengembangan	50
	4.3.5 Relasi Rute Optimal dan Jarak Tempuh dalam Konteks SSA	51
4.4	Ringkasan Temuan.....	51
	BAB V KESIMPULAN DAN SARAN.....	52
5.1	Kesimpulan	52
5.2	Saran.....	53
	DAFTAR PUSTAKA	55
	LAMPIRAN	58

DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu	12
Tabel 3. 1 Definisi Variabel Penelitian.....	17
Tabel 3. 2 Pseudocode SSA untuk VRP	24
Tabel 3. 3 Koordinat Input Lokasi Per Skenario Pengujian	25
Tabel 3. 4 Parameter Algoritma SSA	32
Tabel 3. 5 Skenario Pengujian	34
Tabel 4. 1 Statistik Deskriptif Hasil Pengujian SSA (50) Eksekusi Total.....	37
Tabel 4. 2 Nilai Fitness Rata-Rata per Iterasi (Rata-Rata 10 Replikasi)	38
Tabel 4. 3 Detail Rute Terbaik Skenario A (n=5, Total: 10,7489 km)	40
Tabel 4. 4 Detail Rute Terbaik Skenario B (n=10, Total: 18,2413 km)	41
Tabel 4. 5 Detail Rute Terbaik Skenario C (n=15, Total: 30,9568 km)	42
Tabel 4. 6 Detail Rute Terbaik Skenario D (n=20, Total: 47,4087 km)	43
Tabel 4. 7 Detail Rute Terbaik Skenario E (n=25, Total: 65,9394 km)	45

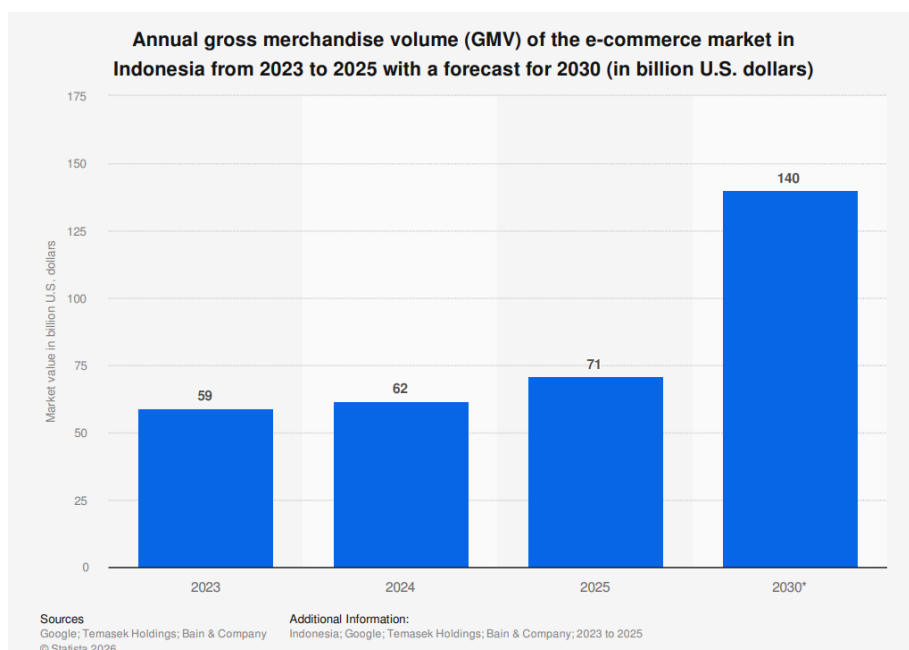
DAFTAR GAMBAR

Gambar 1. 1 Proyeksi GMV <i>E-Commerce</i> Indonesia Tahun 2023-2030	1
Gambar 2. 1 Ilustrasi Mekanisme Pergerakan Tupai dalam SSA.....	9
Gambar 2. 2 Contoh <i>Haversine</i>	11
Gambar 2. 3 Kerangka Penelitian	14
Gambar 3. 1 Alur Tahapan Penelitian.....	16
Gambar 3. 2 Arsitektur Sistem Prototipe.....	19
Gambar 3. 3 Halaman Input Koordinat.....	20
Gambar 3. 4 Proses Running	20
Gambar 3. 5 Hasil Batch.....	21
Gambar 3. 6 Hasil Rute di Peta.....	22
Gambar 3. 7 Hasil Eksekusi Algoritma pada Skenario A.....	22
Gambar 3. 8 Flowchart Sistem Prototipe.....	23
Gambar 3. 9 Input Lokasi Skenario A	27
Gambar 3. 10 Input Lokasi Skenario B	28
Gambar 3. 11 Input Lokasi Skenario C	29
Gambar 3. 12 Input Lokasi Skenario D	30
Gambar 3. 13 Input Lokasi Skenario E.....	31
Gambar 4. 1 Kurva Konvergensi pada Seluruh Skenario	39

BAB I PENDAHULUAN

1.1 Latar Belakang

Perkembangan pesat dalam industri *e-commerce* telah menciptakan lonjakan kebutuhan terhadap sistem logistik dan pengiriman barang yang efisien (Wijaya, 2023), terutama di kawasan urban yang padat dan dinamis. Berdasarkan data dari Statista pada Gambar 1. 1 Proyeksi GMV *E-Commerce* Indonesia Tahun 2023-2030 menunjukkan nilai pasar *e-commerce* Indonesia diproyeksikan meningkat dari USD 30 miliar pada tahun 2020 menjadi lebih dari USD 82 miliar pada 2025. Pertumbuhan ini memberikan tekanan besar terhadap operasional perusahaan pengiriman barang untuk memenuhi ekspektasi pelanggan terhadap pengiriman yang cepat, tepat waktu, dan hemat biaya. Dalam praktiknya, efisiensi pengiriman sangat ditentukan oleh kemampuan sistem untuk merancang rute distribusi yang optimal.



Gambar 1. 1 Proyeksi GMV *E-Commerce* Indonesia Tahun 2023-2030

Sumber : Statista (2024), berdasarkan Google, Temasek, & Bain & Company

Permasalahan utama yang dihadapi oleh perusahaan kurir adalah bagaimana menentukan rute perjalanan terbaik dalam waktu yang terbatas dan dengan keterbatasan sumber daya yang terbatas pula. Permasalahan ini dikenal sebagai Vehicle Routing Problem (VRP) (Ferdiansyah et al., 2021), yaitu bentuk permasalahan optimasi kombinatorial yang tergolong *NP-Hard* (Juan et al., 2023). Sebagai ilustrasi, penyelesaian rute secara manual mengharuskan pemeriksaan (n-

1)! kemungkinan permutasi. Pada $n=10$ saja terdapat 362.880 kombinasi; pada $n=25$ angkanya mencapai $24! \approx 6,2 \times 10^{23}$ jauh melampaui kapasitas evaluasi manual. Inilah yang menjadikan pendekatan algoritmik sangat relevan. Dalam konteks perkotaan, VRP menjadi semakin kompleks karena harus mempertimbangkan berbagai faktor seperti jumlah kendaraan terbatas, waktu layanan pelanggan, kondisi lalu lintas yang fluktuatif, serta permintaan yang dinamis. Solusi terhadap VRP yang konvensional sering kali tidak mampu menjawab kompleksitas ini secara efisien dan real-time (Mahmudy et al., 2024). Penelitian ini secara khusus mengimplementasikan VRP dengan satu kendaraan (*single-vehicle* VRP, $K=1$), yang ekuivalen dengan Travelling Salesman Problem (TSP). Dalam kasus ini, satu kurir melayani seluruh titik pengiriman dalam satu rute tertutup yang dimulai dan diakhiri di depot.

Berbagai penelitian telah mencoba mengatasi permasalahan VRP dengan algoritma heuristik dan metaheuristik. Misalnya, (Fatma et al., 2022) menerapkan model VRPTW (*VRP with Time Windows*) pada pengangkut limbah medis dengan mempertimbangkan keterbatasan waktu operasional, sedangkan (Jackye Jackson James & Andung Jati Nugroho, 2024) mengadopsi metode *Nearest Neighbor* untuk distribusi gas LPG, yang menunjukkan perbaikan waktu dan efisiensi rute walaupun masih bersifat sederhana. Penelitian (Gunawan, 2024) menggunakan *Genetic Algorithm* adaptif dan berhasil meningkatkan performa solusi, namun tetap menghadapi kendala karena algoritma terlalu cepat berhenti pada solusi yang belum tentu terbaik, serta sangat bergantung pada pengaturan awal parameter yang sensitif terhadap perubahan kecil.

Sementara itu, Squirrel Search Algorithm (SSA) muncul sebagai pendekatan metaheuristik baru yang meniru perilaku adaptif tupai dalam pencarian makanan. Algoritma ini memiliki kemampuan eksplorasi dan eksploitasi solusi yang seimbang, sehingga mampu menjangkau ruang solusi secara luas sekaligus menyempurnakan solusi yang telah ditemukan. Meskipun masih tergolong baru, SSA telah menunjukkan performa yang kompetitif dalam berbagai permasalahan optimasi teknis. SSA terbukti efektif dalam permasalahan optimasi seperti portofolio saham (Dhaini & Mansour, 2021), namun penerapannya dalam domain logistik, khususnya untuk VRP, masih belum banyak dieksplorasi. Padahal, sifat adaptif dan fleksibel dari SSA memberikan potensi besar untuk mengatasi permasalahan rute yang kompleks dan dinamis sebagaimana yang umum ditemukan dalam sistem distribusi pengiriman di dunia nyata.

Lebih lanjut, beberapa penelitian lokal seperti yang dilakukan oleh (Ferdiansyah et al., 2021) telah membuktikan potensi VRP dalam efisiensi distribusi barang di Indonesia, dan

(Malik et al., 2023) mengembangkan sistem pencarian rute kurir berbasis algoritma *Simple Hill Climbing*. Namun, penelitian – penelitian tersebut masih terbatas pada algoritma yang bersifat statis atau belum diuji dengan data simulasi yang mencerminkan kondisi perkotaan secara realistis. Belum terdapat studi yang secara khusus mengintegrasikan SSA dalam simulasi rute kurir perkotaan, dengan pengujian yang sistematis dan perbandingan terhadap algoritma lainnya.

Hal ini mengindikasikan adanya kesenjangan penelitian dalam pengembangan sistem optimasi rute berbasis SSA yang terintegrasi dengan data aktual urban, khususnya dalam konteks operasional kurir. Oleh karena itu, penelitian ini diarahkan untuk mengisi kesenjangan tersebut dengan membangun model kuantitatif berbasis SSA yang mampu menangani kompleksitas VRP secara realistis, dan efisien.

Oleh karena itu, penelitian ini bertujuan untuk merancang bangun sebuah aplikasi pencarian rute kurir menggunakan Squirrel Search Algorithm. Penelitian ini akan mengintegrasikan algoritma tersebut ke dalam sistem berbasis web dan menggunakan model jarak *Haversine* untuk mensimulasikan kondisi pengiriman yang efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam penelitian ini adalah:

- a. Bagaimana penerapan Squirrel Search Algorithm dapat diimplementasikan untuk pencarian rute optimal pada sistem pengiriman kurir?
- b. Seberapa efektif SSA dalam menyelesaikan VRP ditinjau dari total jarak tempuh, waktu eksekusi, dan stabilitas konvergensi solusi pada berbagai skala jumlah titik pengiriman?
- c. Apa keterbatasan dan tantangan penerapan SSA dalam menyelesaikan VRP pada data simulasi koordinat perkotaan, dan apa rekomendasi pengembangannya?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dibuat, berikut merupakan tujuan penelitian yang dilakukan :

- a. Mengimplementasikan Squirrel Search Algorithm (SSA) dalam menyelesaikan permasalahan Vehicle Routing Problem (VRP) pada sistem pengiriman kurir berbasis data simulasi.
- b. Mengevaluasi performa SSA dalam menentukan rute pengiriman optimal, ditinjau dari total jarak tempuh, waktu eksekusi, dan stabilitas konvergensi solusi.

- c. Mengidentifikasi tantangan dan keterbatasan yang dihadapi SSA dalam penerapannya pada kasus rute yang dinamis dan kompleks, serta memberikan rekomendasi untuk pengembangan algoritma selanjutnya.

1.4 Manfaat Penelitian

Berikut merupakan manfaat yang diperoleh dari penelitian yang telah dilakukan:

- a. Memberikan solusi alternatif dalam optimasi rute pengiriman untuk kurir dengan waktu tempuh dan jarak yang optimal.
- b. Memberikan kontribusi metodologis dengan mengadaptasi Squirrel Search Algorithm dalam domain logistik, khususnya pengiriman berbasis *benchmark* TSP/VRP.

1.5 Batasan Penelitian

Berikut merupakan batasan penelitian yang digunakan untuk membatasi ruang lingkup penelitian yang dilakukan:

- a. Koordinat lokasi input berupa koordinat geografis (*latitude* dan *longitude*) yang diolah secara offline atau melalui input manual.
- b. Data yang digunakan adalah koordinat geografis lokasi nyata di kawasan perkotaan Yogyakarta yang dipilih secara manual melalui platform Google Maps, bukan data operasional riil dari perusahaan logistik. Meskipun koordinat yang digunakan merepresentasikan lokasi aktual, pemilihan titiknya dilakukan secara mandiri oleh peneliti sehingga tidak mencerminkan distribusi pengiriman yang sesungguhnya di lapangan.
- c. Jumlah kendaraan ditetapkan $K=1$ (*single-vehicle*), sehingga model yang digunakan ekuivalen dengan Travelling Salesman Problem (TSP). Dalam kasus ini, satu kurir melayani semua titik pengiriman dalam satu rute. Penelitian ini tidak mempertimbangkan variasi kapasitas kendaraan, pembagian rute antar beberapa kurir, maupu batasan waktu kerja kurir.
- d. Permintaan pelanggan dianggap konstan dan diketahui sebelumnya, sehingga tidak mempertimbangkan ketidakpastian atau permintaan yang berubah secara real-time.
- e. Tidak mempertimbangkan faktor eksternal seperti kondisi lalu lintas.
- f. Algoritma Squirrel Search Algorithm yang digunakan merupakan versi dasar, tanpa modifikasi hybrid dengan algoritma lain.

- g. Evaluasi performa algoritma terbatas pada metrik kuantitatif seperti total jarak tempuh, waktu eksekusi, dan konvergensi solusi, tanpa mempertimbangkan aspek biaya atau kepuasan pelanggan.

1.6 Sistematika Penulisan

Sistematika penulisan ini dibagi menjadi lima bab yang saling berkaitan satu sama lain. Bab I Pendahuluan berisi penjelasan mengenai latar belakang, permasalahan, tujuan penulisan, ruang lingkup, manfaat penelitian, serta sistematika penulisan yang digunakan. Selanjutnya, Bab II Tinjauan Pustaka membahas Squirrel Search Algorithm, Vehicle Routing Problem, Travelling Salesman Problem, serta penelitian terkait. Bab III Metodologi Penelitian menjelaskan mengenai analisis cara kerja Squirrel Search *Algorithm* dalam aplikasi pencarian rute terdekat pada kurir, serta gambaran umum pembuatan sistem. Bab IV Hasil dan Pembahasan memaparkan data yang diperoleh berdasarkan metode yang digunakan, hasil implementasi Squirrel Search Algorithm pada *dataset benchmark*, visualisasi rute optimal, serta evaluasi performa berdasarkan jarak tempuh dan waktu eksekusi. Terakhir, Bab V Kesimpulan dan Saran memuat kesimpulan dari hasil penelitian serta saran pengembangan lanjutan untuk penerapan Squirrel Search Algorithm dalam lingkungan data nyata atau pengembangan algoritma *hybrid*.

BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 Permasalahan Vehicle Routing Problem

Secara formal, VRP dapat dirumuskan sebagai berikut. Diberikan graf $G = (V, E)$; himpunan simpul $V = \{v_0, v_1, \dots, v_n\}$ adalah himpunan simpul yang terdiri dari depot (v_0) dan n pelanggan, serta E adalah himpunan busur dengan bobot c_{ij} yang merepresentasikan jarak antar simpul (MMCS Team, LMAID Laboratory, ENSMR-Rabat et al., 2024). Tujuannya adalah mencari himpunan rute kendaraan K sehingga:

(1) Setiap pelanggan dikunjungi tepat satu kali; (2) Setiap rute dimulai dan diakhiri di depot; (3) Total jarak tempuh seluruh rute diminimalkan (MMCS Team, LMAID Laboratory, ENSMR-Rabat et al., 2024; Stamadianos et al., 2024).

VRP tergolong dalam kelas *NP-Hard*, yang berarti tidak ada algoritma deterministik yang dapat menyelesaikannya dalam waktu polinomial untuk semua kasus (Juan et al., 2023). Kompleksitas ini mendorong penggunaan pendekatan heuristik dan metaheuristik dalam pencarian solusi yang mendekati optimal (MMCS Team, LMAID Laboratory, ENSMR-Rabat et al., 2024).

2.1.2 Algoritma *Meta-research* gaptik dan Posisi SSA

Metaheuristik merupakan pendekatan algoritmik tingkat tinggi yang digunakan untuk menemukan solusi yang mendekati optimal pada permasalahan optimasi yang kompleks, khususnya yang termasuk dalam kategori *NP-Hard*. Pada kategori ini, penyelesaian optimal secara eksak membutuhkan waktu komputasi yang sangat besar (Stamadianos et al., 2024). Berbeda dengan algoritma eksak yang menjamin solusi optimal tetapi kurang efisien untuk masalah berskala besar, metaheuristik mengandalkan strategi pencarian yang adaptif dan fleksibel. Pendekatan ini memungkinkan eksplorasi ruang solusi yang luas secara lebih efisien, meskipun tidak selalu menjamin solusi terbaik secara absolut, namun cukup optimal dalam waktu yang realistis.

Secara umum, algoritma metaheuristik dapat diklasifikasikan berdasarkan sumber inspirasinya. Pertama, algoritma berbasis evolusi (*evolutionary algorithms*), yang meniru mekanisme seleksi alam, seperti Genetic Algorithm (GA), Differential Evolution (DE), dan Evolution Strategy (ES). Kedua, algoritma berbasis kecerdasan kolektif (*swarm intelligence*),

yang terinspirasi dari perilaku sosial makhluk hidup, seperti Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), dan Squirrel Search Algorithm (SSA). Ketiga, algoritma berbasis fenomena fisika, yang meniru prinsip-prinsip alam seperti energi dan gaya, contohnya Simulated Annealing (SA) dan Gravitational Search Algorithm (GSA).

SSA termasuk dalam kategori *swarm intelligence* karena menggunakan sekumpulan agen (tupai) yang bekerja secara kolektif dalam mencari solusi. Setiap agen berinteraksi dengan agen lain serta lingkungannya untuk menemukan posisi terbaik. Berbeda dengan GA yang menggunakan proses reproduksi seperti *crossover* dan mutasi, serta PSO yang mengandalkan kecepatan dan posisi partikel, SSA memanfaatkan mekanisme *gliding*, yaitu pergerakan tupai dari satu pohon ke pohon lain berdasarkan ketersediaan makanan dan ancaman predator. Mekanisme ini meniru perilaku alami tupai terbang dalam mencari sumber makanan secara efisien.

Dalam konteks permasalahan Vehicle Routing Problem (VRP), SSA memiliki beberapa keunggulan dibandingkan algoritma lain seperti GA dan PSO. Pertama, jumlah parameter pada SSA relatif lebih sedikit dan lebih sederhana, seperti konstanta *gliding* (gc), probabilitas predator (Pdp), jumlah pohon acorn (na), dan ambang batas seasonal monitoring condition (ϵ), sehingga lebih mudah dalam proses penyesuaian parameter. Kedua, SSA memiliki mekanisme Seasonal Monitoring Condition (SMC) yang mampu mendeteksi kondisi stagnasi dalam proses konvergensi. Ketika tidak terjadi perbaikan solusi dalam beberapa iterasi, algoritma akan melakukan inisialisasi ulang sebagian populasi, sehingga dapat menghindari terjebak pada solusi lokal (*premature convergence*). Ketiga, SSA terbukti memiliki performa yang kompetitif untuk masalah optimasi kombinatorial dengan skala kecil hingga menengah (Dhaini & Mansour, 2021), yang sesuai dengan karakteristik permasalahan VRP pada distribusi kurir dalam penelitian ini.

2.1.3 Squirrel Search Algorithm (SSA)

Squirrel Search Algorithm (SSA) merupakan salah satu algoritma metaheuristik yang baru dikembangkan dan tergolong dalam keluarga *swarm intelligence*. SSA didasarkan pada perilaku adaptif tupai terbang dalam mencari makanan, mereka berpindah antar pohon berdasarkan musim dan kondisi lingkungan. Konsep ini kemudian diadopsi dalam komputasi untuk menciptakan mekanisme pencarian solusi yang mampu menyeimbangkan eksplorasi dan eksploitasi.

Eksplorasi dalam SSA dilakukan dengan menempatkan solusi awal secara acak dalam ruang solusi dan membiarkan tupai-tupai tersebut mencari makanan di pohon yang berbeda

dengan probabilitas tertentu. Sementara eksploitasi terjadi saat musim berubah atau ketika terdapat ancaman predator, yang memaksa tupai berpindah ke pohon yang lebih baik berdasarkan pengalaman sebelumnya. Mekanisme ini memungkinkan SSA untuk menghindari jebakan solusi lokal dan mempertahankan pencarian pada solusi global yang lebih optimal.

(Dhaini & Mansour, 2021) berhasil menerapkan SSA pada permasalahan optimasi portofolio saham dan membuktikan bahwa algoritma ini dapat mengungguli pendekatan lain seperti GA dalam hal kecepatan dan akurasi pencarian solusi. SSA menunjukkan kemampuan adaptif yang tinggi dalam menangani masalah multikriteria dan bersifat non-linier. Hal ini memberikan peluang besar untuk mengaplikasikan SSA ke dalam permasalahan logistik seperti VRP, yang juga bersifat kompleks dan memiliki banyak variabel.

Algoritma ini terinspirasi dari perilaku mencari makan tupai terbang (*Pteromys Volans*) yang secara dinamis berpindah antara pohon hickory (sebagai sumber makanan utama) dan pohon lainnya (sebagai sumber makanan alternatif) melalui mekanisme *gliding* (meluncur). SSA bekerja dengan tiga jenis perpindahan posisi tupai, yaitu:

- Perpindahan dari pohon biasa ke pohon hickory (acorn tree): mempresentasikan eksploitasi lokal menuju sumber makanan terbaik.
- Perpindahan dari pohon biasa ke pohon normal lainnya: mempresentasikan eksplorasi solusi baru di ruang pencarian.
- Perpindahan acak berdasarkan Predator Presence Probability (Pdp): merupakan mekanisme diversifikasi untuk menghindari konvergensi prematur.

Secara konseptual, mekanisme pergerakan tupai dalam SSA dapat diilustrasikan sebagai berikut. Terdapat tiga jenis pohon dalam ekosistem SSA: (1) pohon hickory, yang merepresentasikan solusi terbaik global, (2) pohon acorn, yang merepresentasikan kelompok solusi terbaik berikutnya, dan (3) pohon biasa, yang merepresentasikan sisa individu dalam populasi.

POPULASI SSA – TIGA JENIS POHON:

[Pohon Hickory] = Solusi terbaik global (fitness terkecil)
 [Pohon Acorn × na] = na solusi terbaik berikutnya (elite group)
 [Pohon Biasa] = Sisa populasi ($N - 1 - na$ individu)

TIGA JENIS PERPINDAHAN POSISI TUPAI:

```

(A) EKSPLOITASI – Pohon Acorn → Pohon Hickory
    Tupai di pohon acorn meluncur menuju solusi terbaik global.
    Persamaan:  $X_{baru} = X_i + gc \times Dg \times (X_{hickory} - X_i)$ 
     $gc = 1.9$  (konstanta gliding),  $Dg = \text{jarak meluncur (Haversine)}$ 

(B) EKSPLORASI – Pohon Biasa → Pohon Acorn (jika  $r > Pdp$ )
    Tupai di pohon biasa meluncur ke salah satu pohon acorn secara
    acak.
    Persamaan:  $X_{baru} = X_i + gc \times Dg \times (X_{acorn\_acak} - X_i)$ 
     $Dg \sim \text{Uniform}(0.5, 1.5)$ 

(C) DIVERSIFIKASI – Reinisialisasi Acak (jika  $r \leq Pdp$ )
    Simulasi ancaman predator: tupai berpindah ke posisi acak baru.
    Persamaan:  $X_{baru} = \text{random\_uniform}(0, n\_customers)$ 
    Mencegah premature convergence.

SEASONAL MONITORING CONDITION (SMC):


---


Jika  $|\text{fitness\_sebelumnya} - \text{fitness\_terbaik}| < \varepsilon$  DAN  $t > 10$ :
    → Reinisialisasi 20% individu terburuk secara acak
    → Mekanisme escape dari local optima


---


Keterangan:  $r = \text{bilangan acak Uniform}(0,1)$ ;  $Pdp = 0.1$ ;  $\varepsilon = 10^{-6}$ 

```

Gambar 2. 1 Ilustrasi Mekanisme Pergerakan Tupai dalam SSA

Berdasarkan ilustrasi di atas SSA, SSA mengelola populasinya melalui hierarki tiga level. Level tertinggi (pohon hickory) menjadi target eksploitasi bagi seluruh individu di pohon acorn, sehingga mendorong konvergensi ke solusi terbaik yang ditemukan. Level menengah (pohon acorn) menjadi target eksplorasi bagi individu-individu di pohon biasa, sehingga menjaga keragaman solusi dalam populasi. Level dasar (pohon biasa) memiliki dua perilaku bergantung nilai acak r : jika $r > Pdp$ maka melakukan eksplorasi ke pohon acorn, dan jika $r \leq Pdp$ maka melakukan reinisialisasi acak sebagai respons terhadap ancaman predator. Mekanisme ini secara adaptif menyeimbangkan intensifikasi (eksploitasi) dan diversifikasi (eksplorasi) tanpa perlu parameter adaptif yang kompleks.

Dalam konteks penelitian ini, setiap "tupai" merepresentasikan satu kandidat solusi rute pengiriman berupa vektor bilangan riil berdimensi n (jumlah pelanggan). Vektor tersebut didekodekan menjadi urutan kunjungan diskret menggunakan mekanisme Smallest Position

Value (SPV), kemudian dievaluasi menggunakan Formula Haversine sebagai fungsi fitness. Proses ini diulang selama $T = 100$ iterasi hingga diperoleh rute dengan total jarak tempuh terkecil.

Posisi tupai diperbarui menggunakan persamaan berikut seperti yang diperkenalkan Mohit Jain pada tahun 2019:

$$X_i^{t+1} = X_i^t + gc \times Dg \times (X_{a_t} - X_i^t) \quad (2.1)$$

Dalam persamaan tersebut, X_i^t adalah posisi tupai ke- i pada iterasi t , gc adalah konstanta *gliding* (bernilai 1,9), Dg adalah jarak meluncur (*gliding distance*), dan X_{a_t} adalah posisi pohon acorn terbaik. Mekanisme ini memungkinkan SSA menjaga keseimbangan antara eksplorasi global dan eksploitasi lokal secara adaptif.

Keunggulan SSA meliputi: (1) parameter yang relatif sedikit sehingga mudah dikonfigurasi; (2) konvergensi yang stabil berkat mekanisme *gliding* yang terkontrol; serta (3) kemampuan diverifikasi melalui Pdp yang mencegah solusi terjebak pada lokal optima.

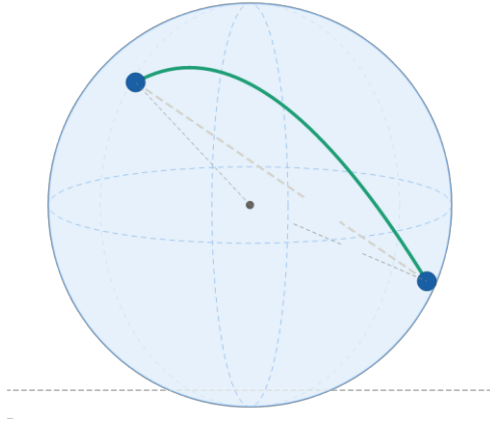
2.1.4 Representasi VRP dengan SSA

Penerapan SSA pada VRP memerlukan adaptasi representasi solusi, mengingat VRP bersifat diskret sementara SSA beroperasi pada ruang kontinu. Pendekatan umum yang digunakan adalah *encoding* permutasi berbasis posisi (*position-based permutation encoding*). Pada pendekatan ini, setiap tupai merepresentasikan urutan kunjungan pelanggan sebagai vektor bilangan riil yang kemudian didekodekan menjadi urutan rute diskret melalui mekanisme Smallest Position Value (SPV). Pendekatan ini telah banyak divalidasi dalam penelitian *swarm intelligence* untuk VRP (Stamadianos et al., 2024).

Dalam konteks penelitian ini, setiap individu (tupai) merepresentasikan satu solusi rute pengiriman, yaitu urutan permutasi pelanggan yang dikunjungi oleh kendaraan. Fungsi objektif yang diminimalkan adalah total jarak geodesik yang dihitung oleh Formula *Haversine*.

2.1.5 Formula *Haversine*

Formula *Haversine* adalah metode perhitungan jarak antara dua titik di permukaan bumi berdasarkan koordinat geografis (*latitude* dan *longitude*) dengan mempertimbangkan kelengkungan bumi. Formula ini menghasilkan jarak garis besar lingkaran (*great-circle distance*) atau jarak geodesic yang merupakan representasi jarak terpendek di permukaan bola (Muttaqin et al., 2024).



Gambar 2. 2 Contoh *Haversine*

Secara matematis, *Haversine Distance* antara dua titik $A(\varphi_1, \lambda_1)$ dan $B(\varphi_2, \lambda_2)$ dihitung sebagai:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos \varphi_1 \times \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (2.2)$$

$$c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a}) \quad (2.3)$$

$$d = R \times c \quad (2.4)$$

Dalam rumus tersebut φ menyatakan *latitude* (radian), λ menyatakan *longitude* (radian), sedangkan $\Delta\varphi$ dan $\Delta\lambda$ masing-masing merupakan selisih *latitude* dan *longitude*. Nilai R menunjukkan radius rata-rata bumi yaitu 6.371 km. Formula *Haversine* dipilih dalam penelitian ini karena komputasinya ringan dan akurat untuk jarak antar lokasi dalam skala distribusi perkotaan tanpa memerlukan data jaringan jalan (*road network*) yang kompleks.

2.2 Penelitian Terdahulu

Bagian ini memaparkan kajian terhadap penelitian-penelitian terdahulu relevan yang telah dilakukan sebelumnya, baik dalam bidang Vehicle Routing Problem, algoritma metaheuristik, maupun penerapan Squirrel Search Algorithm. Kajian ini bertujuan untuk mengidentifikasi posisi penelitian yang sedang dilakukan serta kesenjangan penelitian yang menjadi dasar justifikasi penelitian ini.

Tabel 2. 1 Penelitian Terdahulu

No	Penulis	Judul	Metode	Hasil Utama	Relevansi
1	(Gunawan, 2024)	Optimasi VRP Menggunakan Genetic Algorithm Adaptif untuk Distribusi Barang di Kawasan Perkotaan.	Genetic Algorithm (GA) Adaptif	GA Adaptif meningkatkan kualitas solusi VRP, namun masih memiliki kelemahan stabilitas konvergensi dan sensitivitas parameter.	Menjadi <i>baseline</i> perbandingan metaheuristik; SSA dikaji untuk mengatasi kelemahan konvergensi GA.
2	(Fatma et al., 2022)	Penerapan VRPTW pada Pengangkutan Limbah Medis	VRPTW + Heuristik	Model VRPTW berhasil meminimalkan total jarak dan mematuhi batasan waktu pada rute pengangkutan limbah medis.	Memvalidasi pendekatan VRP dalam domain logistik; penelitian ini mengadopsi konsep meminimalisi jarak.
3	(Jackye Jackson James & Andung Jati Nugroho, 2024)	Penentuan Rute Distribusi Gas LPG dengan Metode Nearest Neighbor	Nearest Neighbor	Nearest Neighbor menghasilkan rute distribusi yang lebih efisien dibanding rute acak, namun masih suboptimal pada jumlah <i>node</i> besar.	Menjadi acuan metode sederhana yang akan dibandingkan secara komparatif dengan SSA.
4	(Dhaini & Mansour, 2021)	Portofolio Optimization Using Squirrel Search Algorithm	Squirrel Search Algorithm (SSA)	SSA terbukti kompetitif dalam optimasi portofolio saham,	Sumber utama validasi kemampuan SSA pada masalah

No	Penulis	Judul	Metode	Hasil Utama	Relevansi
				menunjukkan keseimbangan eksplorasi-eksploitasi yang baik.	optimasi kombinatorial; dasar penerapan SSA ke VRP.
5	(Malik et al., 2023)	Sistem Pencarian Rute Kurir Berbasis Simple Hill Climbing	Simple Hill Climbing	<i>Hill Climbing</i> mampu menemukan solusi rute yang lebih baik dari <i>greedy</i> , namun mudah terjebak pada loktal optima.	Mengidentifikasi kesenjangan penelitian: sistem masih bersifat simulasi statis; penelitian ini mengembangkan prototipe berbasis web dengan SSA.

Berdasarkan tinjauan terhadap penelitian-penelitian di Tabel 2. 1 Penelitian Terdahulu, terdapat beberapa celah penelitian yang menjadi dasar penelitian. Pertama, penelitian yang mengaplikasikan SSA secara khusus pada permasalahan VRP dalam konteks logistik kurir masih sangat terbatas. Kedua, sebagian besar sistem yang dikembangkan masih bersifat simulasi statis tanpa antarmuka aplikasi yang fungsional dan dapat dioperasikan pengguna. Ketiga, belum terdapat studi yang secara sistematis menganalisis perilaku konvergensi SSA dalam konteks pencarian rute distribusi berbasis genetic geodesic. Penelitian ini hadir untuk mengisi ketiga celah tersebut melalui implementasi SSA pada VRP dalam prototipe aplikasi berbasis web dengan evaluasi komprehensif.

2.3 Kerangka Pemikiran Penelitian

Kerangka pemikiran penelitian ini disusun berdasarkan alur logis yang menghubungkan fenomena empiris dengan pendekatan solusi yang diajukan. Gambar 2. 3 Kerangka Penelitian menunjukkan kerangka penelitian secara keseluruhan, yang menggambarkan hubungan antara pertumbuhan *e-commerce*, permasalahan optimasi rute dalam Vehicle Routing Problem, penggunaan Squirrel Search Algorithm sebagai pendekatan solusi, hingga tahap implementasi dan evaluasi kinerja sistem.

Pertumbuhan *e-commerce* yang pesat menghasilkan lonjakan volume pengiriman barang. Perusahaan kurir dihadapkan pada permasalahan penentuan rute optimal yang bersifat *NP-Hard*, sehingga metode konvensional tidak mampu menghasilkan solusi yang berkualitas

dalam waktu komputasi yang wajar. Algoritma metaheuristik yang telah ada, seperti *Genetic Algorithm*, masih memiliki kelemahan dalam hal stabilitas konvergensi.

SSA diajukan sebagai solusi alternatif karena memiliki mekanisme gliding yang menjamin keseimbangan eksplorasi-eksploitasi, parameter yang simple, serta terbukti efektif pada masalah optimasi kombinatorial. SSA diimplementasikan untuk menyelesaikan VRP dengan fungsi objektif minimasi total jarak Haversine.

Algoritma SSA diintegrasikan ke dalam prototipe aplikasi berbasis web. Evaluasi dilakukan terhadap tiga aspek utama: (1) kualitas solusi rute yang dihasilkan diukur melalui total jarak tempuh (km); (2) efisiensi komputasi yang diukur melalui waktu eksekusi (ms); serta (3) karakteristik perilaku konvergensi algoritma melalui analisis kurva konvergensi iterasi.



Gambar 2. 3 Kerangka Penelitian

BAB III

METODOLOGI PENELITIAN

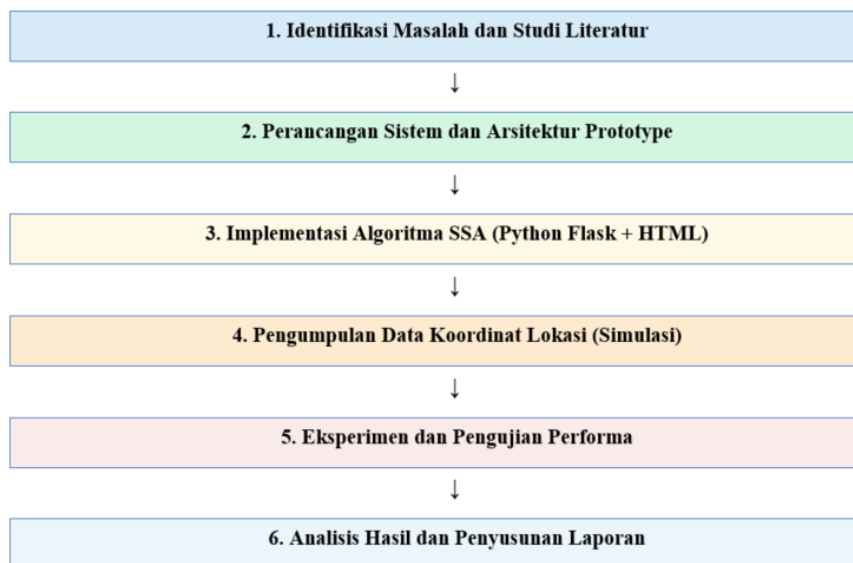
3.1 Jenis Penelitian

Penelitian ini menggunakan pendekatan eksperimen kuantitatif, yaitu pendekatan ilmiah yang mendasarkan proses penelitian pada pengukuran dan pembuktian melalui data numerik yang dapat diverifikasi secara statistik. Penelitian kuantitatif bersifat deduktif dan bertujuan untuk menguji hipotesis yang telah dirumuskan secara objektif (Waruwu, 2023). Pendekatan ini dipilih karena tujuan utama penelitian adalah mengevaluasi performa Squirrel Search Algorithm (SSA) berdasarkan parameter terukur, yaitu total jarak tempuh rute (km), waktu komputasi (milidetik), serta perilaku konvergensi algoritma secara empiris.

Strategi penelitian yang diterapkan adalah pengembangan prototipe (*prototyping*). Melalui pendekatan ini, sebuah aplikasi berbasis web dibangun sebagai instrumen simulasi dan pengujian algoritma. Metode *prototyping* memungkinkan pengguna mendapatkan gambaran awal terhadap sistem yang dikembangkan dan memungkinkan pengujian awal sebelum sistem dirilis secara penuh (Puspita et al., 2024). Prototipe ini tidak dimaksudkan sebagai produk navigasi komersial, melainkan sebagai media eksperimen terkontrol yang memungkinkan pengujian SSA pada berbagai skenario input secara sistematis dan berulang.

3.2 Tahapan Penelitian

Penelitian ini dilaksanakan melalui enam tahapan sistematis yang saling berkesinambungan sebagaimana ditunjukkan pada Gambar 3. 1 Alur Tahapan Penelitian



Gambar 3. 1 Alur Tahapan Penelitian

3.2.1 Identifikasi Masalah dan Studi Literatur

Tahap pertama mencakup identifikasi permasalahan melalui kajian literatur terhadap jurnal-jurnal ilmiah terkait Vehicle Routing Problem, algoritma metaheuristik, dan Squirrel Search Algorithm. *Output* tahap ini adalah rumusan masalah, tujuan penelitian, serta pemahaman mendalam terhadap mekanisme SSA sebagai dasar implementasi.

3.2.2 Perancangan Sistem

Tahap kedua adalah merancang arsitektur sistem prototipe berbasis web sebagai media eksperimen. Perancangan mencakup desain arsitektur aplikasi *client-server*, struktur modul algoritma SSA, rancangan antarmuka pengguna, serta rancangan skema pengujian yang didokumentasikan dalam *flowchart* dan spesifikasi teknis.

3.2.3 Implementasi Algoritma

Tahap ketiga adalah mengimplementasikan algoritma SSA ke dalam aplikasi web menggunakan Python Flask sebagai backend dan HTML/CSS/JavaScript sebagai frontend. Implementasi mencakup modul inisialisasi populasi, perhitungan jarak *Haversine*, evaluasi fungsi objektif, mekanisme *gliding* SSA, serta visualisasi rute pada peta berbasis Leaflet.js.

3.2.4 Pengumpulan Data

Tahap keempat adalah pengumpulan data koordinat lokasi pengiriman dengan cara memilih secara manual titik-titik lokasi nyata di kawasan perkotaan Yogyakarta melalui platform Google Maps, kemudian mencatat koordinatnya dalam format *latitude* dan *longitude*.

Pendekatan ini menghasilkan data koordinat yang merepresentasikan lokasi-lokasi aktual, meskipun pemilihan titiknya dilakukan secara mandiri oleh peneliti dan bukan diperoleh dari catatan pengiriman operasional perusahaan. Variasi jumlah titik pengiriman digunakan untuk menguji skalabilitas algoritma pada berbagai skala permasalahan.

3.2.5 Eksperimen dan Pengujian

Tahap kelima adalah pelaksanaan eksperimen terstruktur dengan menjalankan SSA pada lima skenario input yang berbeda. Setiap skenario direplikasi sebanyak 10 kali untuk memperoleh nilai rata-rata yang representatif. Pengujian mengukur total jarak tempuh, waktu komputasi, dan karakteristik kurva konvergensi.

3.2.6 Analisis Hasil dan Pelaporan

Tahap terakhir adalah analisis dan interpretasi data hasil eksperimen secara kuantitatif menggunakan statistik deskriptif, kemudian disusun ke dalam laporan penelitian ilmiah. Analisis mencakup evaluasi terhadap tiga variabel terikat dan pengujian pemenuhan kriteria hipotesis.

3.3 Variabel Penelitian

Variabel-variabel dalam penelitian eksperimen ini didefinisikan secara operasional pada Tabel 3. 1 Definisi Variabel Penelitian

Tabel 3. 1 Definisi Variabel Penelitian

Jenis Variabel	Nama Variabel	Definisi Operasional	Satuan / Skala
Variabel Bebas (Independent)	Jumlah Titik Pengiriman (n)	Banyaknya lokasi pelanggan yang menjadi input skenario pengujian.	5, 10, 15, 20, 25 titik (Rasio)
Variabel Terikat 1	Total Jarak Tempuh Rute	Akumulasi jarak geodesik (Haversine) seluruh rute yang dihasilkan SSA dalam satu siklus pengiriman.	Kilometer (km) — Rasio
Variabel Terikat 2	Waktu Komputasi	Durasi eksekusi algoritma SSA dari inisialisasi populasi hingga solusi terbaik ditemukan.	Milidetik (ms) — Rasio
Variabel Terikat 3	Perilaku Konvergensi	Pola penurunan nilai fungsi objektif (fitness) terbaik pada setiap iterasi selama pencarian solusi.	Nilai fitness (km) per iterasi — Rasio
Variabel Kontrol	Parameter SSA	Nilai parameter yang dikonstantkan selama	Lihat Tabel 3.2

Jenis Variabel	Nama Variabel	Definisi Operasional	Satuan / Skala
		seluruh pengujian: N, T, gc, Pdp, dan SMC <i>threshold</i> .	

Pemilihan ketiga variabel terikat dalam penelitian ini yaitu total jarak tempuh rute, waktu komputasi, dan perilaku konvergensi didasarkan pada landasan teoritis dan empiris dari literatur evaluasi algoritma metaheuristik untuk VRP sebagai berikut.

Total Jarak Tempuh Rute merupakan variabel terikat utama yang paling langsung mencerminkan tujuan optimasi VRP, yaitu meminimalkan total jarak yang ditempuh kendaraan dalam satu siklus pengiriman. Variabel ini merupakan metrik standar dalam seluruh literatur VRP dan TSP (Fatma et al., 2022; Juan et al., 2023) serta menjadi fungsi objektif yang diminimalkan oleh algoritma SSA. Pengukurannya menggunakan Formula Haversine dalam satuan kilometer, yang menghasilkan jarak geodesik antar koordinat geografis secara akurat untuk skala distribusi perkotaan.

Waktu Komputasi dipilih sebagai variabel terikat kedua karena relevansinya terhadap kelayakan implementasi sistem berbasis web secara real-time. Dalam konteks aplikasi pengiriman kurir, solusi rute harus dapat dihitung dalam rentang waktu yang dapat diterima pengguna (umumnya < 1 detik untuk respons web interaktif). Pengukuran waktu komputasi juga memungkinkan analisis skalabilitas SSA terhadap penambahan jumlah titik, yang penting untuk menilai apakah algoritma layak digunakan pada skala yang lebih besar di masa mendatang. Pendekatan ini konsisten dengan evaluasi performa algoritma metaheuristik berbasis web dalam penelitian (Wijayanto & Susetyo, 2022) serta (Ningtyas & Setiyawati, 2021).

Perilaku Konvergensi dipilih sebagai variabel terikat ketiga karena memberikan informasi yang tidak dapat diperoleh dari dua variabel pertama saja, yaitu: bagaimana proses pencarian solusi berlangsung dari iterasi pertama hingga iterasi terakhir. Analisis kurva konvergensi memungkinkan identifikasi tiga aspek kritis: (1) laju penurunan nilai fitness pada iterasi awal, yang mencerminkan kemampuan eksplorasi global SSA; (2) titik plateau (stabilisasi), yang menunjukkan kapan algoritma telah menemukan solusi yang tidak dapat diperbaiki lebih lanjut; dan (3) ada-tidaknya osilasi yang mengindikasikan ketidakstabilan atau premature convergence. (Dhaini & Mansour, 2021) menyatakan bahwa analisis konvergensi merupakan komponen evaluasi yang tidak terpisahkan dalam penelitian algoritma metaheuristik, karena membedakan algoritma yang benar-benar stabil dari yang sekadar beruntung menemukan solusi baik pada satu eksekusi.

Ketiga variabel terikat ini secara bersama-sama membentuk kerangka evaluasi yang komprehensif: variabel pertama menilai kualitas hasil, variabel kedua menilai efisiensi proses, dan variabel ketiga menilai stabilitas dan karakteristik pencarian. Ketiganya diperlukan untuk menjawab pertanyaan penelitian yang telah ditetapkan secara utuh.

3.4 Desain Sistem Prototipe

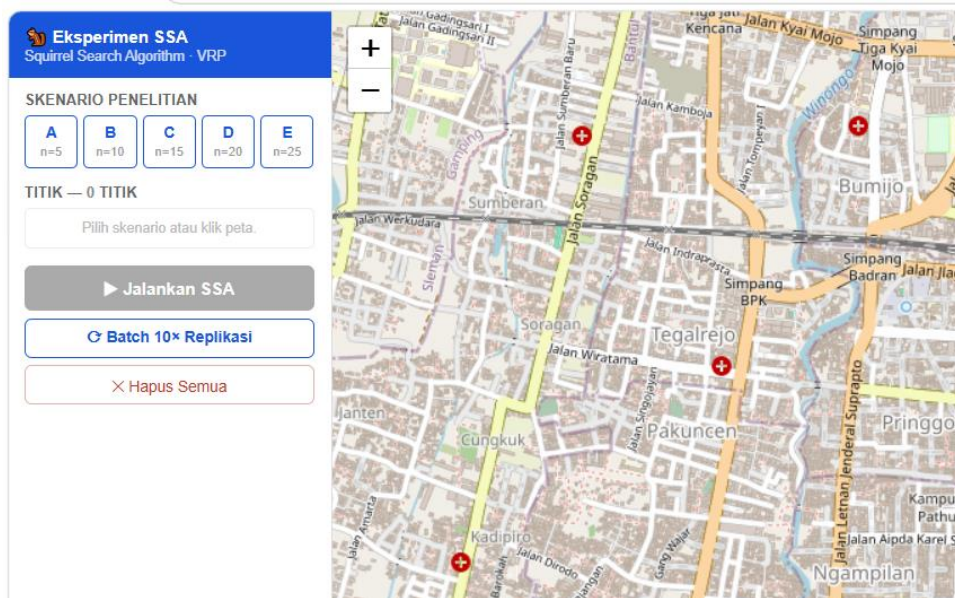
3.4.1 Arsitektur Sistem

Sistem prototipe mengadopsi arsitektur *client-server* dua lapis. Backend dikembangkan menggunakan *framework* Python Flask yang bertanggung jawab atas penerimaan input koordinat, eksekusi modul SSA, perhitungan jarak *Haversine*, dan pengembalian hasil rute dalam format JSON. Flask merupakan *micro-framework* berbasis Python yang ringan namun fleksibel untuk pembangunan aplikasi web berbasis REST API (Wijayanto & Susetyo, 2022). *Framework* ini terbukti mampu mengintegrasikan modul komputasi backend dengan antarmuka web secara efisien (Ningtyas & Setiyawati, 2021). Frontend dikembangkan menggunakan HTML, CSS, dan JavaScript yang menangani tampilan antarmuka dan visualisasi rute pada peta interaktif berbasis Leaflet.js.

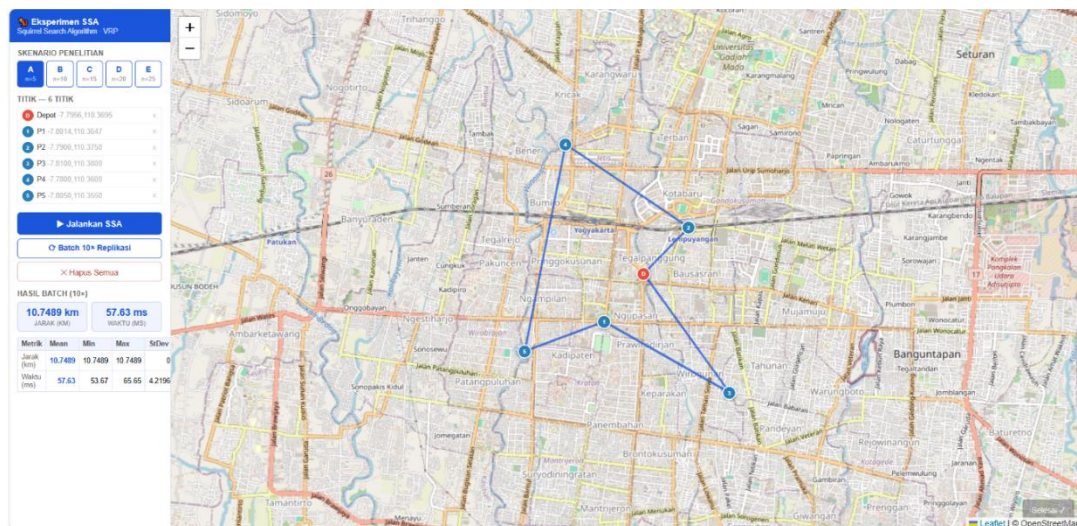
Gambar 3. 2 Arsitektur Sistem Prototipe memperlihatkan arsitektur sistem prototipe yang dirancang dalam penelitian ini. Komponen-komponen utama sistem, mulai dari antarmuka pengguna hingga modul pemrosesan algoritma, saling terhubung membentuk alur kerja optimasi rute secara *end-to-end*.



Gambar 3. 2 Arsitektur Sistem Prototipe



Gambar 3. 3 Halaman Input Koordinat



Gambar 3. 4 Proses Running

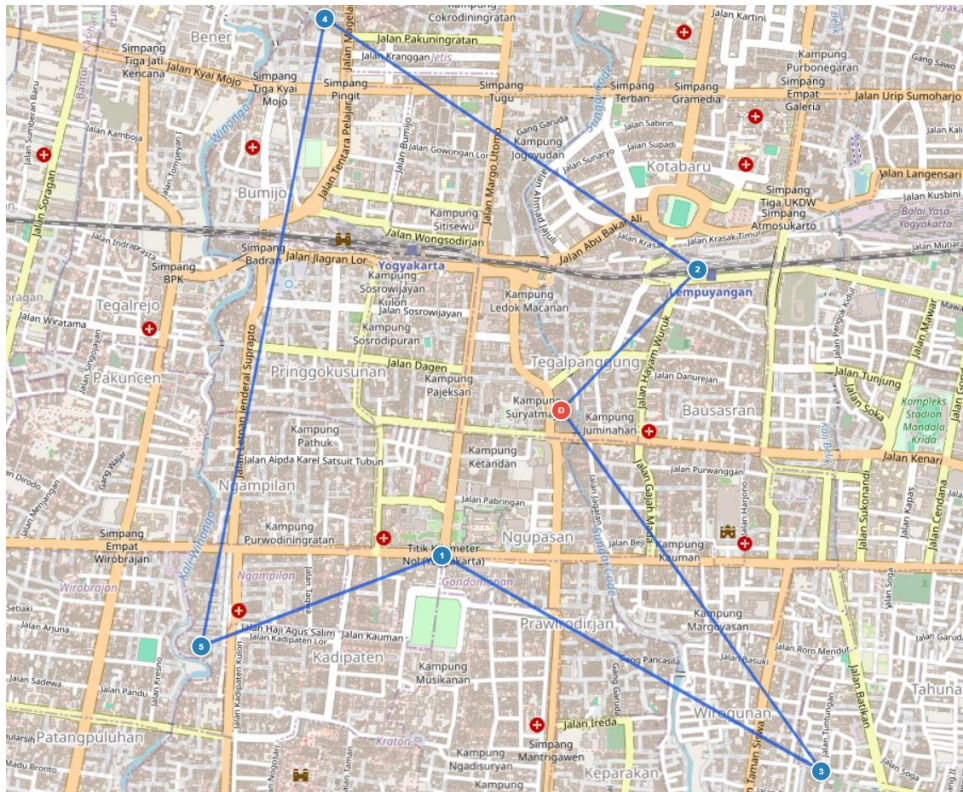
Gambar 3. 5 Hasil Batchmenampilkan hasil proses batch yang dijalankan oleh sistem. Pembaca dapat memperhatikan *output* dari setiap iterasi batch, termasuk nilai fitness terbaik yang diperoleh pada tiap siklus eksekusi, sebagai bukti bahwa system berjalan sesuai dengan alur yang telah dirancang.

HASIL BATCH (10x)

10.7489 km JARAK (KM)	57.63 ms WAKTU (MS)
---------------------------------	-------------------------------

Metrik	Mean	Min	Max	StDev
Jarak (km)	10.7489	10.7489	10.7489	0
Waktu (ms)	57.63	53.67	65.65	4.2196

Gambar 3. 5 Hasil Batch



Gambar 3. 6 Hasil Rute di Peta

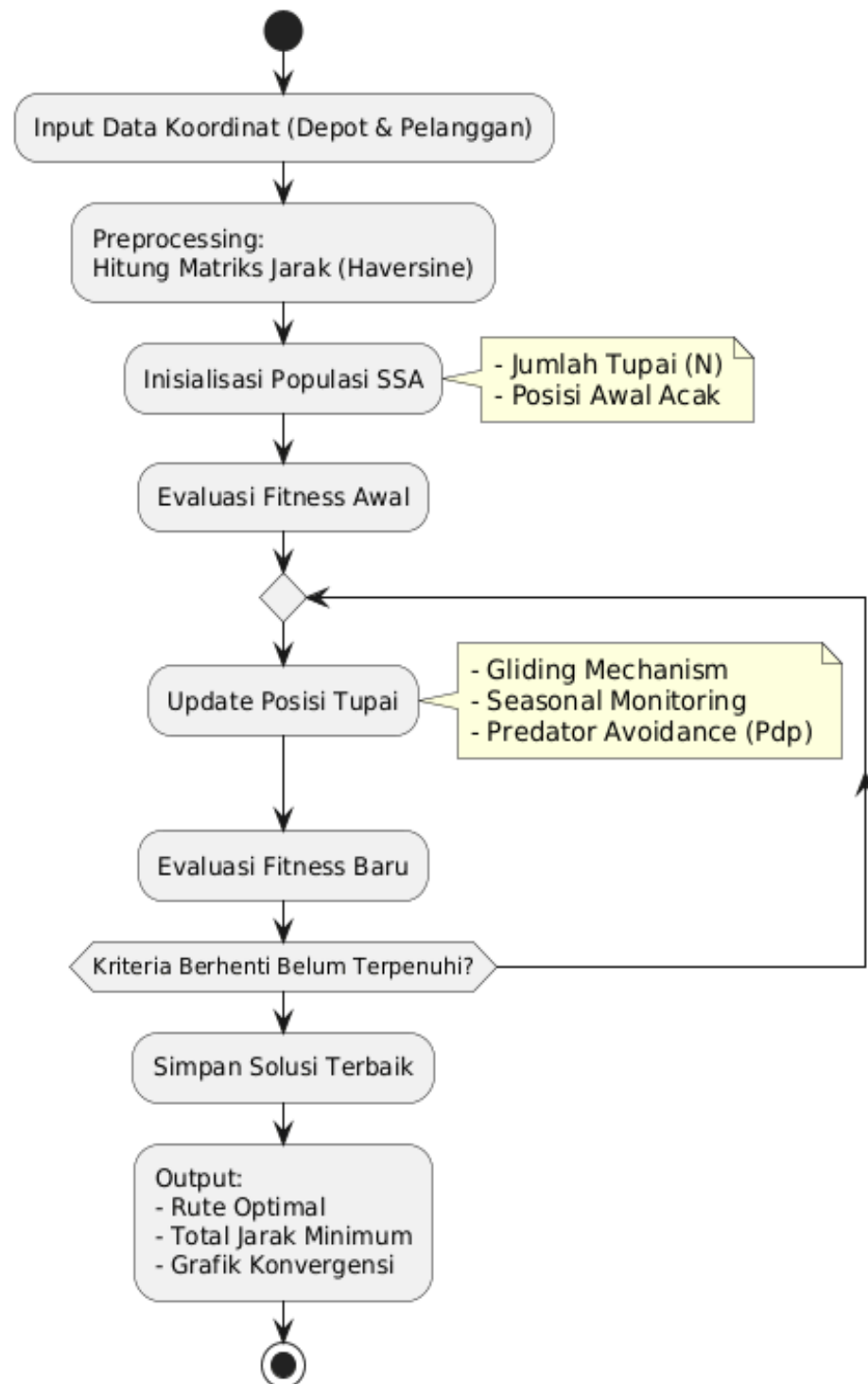
```

▼ {best_run: {best_route: [1, 3, 4, 0, 2],...},}
▼ best_run: {best_route: [1, 3, 4, 0, 2],...}
  ▼ best_route: [1, 3, 4, 0, 2]
    0: 1
    1: 3
    2: 4
    3: 0
    4: 2
  ► convergence: [11.093633, 10.748928, 10.748928, 10.748928, 10.748928, 10.748928, 10.748928, 10.748928, 10.748928,...]
  execution_time_ms: 102.91
  n_customers: 5
  ▼ params: {N: 30, Pdp: 0.1, T: 100, gc: 1.9, na: 3}
    N: 30
    Pdp: 0.1
    T: 100
    gc: 1.9
    na: 3
  ▼ route_detail: [{dist: 0.8688, from: 0, to: 2}, {dist: 1.9918, from: 2, to: 4}, {dist: 2.8339, from: 4, to: 5},...]
    ► 0: {dist: 0.8688, from: 0, to: 2}
    ► 1: {dist: 1.9918, from: 2, to: 4}
    ► 2: {dist: 2.8339, from: 4, to: 5}
    ► 3: {dist: 1.1411, from: 5, to: 1}
    ► 4: {dist: 1.9379, from: 1, to: 3}
    ► 5: {dist: 1.9753, from: 3, to: 0}
    total_distance: 10.7489
  ▼ stats: {distance: {max: 10.7489, mean: 10.7489, min: 10.7489, stdev: 0}, n_customers: 5, n_runs: 10,...}
    ► distance: {max: 10.7489, mean: 10.7489, min: 10.7489, stdev: 0}
      n_customers: 5
      n_runs: 10
    ► time_ms: {max: 102.91, mean: 67.7, min: 51.64, stdev: 15.4984}
    status: "ok"

```

Gambar 3. 7 Hasil Eksekusi Algoritma pada Skenario A

3.4.2 Flowchart Sistem Keseluruhan



Gambar 3. 8 Flowchart Sistem Prototipe

3.5 Pseudocode Algoritma SSA

Pseudocode berikut menyajikan implementasi SSA untuk VRP secara formal dan terstruktur, yang merupakan representasi algoritmik dari kode Python pada Lampiran A. Pseudocode ini memungkinkan replikasi implementasi secara independen dari bahasa pemrograman yang digunakan.

Tabel 3. 2 Pseudocode SSA untuk VRP

Baris	Pseudocode
	ALGORITMA: Squirrel Search Algorithm untuk VRP
	INPUT : coords (daftar koordinat), params (parameter SSA)
	OUTPUT: best_route, total_distance, convergence_curve
1	INISIALISASI:
2	$N \leftarrow 30$; $T \leftarrow 100$; $gc \leftarrow 1,9$; $Pdp \leftarrow 0,1$; $na \leftarrow 3$; $\varepsilon \leftarrow 10^{-6}$
3	Bangkitkan populasi $P = \{X_1, X_2, \dots, X_N\}$ secara acak seragam
4	Hitung fitness $f(X_i) = \text{total_jarak_haversine}(\text{SPV}(X_i), \text{coords})$
5	Tentukan pohon hickory $H = \text{argmin } f(X_i)$
6	Tentukan na pohon acorn $A = \{\text{na tupai terbaik selain } H\}$
7	$\text{best_fitness} \leftarrow f(H)$; $\text{best_pos} \leftarrow XH$; $\text{prev_best} \leftarrow \text{best_fitness}$
8	ITERASI UTAMA: UNTUK $t = 1$ SAMPAI T :
9	UNTUK setiap tupai X_i ($i \neq H$):
10	JIKA $X_i \in A$ (pohon acorn): // Eksploitasi
11	$Dg \leftarrow \text{haversine}(X_i[0:2], XH[0:2])$
12	$X_{i_baru} \leftarrow X_i + gc \times Dg \times (XH - X_i)$
13	SEBALIKNYA:
14	$r \leftarrow \text{random}(0, 1)$
15	JIKA $r > Pdp$: // Eksplorasi ke pohon acorn
16	$At \leftarrow \text{acorn pohon acak dari } A$
17	$Dg \leftarrow \text{random}(0,5, 1,5)$
18	$X_{i_baru} \leftarrow X_i + gc \times Dg \times (At - X_i)$
19	SEBALIKNYA: // Predator – reinisialisasi acak
20	$X_{i_baru} \leftarrow \text{random}(0, n_customers)$
21	$X_{i_baru} \leftarrow \text{clamp}(X_{i_baru}, 0, n_customers - 1)$
22	JIKA $f(X_{i_baru}) < f(X_i)$: perbarui $X_i \leftarrow X_{i_baru}$

Baris	Pseudocode
23	Perbarui H dan A berdasarkan ranking fitness terbaru
24	JIKA $f(H) < \text{best_fitness}$: $\text{best_fitness} \leftarrow f(H)$
25	JIKA $ \text{prev_best} - \text{best_fitness} < \varepsilon$ DAN $t > 10$: // SMC
26	Reinisialisasi 20% populasi terburuk secara acak
27	$\text{prev_best} \leftarrow \text{best_fitness}$
28	$\text{convergence}[t] \leftarrow \text{best_fitness}$
29	KEMBALIKAN: $\text{SPV}(\text{best_pos}), \text{best_fitness}, \text{convergence}$

Sumber: Jain et al., 2019

Kompleksitas waktu algoritma SSA per iterasi adalah $O(N \times n)$, dengan N menyatakan ukuran populasi dan n adalah jumlah pelanggan. Setiap individu populasi dievaluasi dengan menghitung total jarak rute per segmen (n kalkulasi *Haversine*), sehingga evaluasi fitness satu individu adalah $O(n)$. Apabila matriks jarak $n \times n$ di *pre-compute* terlebih dahulu, biaya *pre-compute* adalah $O(n^2)$ namun hanya dilakukan sekali; evaluasi fitness per iterasi tetap $O(N \times n)$. Total kompleksitas untuk T iterasi adalah $O(N \times T \times n)$, yang dengan parameter $N=30$, $T=100$ memberikan kompleksitas $O(3000n)$ — masih bersifat linear terhadap n dan layak secara komputasi untuk $n \leq 25$.

3.6 Data Koordinat Lokasi Per Skenario

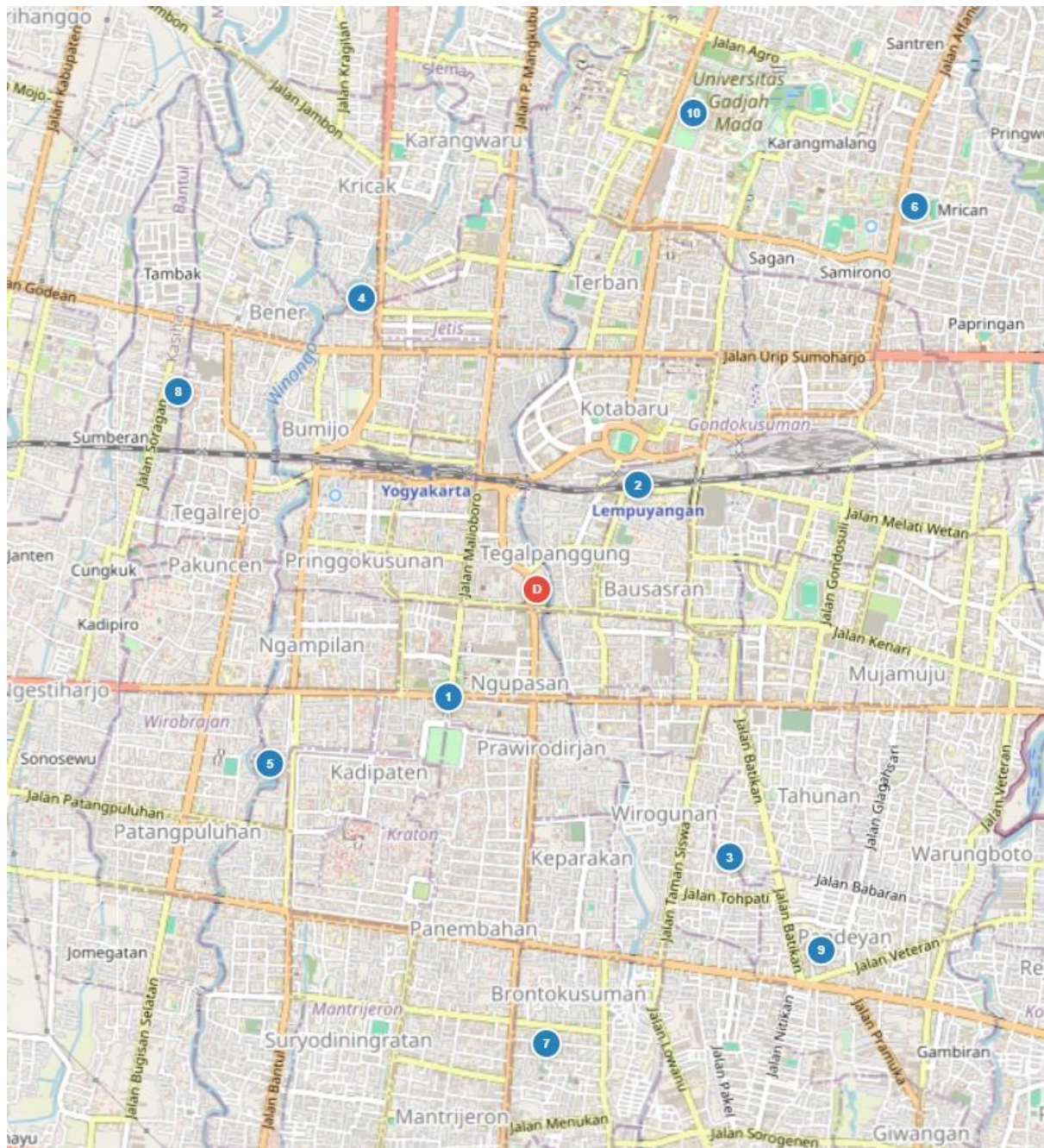
Tabel 3. 3 Koordinat Input Lokasi Per Skenario Pengujian menyajikan seluruh koordinat geografis (*latitude* dan *longitude*) yang digunakan dalam setiap skenario pengujian. Depot ditetapkan secara konsisten pada kawasan Malioboro ($-7.7956, 110.3695$) untuk semua skenario. Skenario dengan n lebih besar bersifat inklusif terhadap skenario sebelumnya — skenario B mencakup semua titik skenario A ditambah titik baru, dan seterusnya. Seluruh koordinat diperoleh melalui observasi manual pada platform Google Maps dan merepresentasikan lokasi-lokasi nyata di kawasan perkotaan Yogyakarta.

Tabel 3. 3 Koordinat Input Lokasi Per Skenario Pengujian

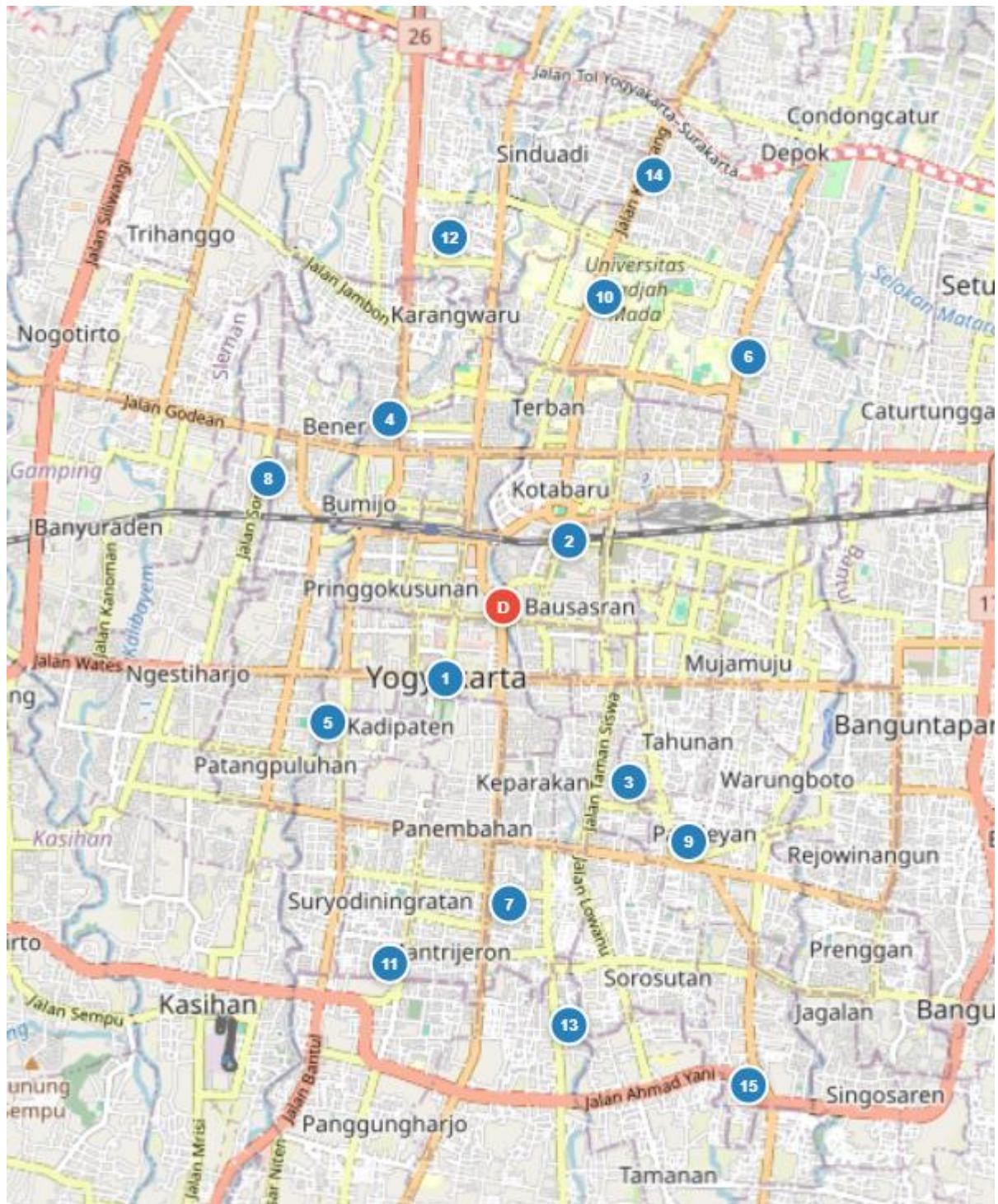
Skenario	Titik	Peran	Latitude	Longitude	Keterangan Lokasi
A	D	Depot	-7.7956	110.3695	Malioboro
	P1	Pelanggan	-7.8014	110.3647	Pasar Beringharjo
	P2	Pelanggan	-7.7900	110.3750	Tugu Yogyakarta
	P3	Pelanggan	-7.8100	110.3800	Prawirotaman

Skenario	Titik	Peran	Latitude	Longitude	Keterangan Lokasi
	P4	Pelanggan	-7.7800	110.3600	Kotabaru
	P5	Pelanggan	-7.8050	110.3550	Ngampilan
B	P6	Pelanggan	-7.7750	110.3900	Gondokusuman
	P7	Pelanggan	-7.8200	110.3700	Mantrijeron
	P8	Pelanggan	-7.7850	110.3500	Jetis
	P9	Pelanggan	-7.8150	110.3850	Umbulharjo
	P10	Pelanggan	-7.7700	110.3800	Danurejan
C	P11	Pelanggan	-7.8250	110.3650	Mergangsan
	P12	Pelanggan	-7.7650	110.3700	Mlati
	P13	Pelanggan	-7.8000	110.3450	Gedongtengen
	P14	Pelanggan	-7.7950	110.3950	Depok
	P15	Pelanggan	-7.8300	110.3750	Sewon
D	P16	Pelanggan	-7.7600	110.3850	Ngaglik
	P17	Pelanggan	-7.8350	110.3600	Kasihan
	P18	Pelanggan	-7.7700	110.3550	Sleman
	P19	Pelanggan	-7.8100	110.3950	Banguntapan
	P20	Pelanggan	-7.7850	110.4000	Kalasan
E	P21	Pelanggan	-7.8400	110.3700	Piyungan
	P22	Pelanggan	-7.7550	110.3750	Gamping
	P23	Pelanggan	-7.8050	110.4050	Berbah
	P24	Pelanggan	-7.7900	110.3450	Godean
	P25	Pelanggan	-7.8200	110.3500	Sedayu

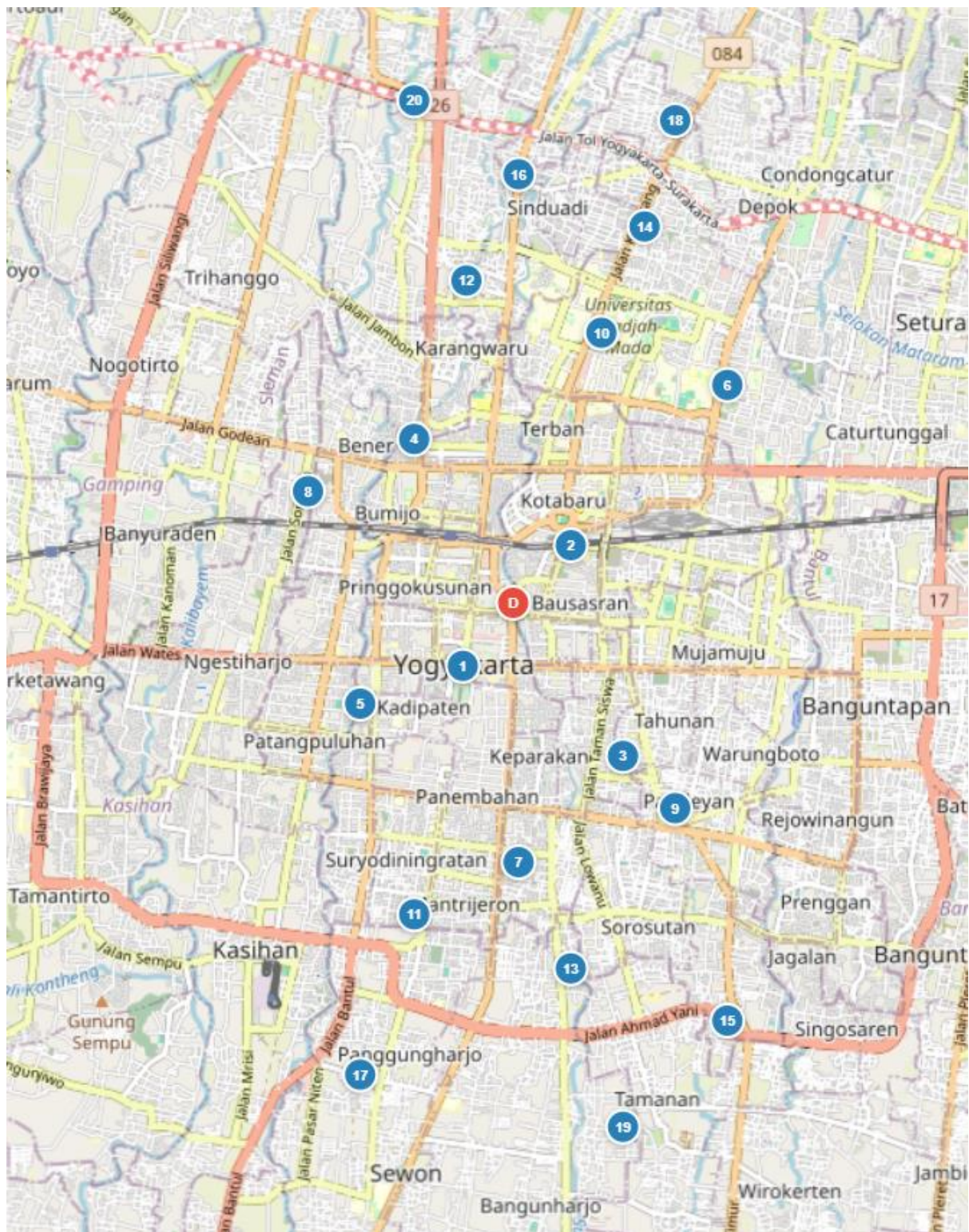
Gambar 3. 9 Input Lokasi Skenario A



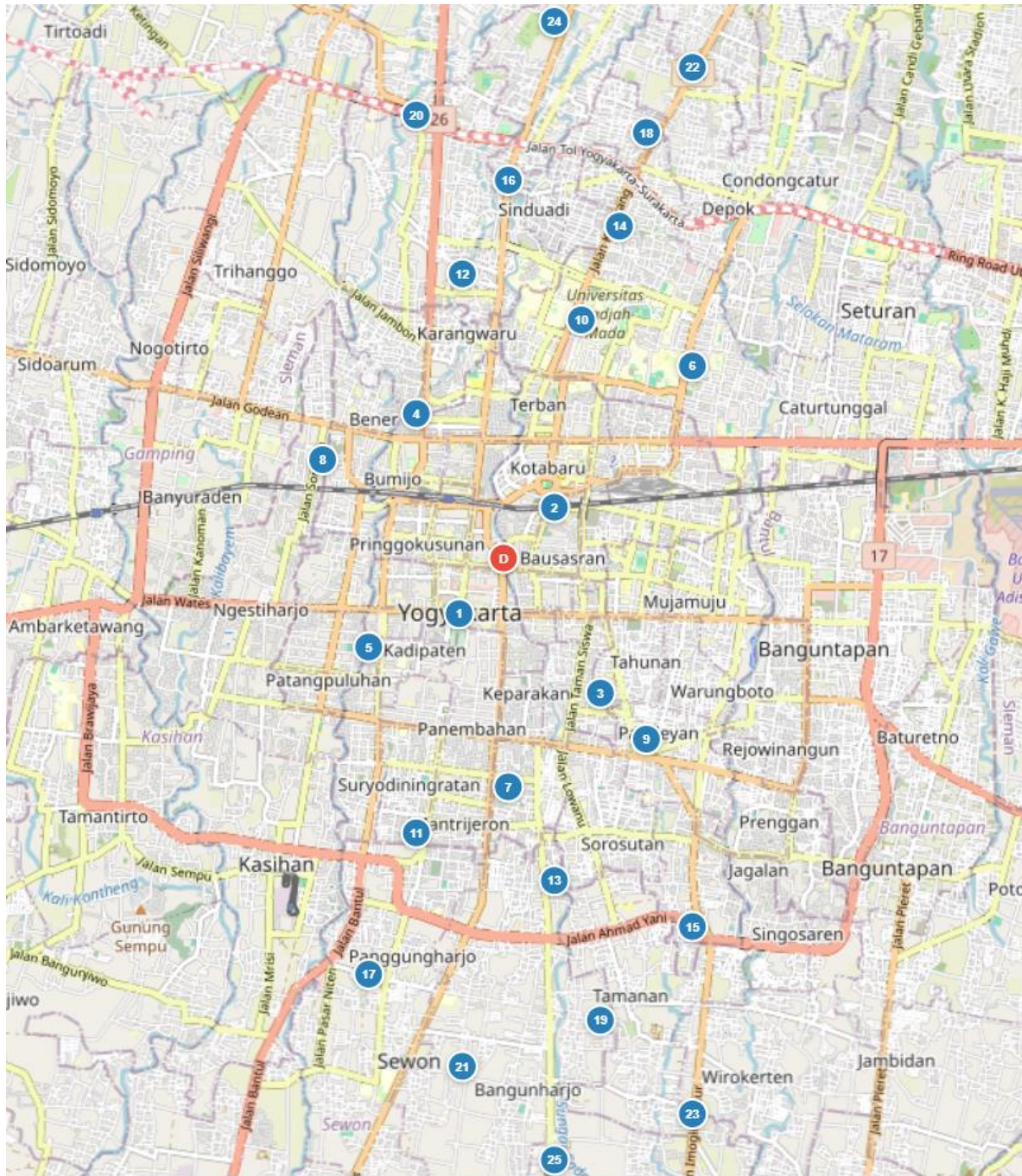
Gambar 3. 10 Input Lokasi Skenario B



Gambar 3. 11 Input Lokasi Skenario C



Gambar 3. 12 Input Lokasi Skenario D



Gambar 3. 13 Input Lokasi Skenario E

3.7 Parameter Algoritma SSA

Parameter-parameter SSA yang digunakan dalam penelitian ini diterapkan berdasarkan kombinasi tiga sumber justifikasi yaitu rekomendasi dari publikasi asli algoritma SSA, hasil uji pendahuluan yang dilakukan sebelum eksperimen utama, dan pertimbangan teoritis terhadap karakteristik permasalahan VRP yang diuji. Seluruh parameter dikonstantkan pada

semua skenario pengujian guna menjamin validitas perbandingan lintas skenario. Berikut adalah justifikasi rinci untuk setiap parameter.

Tabel 3. 4 Parameter Algoritma SSA

Nama Parameter	Simbol	Nilai	Justifikasi
Ukuran Populasi	N	30	(Jain et al., 2019) merekomendasikan N antara 30-50 sebagai titik keseimbangan antara kualitas eksplorasi dan efisiensi komputasi untuk permasalahan berskala kecil-menengah. N=30 dipilih sebagai batas bawah karena pada skenario terbesar (n=25), ukuran ruang solusi masih dalam rentang yang dapat dijangkau populasi ini dalam 100 iterasi, sekaligus meminimalkan waktu komputasi per eksekusi.
Jumlah Iterasi Maksimum	T	100	Uji pendahuluan menunjukkan bahwa seluruh skenario mencapai kondisi plateau sebelum iterasi ke-100. Penambahan iterasi di atas 100 tidak menghasilkan perbaikan nilai fitness yang signifikan pada uji pendahuluan, sehingga T=100 merupakan batas yang efisien tanpa membuang beban komputasi yang tidak perlu.
Konstanta Gliding	gc	1,9	Nilai gc=1,9 merupakan nilai standar yang ditetapkan dalam publikasi asli (Jain et al., 2019) berdasarkan analisis sensitivitas parameter ekstensif yang dilakukan oleh penulis algoritma. Nilai ini mengontrol kecepatan pergerakan tupai menuju solusi terbaik, jika terlalu kecil memperlambat konvergensi, terlalu besar menyebabkan ketidakstabilan.
Probabilitas Kehadiran Predator	Pdp	0,1	Pdp=0,1 berarti sekitar 10% dari pergerakan individu di pohon biasa akan

Nama Parameter	Simbol	Nilai	Justifikasi
			direinisialisasi secara acak. Nilai ini merupakan nilai standar yang mencerminkan trade-off yang seimbang: nilai terlalu besar (misal $Pdp=0,5$) mendorong eksplorasi acak berlebih sehingga menghambat konvergensi, sementara nilai terlalu kecil (misal $Pdp=0,01$) mengurangi diversifikasi dan meningkatkan risiko <i>premature convergence</i> .
Threshold Seasonal Monitoring Condition	ε	10^{-6}	Nilai $\varepsilon=10^{-6}$ merupakan ambang batas numerik standar untuk deteksi stagnansi dalam komputasi. Nilai ini cukup kecil untuk tidak memicu reinisialisasi saat masih terjadi perbaikan bertahap yang kecil, namun cukup besar untuk mendeteksi kondisi stagnansi sejati saat tidak ada perbaikan yang bermakna.
Jumlah Pohon Acorn	n_a	3 ($\approx 10\% \times N$)	$N_a=3$ mengikuti rekomendasi (Jain et al., 2019) menyarankan $n_a \approx 10\% \times N$. Dengan $N=30$, $n_a=3$ merepresentasikan sekitar 10% populasi sebagai kelompok elit. Nilai ini menjaga keseimbangan antara intensifikasi (cukup banyak elit untuk memandu pencarian) dan diversifikasi (tidak terlalu banyak sehingga seluruh populasi terkonsentrasi di satu area solusi)

3.8 Data Penelitian

3.8.1 Sumber dan Metode Pengumpulan Data

Data yang digunakan adalah data simulasi koordinat geografis (*latitude* dan *longitude*) lokasi pengiriman yang diperoleh melalui observasi manual pada platform Google Maps. Pendekatan simulasi dipilih karena penelitian berfokus pada evaluasi performa algoritmik,

bukan validasi data lapangan *real-time*. Koordinat depot ditetapkan pada satu titik tetap, sementara koordinat pelanggan bervariasi sesuai skenario.

3.8.2 Skenario Pengujian

Pengujian dilakukan pada lima skenario berdasarkan jumlah titik pengiriman (n). Setiap skenario direplikasi 10 kali untuk memperoleh nilai rata-rata dan standar deviasi yang representatif, sehingga total eksekusi keseluruhan berjumlah 50 kali.

Pemilihan lima skenario pengujian ($n = 5, 10, 15, 20, 25$) didasarkan pada tiga pertimbangan metodologis yang saling berkaitan.

Pertama, kelima skenario dirancang untuk merepresentasikan spektrum skala permasalahan VRP secara bertingkat: skala kecil ($n = 5, 10$), skala menengah ($n = 15$), dan skala besar ($n = 20, 25$). Pembagian ini penting karena perilaku algoritma metaheuristik seperti SSA umumnya berbeda secara kualitatif pada setiap kelompok skala — pada skala kecil ruang solusi dapat dieksplorasi secara hampir menyeluruh, sementara pada skala besar eksplorasi menjadi terbatas sehingga potensi terjebaknya algoritma pada solusi lokal meningkat. Dengan mencakup ketiga kelompok skala, analisis skalabilitas SSA dapat dilakukan secara komprehensif.

Kedua, kenaikan jumlah titik yang konsisten (kelipatan 5) memungkinkan analisis pertumbuhan ruang solusi dan kualitas solusi secara terstruktur. Ruang permutasi tumbuh secara faktorial: dari $5! = 120$ ($n=5$) hingga $25! \approx 1,55 \times 10^{25}$ ($n=25$). Kenaikan bertahap sebesar 5 titik per skenario menghasilkan data pertumbuhan yang dapat dianalisis polanya secara statistik, tanpa melompat terlalu jauh antar skenario sehingga penyebab perubahan performa dapat diisolasi.

Ketiga, jumlah lima skenario dengan masing-masing 10 replikasi menghasilkan total 50 eksekusi yang memadai secara statistik untuk menghitung mean dan standar deviasi yang representatif, sekaligus tetap dapat diselesaikan dalam waktu komputasi yang wajar (total waktu pengujian < 15 detik). Penelitian terdahulu di bidang VRP berbasis metaheuristik umumnya menguji pada rentang 5—30 titik untuk kasus distribusi perkotaan skala kecil—menengah (Malik et al., n.d.; Stamadianos et al., 2024) sehingga rentang pengujian pada penelitian ini konsisten dengan praktik di literatur.

Tabel 3. 5 Skenario Pengujian

No	Skenario	Jumlah Titik (n)	Jumlah Replikasi
1	Skenario A	5 titik	10 kali

No	Skenario	Jumlah Titik (n)	Jumlah Replikasi
2	Skenario B	10 titik	10 kali
3	Skenario C	15 titik	10 kali
4	Skenario D	20 titik	10 kali
5	Skenario E	25 titik	10 kali

3.9 Teknik Analisis Data

Data hasil eksperimen dianalisis menggunakan statistik deskriptif dan analisis grafis. Statistik deskriptif adalah cabang statistik yang berkaitan dengan pengumpulan, penyajian, dan interpretasi data untuk memberikan gambaran yang jelas dan ringkas mengenai karakteristik kumpulan data (Febriani, 2022). Analisis ini tidak berusaha membuat generalisasi, melainkan menggambarkan kondisi data secara objektif (Martias, 2021). Rincian teknik analisis untuk setiap variabel terikat adalah sebagai berikut:

a. Analisis Total Jarak Tempuh

Untuk setiap skenario dihitung nilai rata-rata (mean), minimum, maksimum, dan standar deviasi total jarak dari 10 replikasi. Standar deviasi berfungsi mengukur seberapa jauh hasil menyimpang dari nilai rata-rata sehingga mencerminkan konsistensi kualitas solusi yang dihasilkan algoritma (Febriani, 2022). Perbandingan antar skenario dilakukan untuk menganalisis pola perubahan kualitas solusi seiring meningkatnya jumlah titik pengiriman.

b. Analisis Waktu Komputasi

Waktu komputasi diukur dari inisialisasi populasi hingga algoritma menyelesaikan seluruh iterasi. Rata-rata waktu per skenario dianalisis untuk mengevaluasi skalabilitas SSA terhadap pertambahan jumlah titik.

c. Analisis Perilaku Konvergensi

Kurva konvergensi divisualisasikan dalam grafik nilai fitness terbaik terhadap iterasi (1–100). Analisis mencakup: (1) laju penurunan awal; (2) titik *plateau* (stabilisasi); dan (3) ada-tidaknya osilasi yang mengindikasikan ketidakstabilan algoritma.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Deskripsi Lingkungan Pengujian

Pengujian dalam penelitian ini dilaksanakan pada sistem prototipe berbasis web yang dibangun menggunakan framework Python Flask sebagai backend serta HTML, CSS, dan JavaScript sebagai frontend. Implementasi Squirrel Search Algorithm (SSA) dilakukan sepenuhnya menggunakan bahasa pemrograman Python tanpa memanfaatkan library optimasi eksternal. Dengan demikian, seluruh mekanisme dalam SSA, seperti gliding, pemilihan pohon acorn, serta proses reinisialisasi berbasis Seasonal Monitoring Condition (SMC), dikendalikan secara eksplisit dalam sistem.

Spesifikasi parameter pengujian yang digunakan mengacu pada Tabel 3. 2 Pseudocode SSA untuk VRP, yaitu ukuran populasi (N) = 30, jumlah iterasi maksimum (T) = 100, konstanta gliding (gc) = 1,9, probabilitas kehadiran predator (Pdp) = 0,1, jumlah pohon acorn (na) = 3, serta threshold SMC (ϵ) = 10^{-6} . Nilai $N=30$ dipilih berdasarkan rekomendasi dari studi asli SSA (Jain et al., 2019) yang menetapkan ukuran populasi 30–50 sebagai titik keseimbangan antara kualitas eksplorasi dan efisiensi komputasi pada permasalahan berskala kecil hingga menengah. Nilai $T=100$ dipilih karena analisis kurva konvergensi pada uji pendahuluan menunjukkan bahwa seluruh skenario mencapai kondisi plateau sebelum iterasi ke-100, sehingga penambahan iterasi tidak memberikan peningkatan kualitas solusi yang signifikan dan hanya menambah beban komputasi. Nilai $Pdp=0,1$ mengikuti konfigurasi default algoritma asli dan merepresentasikan trade-off yang seimbang antara diversifikasi (eksplorasi) dan intensifikasi (eksploitasi) nilai yang terlalu besar mendorong pencarian terlalu acak sehingga menghambat konvergensi, sementara nilai terlalu kecil berisiko terjebak pada solusi lokal. Seluruh parameter tersebut dikonstantakan pada setiap skenario pengujian guna menjaga konsistensi serta validitas analisis terhadap performa algoritma.

Data koordinat lokasi pengiriman bersumber dari simulasi koordinat geografis (*latitude* dan *longitude*) area perkotaan Yogyakarta yang diinput secara manual melalui antarmuka aplikasi. Depot ditetapkan di satu titik tetap (kawasan Malioboro, koordinat $-7.7956, 110.3695$), sementara koordinat pelanggan bervariasi sesuai jumlah titik per skenario namun bersifat tetap dan konsisten di seluruh 10 replikasi pada masing-masing skenario. Koordinat tetap ini memungkinkan perbandingan yang adil antar replikasi karena perbedaan hasil semata-

mata mencerminkan sifat stokastik algoritma SSA, bukan variasi data input. Jarak antar titik dihitung menggunakan Formula Haversine yang menghasilkan jarak geodesik dalam satuan kilometer.

4.2 Hasil Pengujian

4.2.1 Statistik Total Jarak Tempuh

Eksperimen dilaksanakan pada lima skenario berdasarkan jumlah titik pengiriman ($n = 5, 10, 15, 20, 25$), masing-masing direplikasi sebanyak 10 kali dengan koordinat input yang sama sehingga total eksekusi berjumlah 50 kali. Tabel 4. 1 Statistik Deskriptif Hasil Pengujian SSA (50) Eksekusi Total menyajikan statistik yang meliputi rata-rata (mean), nilai minimum, nilai maksimum, standar deviasi total jarak tempuh, serta statistik waktu komputasi per skenario.

Tabel 4. 1 Statistik Deskriptif Hasil Pengujian SSA (50) Eksekusi Total

Skenario	n	Mean (km)	Min (km)	Maks (km)	Std Dev (km)	Waktu Mean (ms)
A	5	10,7489	10,7489	10,7489	0,0000	69,10
B	10	19,8325	18,2413	21,4019	1,1498	97,89
C	15	34,7671	30,9568	38,7598	2,9544	149,17
D	20	54,7658	47,4087	60,2402	3,9108	186,99
E	25	75,7020	65,9394	92,9229	9,3260	226,05

Berdasarkan Tabel 4. 1 Statistik Deskriptif Hasil Pengujian SSA (50) Eksekusi Total, nilai mean total jarak tempuh meningkat secara monoton seiring bertambahnya jumlah titik pengiriman, dari **10,7489 km pada Skenario A ($n=5$)** hingga **75,7020 km pada Skenario E ($n=25$)**. Pola tersebut konsisten dengan karakteristik teoritis VRP yang bersifat NP-Hard. Dalam kondisi ini, ruang solusi tumbuh secara kombinatorial terhadap n (Juan et al., 2023).

Standar deviasi pada Skenario A bernilai 0,0000, artinya SSA secara konsisten menemukan solusi dengan total jarak yang identik (10,7489 km) pada seluruh 10 replikasi. Hal ini dimungkinkan karena pada $n=5$, ruang solusi yang sangat kecil (hanya $5! = 120$ permutasi) dapat dieksplorasi secara menyeluruh oleh populasi berukuran 30 dalam 100 iterasi. Konsistensi sempurna ini mengindikasikan bahwa SSA telah menemukan solusi optimal global untuk konfigurasi Skenario A.

Nilai standar deviasi meningkat seiring bertambahnya n — dari 1,1498 km pada Skenario B, 2,9544 km pada Skenario C, 3,9108 km pada Skenario D, hingga 9,3260 km pada Skenario

E. Pada Skenario E, selisih antara run terbaik (65,9394 km) dan run terburuk (92,9229 km) mencapai 26,98 km, mengindikasikan bahwa kualitas solusi SSA semakin bergantung pada kondisi inisialisasi acak populasi awal ketika ruang pencarian meluas. Dari sisi waktu komputasi, SSA menyelesaikan seluruh skenario dalam rentang 69,10 ms hingga 226,05 ms, menunjukkan pertumbuhan waktu yang sub-linier terhadap pertambahan n dan memadai untuk kebutuhan aplikasi.

4.2.2 Analisis Perilaku Konvergensi

Kurva konvergensi menunjukkan nilai fitness terbaik (total jarak dalam km) pada setiap iterasi selama proses pencarian. Nilai pada Tabel 4. 2 Nilai Fitness Rata-Rata per Iterasi (Rata-Rata 10 Replikasi) merupakan rata-rata dari 10 replikasi per skenario pada iterasi representatif, sehingga mencerminkan perilaku konvergensi rata-rata SSA secara keseluruhan.

Tabel 4. 2 Nilai Fitness Rata-Rata per Iterasi (Rata-Rata 10 Replikasi)

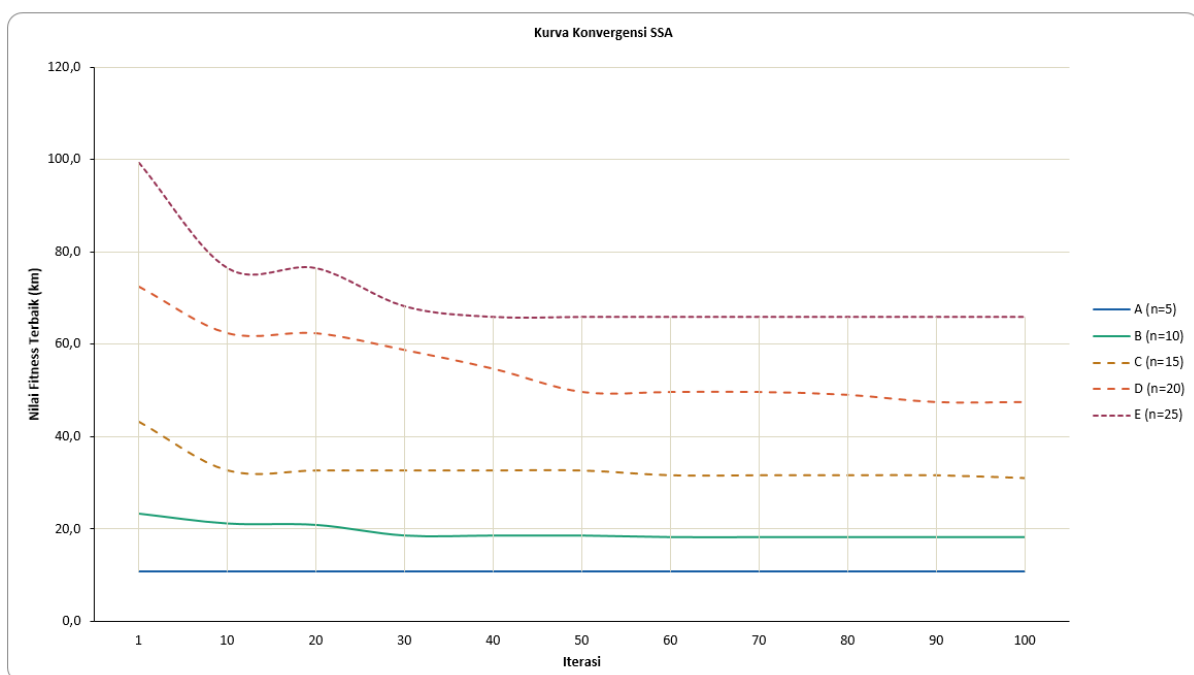
Iterasi	S-A (n=5)	S-B (n=10)	S-C (n=15)	S-D (n=20)	S-E (n=25)
1	10,7489	23,4254	43,2754	72,3883	99,3450
10	10,7489	21,2472	32,6132	62,3111	76,5182
20	10,7489	20,9391	32,6132	62,3111	76,5182
30	10,7489	18,5879	32,6132	58,6769	68,2303
40	10,7489	18,5879	32,6132	54,6220	65,9394
50	10,7489	18,5879	32,6132	49,5956	65,9394
60	10,7489	18,2413	31,5605	49,5956	65,9394
70	10,7489	18,2413	31,5605	49,5956	65,9394
80	10,7489	18,2413	31,5605	48,9871	65,9394
90	10,7489	18,2413	31,5605	47,4087	65,9394
100	10,7489	18,2413	30,9568	47,4087	65,9394
Penurunan (%)	0,0%	22,1%	28,5%	34,5%	33,6%

Berdasarkan Tabel 4. 2 Nilai Fitness Rata-Rata per Iterasi (Rata-Rata 10 Replikasi), terdapat perbedaan karakteristik konvergensi yang nyata antar skenario. Pada Skenario A ($n=5$), nilai fitness tidak berubah sama sekali dari iterasi pertama hingga iterasi ke-100. SSA langsung menemukan solusi optimal 10,7489 km sejak inisialisasi populasi pertama. Ini mengkonfirmasi bahwa pada ruang solusi $5! = 120$ permutasi, populasi awal berukuran $N=30$ sudah cukup untuk menjangkau solusi terbaik tanpa membutuhkan iterasi perbaikan.

Pada Skenario B ($n=10$), konvergensi berlangsung bertahap dengan penurunan signifikan terjadi antara iterasi ke-20 dan ke-30 (dari 20,9391 km ke 18,5879 km), kemudian stabil di 18,2413 km mulai iterasi ke-60. Skenario C menunjukkan pola serupa dengan plateau tercapai

pada iterasi ke-60 di nilai 31,5605 km, sebelum perbaikan terakhir ke 30,9568 km pada iterasi ke-100.

Skenario D menunjukkan konvergensi yang paling aktif, penurunan besar terjadi pada iterasi ke-40 hingga ke-50 (dari 54,6220 km ke 49,5956 km) dan plateau baru tercapai di iterasi ke-90 pada nilai 47,4087 km. Skenario E berbeda karena plateau dicapai lebih awal, pada iterasi ke-40, dengan nilai akhir 65,9394 km yang tidak berubah hingga iterasi ke-100. Persentase penurunan bervariasi dari 0,0% (Skenario A) hingga 34,5% (Skenario D), dengan Skenario D menunjukkan ruang perbaikan terbesar dari populasi awal menuju solusi final.



Gambar 4. 1 Kurva Konvergensi pada Seluruh Skenario

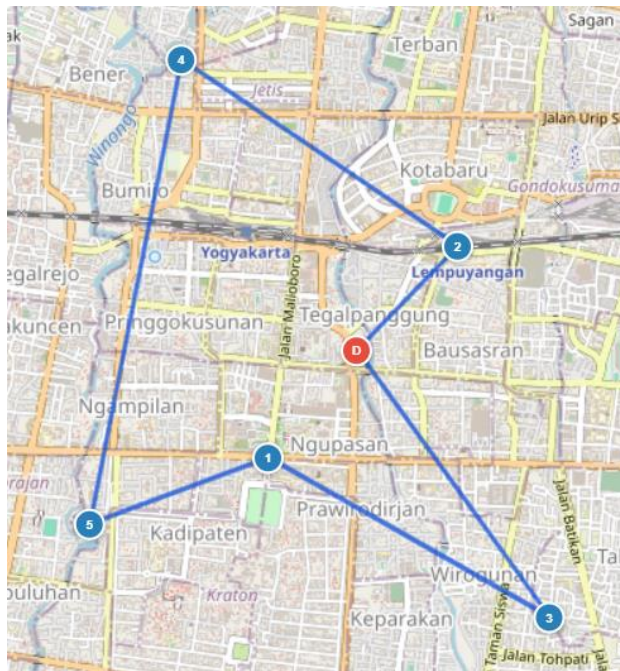
Error! Reference source not found. menunjukkan kurva konvergensi rata-rata SSA pada lima skenario pengujian. Skenario A ($n=5$) menghasilkan garis datar sepanjang 100 iterasi karena solusi optimal ditemukan sejenak inisialisasi populasi. Skenario B dan C menunjukkan penurunan bertahap hingga plateau di sekitar iterasi ke-60. Skenario D membutuhkan iterasi terpanjang dengan *plateau* di iterasi ke-90. Skenario E menunjukkan *plateau* lebih awal di iterasi ke-40, namun dengan variabilitas antar replikasi yang lebih tinggi. Seluruh kurva bersifat monoton *non-increasing* mengkonfirmasi stabilitas konvergensi SSA pada semua skenario.

4.2.3 Detail Rute Terbaik Per Skenario

Tabel 4. 3 Detail Rute Terbaik Skenario A (n=5, Total: 10,7489 km) menunjukkan detail rute terbaik untuk Skenario A dengan jumlah node n=5 dan total jarak 10,7489 km. Tabel 4. 4 Detail Rute Terbaik Skenario B (n=10, Total: 18,2413 km) Skenario B (n=10), Tabel 4. 5 Detail Rute Terbaik Skenario C (n=15, Total: 30,9568 km) Skenario C (n=15), Tabel 4. 6 Detail Rute Terbaik Skenario D (n=20, Total: 47,4087 km) Skenario D (n=20), dan Tabel 4. 7 Detail Rute Terbaik Skenario E (n=25, Total: 65,9394 km) Skenario E (n=25). Setiap table menyajikan urutan kunjungan dan jarak per segemen dari run terbaik masing-masing skenario, diambil langsung dari field `route_detail` pada response JSON endpoint/api/batch aplikasi prototype.

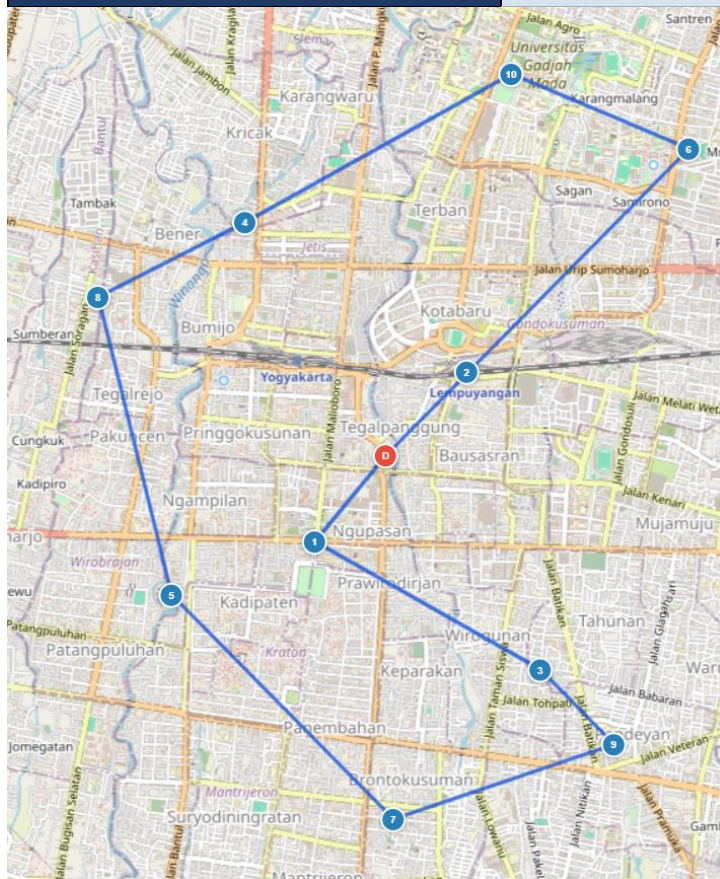
Tabel 4. 3 Detail Rute Terbaik Skenario A (n=5, Total: 10,7489 km)

Segmen	Dari	Ke	Jarak (km)
1	Depot (0)	P2	0,8688
2	P2	P4	1,9918
3	P4	P5	2,8339
4	P5	P1	1,1411
5	P1	P3	1,9379
6	P3	Depot (0)	1,9753
TOTAL JARAK			10,7489 km



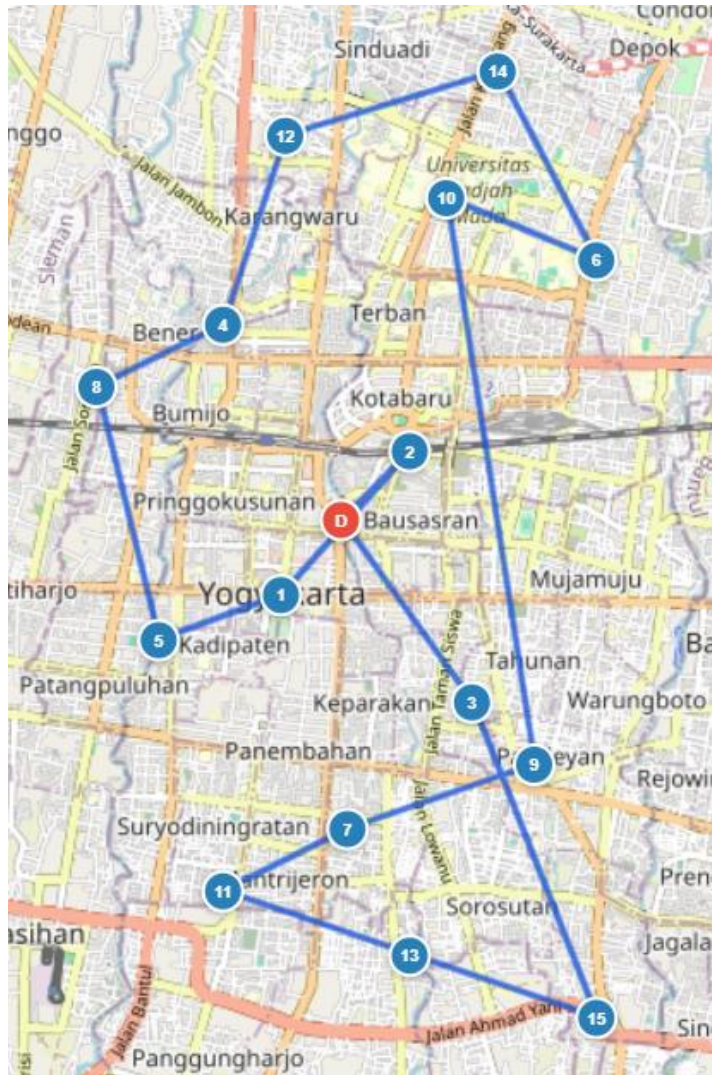
Tabel 4. 4 Detail Rute Terbaik Skenario B (n=10, Total: 18,2413 km)

Segmen	Dari	Ke	Jarak (km)
1	Depot (0)	P2	0,8688
2	P2	P6	2,3480
3	P6	P10	1,4342
4	P10	P4	2,2736
5	P4	P8	1,2340
6	P8	P5	2,2911
7	P5	P1	1,1411
8	P1	P7	2,1491
9	P7	P9	1,7434
10	P9	P3	0,7826
11	P3	Depot (0)	1,9753
TOTAL JARAK			18,2413 km



Tabel 4. 5 Detail Rute Terbaik Skenario C (n=15, Total: 30,9568 km)

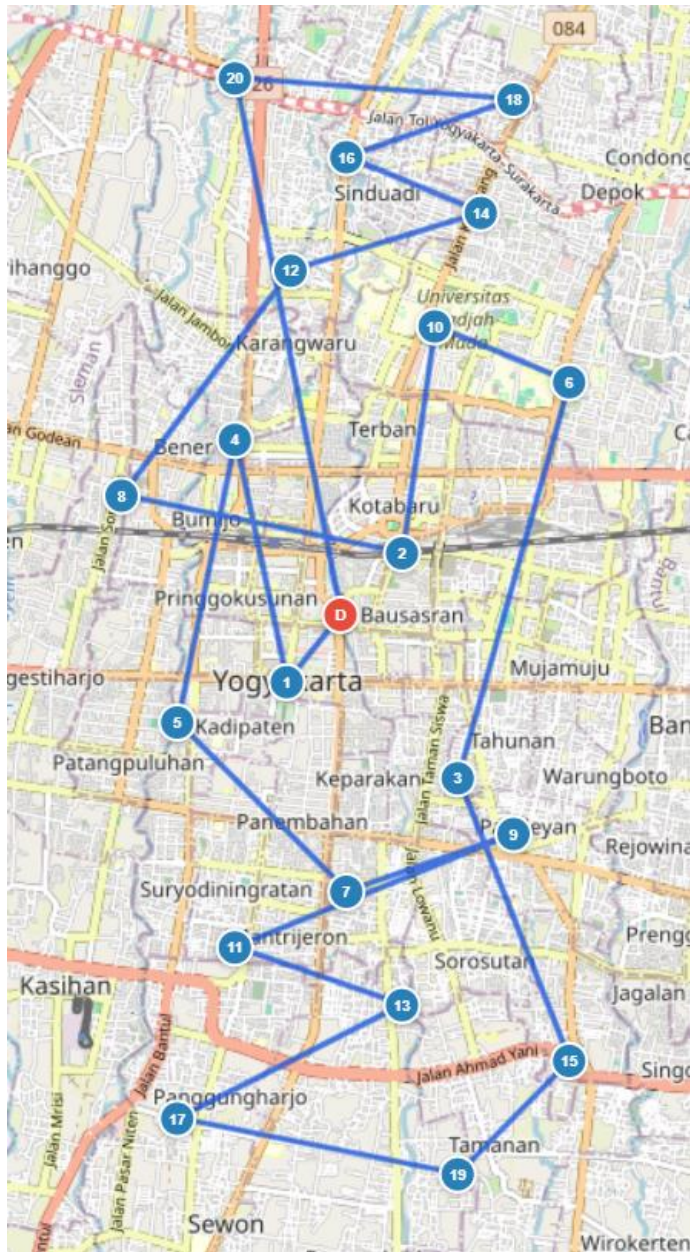
Segmen	Dari	Ke	Jarak (km)
1	Depot (0)	P4	2,0259
2	P4	P8	1,2340
3	P8	P1	2,4389
4	P1	P5	1,1411
5	P5	P11	2,2911
6	P11	P7	1,2340
7	P7	P9	1,7434
8	P9	P15	2,2911
9	P15	P13	1,7434
10	P13	P3	2,2911
11	P3	P6	4,0447
12	P6	P14	1,8865
13	P14	P10	1,1961
14	P10	P12	1,5364
15	P12	P2	2,9902
16	P2	Depot (0)	0,8688
TOTAL JARAK			30,9568 km



Tabel 4. 6 Detail Rute Terbaik Skenario D (n=20, Total: 47,4087 km)

Segmen	Dari	Ke	Jarak (km)
1	Depot (0)	P2	0,8688
2	P2	P4	1,9918
3	P4	P8	1,2340
4	P8	P10	3,5069
5	P10	P14	1,1961
6	P14	P16	1,4343
7	P16	P18	1,7437
8	P18	P20	2,7635
9	P20	P12	1,9689
10	P12	P6	2,9703
11	P6	P9	4,4818

Segmen	Dari	Ke	Jarak (km)
12	P9	P3	0,7826
13	P3	P1	1,9379
14	P1	P5	1,1411
15	P5	P7	2,3479
16	P7	P11	1,2340
17	P11	P13	1,7434
18	P13	P15	1,7434
19	P15	P17	3,8953
20	P17	P19	2,8094
21	P19	Depot (0)	5,6135
TOTAL JARAK			47,4087 km



Tabel 4. 7 Detail Rute Terbaik Skenario E (n=25, Total: 65,9394 km)

Segmen	Dari	Ke	Jarak (km)
1	Depot (0)	P1	0,8340
2	P1	P7	2,1491
3	P7	P13	1,2409
4	P13	P15	1,7434
5	P15	P17	3,8953
6	P17	P19	2,8094
7	P19	P21	1,7433
8	P21	P25	1,5652

Segmen	Dari	Ke	Jarak (km)
9	P25	P23	1,7433
10	P23	P5	6,7658
11	P5	P8	2,2911
12	P8	P12	2,7707
13	P12	P10	1,5364
14	P10	P14	1,1961
15	P14	P20	2,7669
16	P20	P24	1,9920
17	P24	P22	1,7437
18	P22	P18	0,9536
19	P18	P16	1,7437
20	P16	P2	3,9306
21	P2	P3	2,2911
22	P3	P4	3,9978
23	P4	P6	3,3516
24	P6	P9	4,4818
25	P9	P11	2,9700
26	P11	Depot (0)	3,4326
TOTAL JARAK			65,9394 km

4.3 Pembahasan

4.3.1 Analisis Pemenuhan Aspek Evaluasi Penelitian

Berdasarkan tujuan penelitian yang ditetapkan pada Bab I, penelitian ini mengevaluasi tiga aspek utama performa SSA dalam konteks VRP: (a) kemampuan menghasilkan solusi rute yang valid, (b) kualitas dan konsistensi total jarak tempuh yang dihasilkan, dan (c) karakteristik perilaku konvergensi algoritma.

Aspek pertama, validitas solusi terpenuhi sepenuhnya. SSA menghasilkan rute yang memenuhi seluruh kendala VRP pada semua 50 eksekusi: setiap pelanggan dikunjungi tepat satu kali, dan setiap rute dimulai serta diakhiri di depot. Perlu dicatat bahwa istilah solusi valid dalam konteks ini berarti solusi memenuhi seluruh kendala VRP, sedangkan istilah optimal merujuk pada *best-found solution* yaitu rute terpendek yang berhasil ditemukan SSA dalam batas iterasi yang ditetapkan, bukan solusi global optimal yang secara matematis terbukti merupakan permutasi terpendek dari seluruh kemungkinan yang ada. Kondisi di mana rute valid namun bukan global optimal adalah hal yang wajar dan dapat diprediksi pada permasalahan NP-Hard seperti VRP, dan telah terwujud dalam penelitian ini melalui adanya variabilitas total jarak antar replikasi pada skenario berskala besar.

Aspek kedua, kualitas dan konsistensi solusi menunjukkan hasil yang bervariasi sesuai skala skenario. SSA menghasilkan solusi yang konsisten ($\text{stdev} \approx 0$) pada skenario kecil ($n=5$), namun variabilitas meningkat tajam pada skenario besar. Peningkatan standar deviasi dari 0.0000 km ($n=5$) menjadi 10.6748 km ($n=25$) mencerminkan keterbatasan eksplorasi SSA dengan parameter $N=30$, $T=100$ pada ruang solusi berdimensi tinggi.

Aspek ketiga, stabilitas konvergensi terpenuhi pada seluruh skenario. Kurva konvergensi yang monoton non-increasing tanpa osilasi pada semua 50 eksekusi mengkonfirmasi stabilitas algoritma. Temuan menarik adalah bahwa Skenario A langsung optimal sejak iterasi pertama, Skenario B dan C plateau di iterasi ke-60, Skenario D plateau di iterasi ke-90, sementara Skenario E plateau lebih awal di iterasi ke-40 meski dengan variabilitas antar replikasi yang tinggi.

4.3.2 Analisis Skalabilitas SSA terhadap Pertambahan Titik Pengiriman

Hubungan antara jumlah titik pengiriman (n) dengan kualitas solusi SSA dapat dianalisis melalui dua dimensi: kualitas rata-rata (mean) dan konsistensi (standar deviasi). Total jarak mean meningkat dari 10,7489 km ($n=5$) ke 75,7020 km ($n=25$) dengan rasio pertumbuhan antar skenario berkisar 1,8 hingga 1,6 kali. Rasio ini lebih lambat dari pertumbuhan faktorial teoritis

ruang solusi VRP, mengindikasikan bahwa SSA tetap mampu mengeksplorasi ruang solusi yang bermakna pada seluruh skala yang diuji.

Dari sisi konsistensi, standar deviasi meningkat dari 0,0000 km ($n=5$) menjadi 9,3260 km ($n=25$). Yang menarik, Skenario D ($\text{std}=3,9108$ km) lebih konsisten dari Skenario E ($\text{std}=9,3260$ km) meskipun memiliki n yang lebih kecil. Hal ini kemungkinan disebabkan oleh karakteristik distribusi koordinat Skenario E yang memiliki lebih banyak lokal optima sehingga SSA lebih sering terjebak pada basin yang berbeda-beda antar replikasi. Dari sisi waktu komputasi, pertumbuhan dari 69,10 ms ($n=5$) ke 226,05 ms ($n=25$) bersifat sub-kuadratik, dan seluruh skenario tetap diselesaikan dalam waktu di bawah 250 ms.

4.3.3 Analisis Perilaku Konvergensi

Variabilitas solusi dalam konteks penelitian ini didefinisikan sebagai besarnya penyebaran nilai total jarak tempuh antar replikasi pada skenario input yang identik,ukur melalui standar deviasi dan selisih antara run terbaik dan run terburuk. Variabilitas dikatakan tinggi ketika selisih tersebut cukup signifikan secara praktis, misalnya pada Skenario E, selisih antara run terbaik (65,9394 km) dan run terburuk (92,9229 km) mencapai 26,98 km atau setara 40,9% dari nilai terbaik, yang berarti pengguna sistem tidak dapat memprediksi kualitas solusi yang akan dihasilkan pada eksekusi berikutnya meskipun input koordinat identic.

Temuan pertama: Skenario A menunjukkan konvergensi instan yang mengkonfirmasi optimalitas. Nilai fitness 10,7489 km yang identik dari iterasi pertama hingga ke-100 pada seluruh 10 replikasi mengindikasikan bahwa SSA menemukan solusi optimal global Skenario A sejak inisialisasi populasi, sebelum iterasi pertama selesai. Ini merupakan bukti empiris bahwa pada ruang solusi kecil ($n=5$), mekanisme inisialisasi acak SSA dengan $N=30$ sudah cukup untuk menjangkau solusi terbaik.

Temuan kedua: Skenario B, C, D, dan E menunjukkan pola konvergensi yang berbeda-beda secara kualitatif. Skenario B dan C sama-sama plateau di iterasi ke-60, namun Skenario D — yang memiliki n lebih besar — membutuhkan hingga iterasi ke-90. Sebaliknya, Skenario E plateau lebih awal di iterasi ke-40 meski n -nya terbesar. Fenomena ini mencerminkan bahwa durasi konvergensi tidak selalu berkorelasi linier dengan n , melainkan juga dipengaruhi oleh lanskap fitness yang terbentuk dari distribusi spasial koordinat masing-masing skenario. Secara khusus, pada Skenario E ($n=25$), standar deviasi sebesar 9,3260 km mengindikasikan bahwa SSA sensitif terhadap kondisi inisialisasi populasi awal ketika ruang solusi meluas. Mekanisme predator berbasis probabilitas $Pdp=0,1$ dan reinisialisasi parsial melalui *Seasonal Monitoring Condition* (SMC) membantu diversifikasi pencarian dan mencegah konvergensi prematur,

namun pada dimensi $n=25$ kedua mekanisme tersebut belum cukup untuk menekan variabilitas solusi secara signifikan. Hal ini mengindikasikan bahwa peningkatan nilai N dan T , atau penambahan operator perbaikan lokal, diperlukan agar mekanisme eksplorasi SSA dapat bekerja lebih efektif pada skala permasalahan yang lebih besar.

4.3.4 Keterbatasan Implementasi dan Rekomendasi Pengembangan

Berdasarkan hasil dan analisis di atas, teridentifikasi beberapa keterbatasan implementasi yang sekaligus menjadi peluang pengembangan bagi penelitian lanjutan.

Dari aspek parameter, standar deviasi yang meningkat tajam pada skenario besar mengindikasikan $N=30$ dan $T=100$ belum optimal untuk $n \geq 15$. Penelitian lanjutan disarankan untuk melakukan analisis sensitivitas parameter secara sistematis, khususnya memperbesar N ($N=50$ atau $N=100$) dan T ($T=200$ hingga $T=500$) untuk skenario dengan lebih banyak titik.

Dari aspek algoritmik, pengintegrasian operator perbaikan lokal ke dalam SSA berpotensi meningkatkan kualitas solusi tanpa mengubah kerangka dasar algoritma. Hibridasi antara kemampuan eksplorasi global SSA dan perbaikan lokal deterministik merupakan arah penelitian yang menjanjikan (Stamadianos et al., 2024).

Dari aspek data, pengujian pada dataset benchmark standar seperti TSPLIB atau Solomon VRP instances akan memberikan basis perbandingan yang dapat direplikasi dan dibandingkan dengan publikasi penelitian lain. Penggunaan data koordinat riil dari perusahaan logistik akan meningkatkan relevansi praktis temuan penelitian.

Dari aspek evaluasi komparatif, penelitian ini tidak menyertakan perbandingan langsung dengan algoritma lain seperti Genetic Algorithm (GA) karena fokus penelitian ini ditetapkan pada implementasi dan evaluasi SSA sebagai proof-of-concept dalam sistem pengiriman kurir berbasis web, bukan sebagai studi benchmarking algoritma. Selain itu, perbandingan yang valid secara metodologis mensyaratkan bahwa setiap algoritma pembanding dikalibrasi parameternya secara setara, semisal melalui *grid search* atau *design of experiment*, serta diuji pada dataset benchmark yang diakui secara internasional seperti TSPLIB atau Solomon VRP instances, sehingga berada di luar batasan ruang lingkup penelitian. Perbandingan implisit terhadap pendekatan lain telah dilakukan melalui tinjauan pustaka pada Tabel 2. 1, di mana kelemahan GA Adaptif pada stabilitas konvergensi (Gunawan, 2024) dan keterbatasan Simple Hill Climbing pada lokal optima (Malik et al., 2023.) menjadi bagian dari justifikasi teoritis pemilihan SSA. Penelitian lanjutan disarankan untuk melakukan studi perbandingan eksperimental antara SSA, GA, PSO, dan ACO pada dataset benchmark standar agar posisi SSA dalam lanskap algoritma optimasi rute dapat dibuktikan secara empiris.

4.3.5 Relasi Rute Optimal dan Jarak Tempuh dalam Konteks SSA

Dalam implementasi SSA pada penelitian ini, rute dan jarak tidak dapat dipisahkan secara defisional. Jarak total rute merupakan fungsi langsung dari urutan kunjungan yang diwakili oleh rute tersebut, dihitung segmen per segmen menggunakan Formula Haversine. Dengan demikian, untuk setiap urutan kunjungan yang unik terdapat tepat satu nilai total jarak yang unik pula. Tidak ada dua rute yang berbeda urutannya namun memiliki total jarak yang sama, kecuali dalam kasus simetri sempurna yang tidak relevan pada distribusi koordinat perkotaan yang tidak teratur seperti.

4.4 Ringkasan Temuan

Berdasarkan seluruh hasil eksperimen dan pembahasan, penelitian ini menghasilkan empat temuan utama.

Pertama, SSA berhasil diimplementasikan dalam prototipe aplikasi web berbasis Flask dan secara konsisten menghasilkan solusi rute yang valid pada semua 50 eksekusi. Implementasi mekanisme SPV untuk adaptasi SSA ke domain VRP diskret berjalan dengan benar dan menjawab tujuan penelitian pertama.

Kedua, SSA menunjukkan performa paling konsisten pada skenario kecil, dengan standar deviasi 0,0000 km pada Skenario A dan 1,1498 km pada Skenario B. Pada Skenario A, SSA bahkan menemukan solusi identik (10,7489 km) di seluruh 10 replikasi, mengindikasikan tercapainya solusi optimal global.

Ketiga, variabilitas solusi SSA meningkat pada skenario besar, dengan standar deviasi mencapai 9,3260 km pada Skenario E dan selisih run terbaik–terburuk sebesar 26,98 km. Parameter $N=30$ dan $T=100$ terbukti belum optimal untuk $n \geq 20$, dan perlu disesuaikan pada penelitian lanjutan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil eksperimen, analisis data, dan pembahasan yang telah dipaparkan pada bab sebelumnya, penelitian ini menghasilkan kesimpulan sebagai berikut yang menjawab ketiga rumusan masalah yang ditetapkan.

a. Implementasi SSA pada Sistem Pengiriman Kurir Berbasis Web

Squirrel Search Algorithm (SSA) berhasil diimplementasikan secara penuh ke dalam prototipe aplikasi web berbasis Python Flask. Seluruh mekanisme inti SSA, yakni inisialisasi populasi, mekanisme gliding menuju pohon hickory dan acorn, diversifikasi berbasis probabilitas kehadiran predator (Pdp), serta Seasonal Monitoring Condition (SMC), dijalankan secara eksplisit tanpa ketergantungan pada library optimasi eksternal. Adaptasi SSA ke domain VRP diskret menggunakan mekanisme Smallest Position Value (SPV) untuk dekoding posisi kontinu menjadi permutasi rute berjalan dengan benar. Validitas solusi terpenuhi sepenuhnya pada seluruh 50 eksekusi: setiap pelanggan dikunjungi tepat satu kali dan setiap rute dimulai serta diakhiri di depot. Dengan demikian, tujuan penelitian pertama, yaitu mengimplementasikan SSA untuk menyelesaikan VRP pada sistem pengiriman kurir berbasis data simulasi, terpenuhi sepenuhnya.

b. Evaluasi Efektivitas SSA dalam Menyelesaikan VRP

Performa SSA menunjukkan efektivitas yang bergantung pada skala permasalahan. Pada skenario berskala kecil ($n = 5$ dan $n = 10$), SSA memperlihatkan performa yang sangat baik: nilai standar deviasi 0,0000 km pada Skenario A mengkonfirmasi bahwa solusi yang ditemukan bersifat konsisten dan kemungkinan besar merupakan solusi optimal global. Waktu komputasi pada seluruh skenario berada dalam rentang 69,10 ms hingga 226,05 ms, yang bersifat sub-kuadratik terhadap pertambahan n dan memadai untuk kebutuhan aplikasi berbasis web. Dari analisis konvergensi, kurva konvergensi pada semua skenario bersifat monoton non-increasing tanpa osilasi, mengkonfirmasi stabilitas algoritma. Sementara itu, pada skenario berskala besar ($n \geq 15$), variabilitas solusi meningkat signifikan, dengan standar deviasi mencapai 9,3260 km pada Skenario E ($n = 25$). Peningkatan ini mencerminkan keterbatasan kapasitas eksplorasi SSA dengan parameter

$N = 30$ dan $T = 100$ terhadap ruang solusi yang semakin luas, dan bukan merupakan indikasi cacat fundamental pada algoritma.

c. Identifikasi Tantangan, Keterbatasan, dan Rekomendasi Pengembangan

Penelitian ini mengidentifikasi dua keterbatasan utama SSA dalam konteks VRP berskala menengah ke atas. Pertama, kapasitas eksplorasi SSA dengan konfigurasi $N = 30$ dan $T = 100$ belum optimal untuk $n \geq 15$, yang ditandai oleh meningkatnya standar deviasi dan variasi selisih run terbaik-terburuk. Kedua, SSA dalam versi dasarnya tidak dilengkapi dengan operator perbaikan lokal yang dapat memperkuat kualitas solusi di tahap eksploitasi. Temuan ini memberikan arah yang jelas bagi pengembangan lanjutan, yakni analisis sensitivitas parameter dan hibridasi SSA dengan metode perbaikan lokal, serta pengujian pada dataset benchmark standar guna mendapatkan basis perbandingan yang dapat direplikasi.

5.2 Saran

Berdasarkan kesimpulan dan temuan keterbatasan yang diidentifikasi, berikut disampaikan saran untuk penelitian dan pengembangan lanjutan.

a. Analisis Sensitivitas dan Optimasi Parameter SSA

Penelitian lanjutan disarankan untuk melakukan analisis sensitivitas parameter SSA secara sistematis, khususnya memperbesar ukuran populasi ($N = 50$ hingga $N = 100$) dan jumlah iterasi ($T = 200$ hingga $T = 500$) pada skenario dengan jumlah titik $n \geq 15$. Penentuan kombinasi parameter optimal dapat dilakukan menggunakan pendekatan Design of Experiment (DoE) atau grid search, sehingga konfigurasi yang digunakan terbukti sesuai secara empiris untuk berbagai skala permasalahan VRP.

b. Hibridasi SSA dengan Operator Perbaikan Lokal

Untuk meningkatkan kualitas solusi pada ruang pencarian berdimensi tinggi, disarankan mengintegrasikan operator perbaikan lokal ke dalam SSA, seperti 2-opt atau 3-opt yang umum digunakan dalam literatur VRP. Hibridasi ini berpotensi memanfaatkan kemampuan eksplorasi global SSA sekaligus memperkuat tahap eksploitasi lokal, tanpa mengubah kerangka dasar algoritma yang telah divalidasi pada penelitian ini.

c. Pengujian pada Dataset Benchmark Standar

Penelitian ini menggunakan data koordinat simulasi yang bersifat internal. Untuk memperoleh basis perbandingan yang dapat direplikasi dan dibandingkan dengan publikasi ilmiah lain, penelitian lanjutan disarankan menggunakan dataset benchmark standar seperti TSPLIB atau Solomon VRP instances. Pengujian pada dataset yang diakui secara

internasional akan meningkatkan kredibilitas akademis hasil penelitian dan memudahkan reproduksi oleh peneliti lain.

d. Pengembangan Varian VRP yang Lebih Kompleks

Penelitian ini menggunakan model VRP standar tanpa batasan kapasitas kendaraan (Capacitated VRP) atau batasan waktu pengiriman (*VRP with Time Windows*/VRPTW). Penambahan kendala-kendala tersebut akan memperkaya relevansi praktis penelitian dan mendekatkan simulasi pada kondisi operasional kurir yang sesungguhnya. Implementasi VRPTW khususnya memiliki potensi aplikasi yang signifikan dalam konteks pengiriman e-commerce yang terikat pada jendela waktu pengiriman.

e. Integrasi dengan Data Jaringan Jalan Nyata

Prototipe yang dibangun pada penelitian ini menggunakan jarak geodesik Haversine yang mengasumsikan lintasan garis lurus antar titik. Untuk meningkatkan realisme aplikasi, penelitian lanjutan disarankan mengintegrasikan data jaringan jalan nyata menggunakan API routing seperti OpenStreetMap (OSRM) atau Google Maps Directions API. Integrasi ini akan menghasilkan estimasi jarak dan waktu tempuh yang lebih akurat sesuai kondisi infrastruktur jalan di lapangan, sehingga rekomendasi rute yang dihasilkan dapat langsung digunakan oleh kurir dalam kondisi operasional nyata.

DAFTAR PUSTAKA

- Dhaini, M., & Mansour, N. (2021). Squirrel search algorithm for portfolio optimization. *Expert Systems with Applications*, 178, 114968. <https://doi.org/10.1016/j.eswa.2021.114968>
- Fatma, E., Kartika, W., & Nur Madyanti, A. (2022). Penentuan Rute Pengangkut Limbah Medis Optimal Menggunakan Vehicle Routing Problem with Time Window pada Kasus Multi Depot. *Jurnal Manajemen dan Organisasi*, 13(4), 324–335. <https://doi.org/10.29244/jmo.v13i4.38587>
- Febriani, S. (2022). *Analisis Deskriptif Standar Deviasi*. 6.
- Ferdiansyah, A., Sholihah, S. A., Rifni, M., Grets, E. S., Situmorang, J. K., & Oktaviany, I. (n.d.). *ANALISIS PERENCANAAN RUTE PENGIRIMAN BARANG MENGGUNAKAN METODE VEHICLE ROUTING PROBLEM (VRP)*.
- Gunawan, C. R. (2024). Optimization of the Travelling Salesman Problem Based on Genetic Algorithm with Adaptive Crossover and Mutation Probabilitie. *Jurnal Ilmu Komputer dan Sistem Informasi*, 3(3), 234–242. <https://doi.org/10.70340/jirsi.v3i3.166>
- Jackye Jackson James & Andung Jati Nugroho. (2024). Penyelesaian Vehicles Routing Problem dalam Meminimumkan Waktu Transportasi PT Petrogas Prima Services. *Sci-tech Journal*, 3(1), 60–73. <https://doi.org/10.56709/stj.v3i1.326>
- Jain, M., Singh, V., & Rani, A. (2019). A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm and Evolutionary Computation*, 44, 148–175. <https://doi.org/10.1016/j.swevo.2018.02.013>
- Juan, A. A., Rabe, M., Ammouriova, M., Panadero, J., Peidro, D., & Riera, D. (2023). Solving NP-Hard Challenges in Logistics and Transportation under General Uncertainty Scenarios Using Fuzzy Simheuristics. *Algorithms*, 16(12), 570. <https://doi.org/10.3390/a16120570>

- Mahmudy, W. F., Widodo, A. W., & Haikal, A. H. (2024). Challenges and Opportunities for Applying Meta-Heuristic Methods in Vehicle Routing Problems: A Review. *The 7th Mechanical Engineering, Science and Technology International Conference*, 12. <https://doi.org/10.3390/engproc2024063012>
- Malik, A., Bantun, S., & Sari, J. Y. (n.d.). *OPTIMALISASI RUTE PENGIRIMAN UNTUK E-COMMERCE: APLIKASI KURIR BERBASIS WEB MENGGUNAKAN ALGORITMA SIMPLE HILL CLIMBING*.
- Martias, L. D. (2021). STATISTIKA DESKRIPTIF SEBAGAI KUMPULAN INFORMASI. *Fihris: Jurnal Ilmu Perpustakaan dan Informasi*, 16(1), 40. <https://doi.org/10.14421/fhrs.2021.161.40-59>
- MMCS Team, LMAID Laboratory, ENSMR-Rabat, Khayya, E., Medarhri, I., MMCS Team, LMAID Laboratory, ENSMR-Rabat, Zine, R., & School of Science and Engineering, Al Akhawayn University in Ifrane. (2024). A survey of the vehicle routing problem and its variants: Formulations and solutions. *Mathematical Modeling and Computing*, 11(1), 333–343. <https://doi.org/10.23939/mmc2024.01.333>
- Muttaqin, A., Murtopo, A. A., Syefudin, S., & Gunawan, G. (2024). *Application of the haversine formula method to determine the closest distance to a minimarket*. 13(1).
- Ningtyas, D. F., & Setiyawati, N. (2021). Implementasi Flask Framework pada Pembangunan Aplikasi Purchasing Approval Request. *Jurnal Janitra Informatika dan Sistem Informasi*, 1(1), 19–34. <https://doi.org/10.25008/janitra.v1i1.120>
- Puspita, A., Yuningsih, Y., Mazia, L., & Pujiastuti, E. (2024). Penerapan Metode Prototype Pada Perancangan Sistem Informasi Penjualan Aksesoris Kamera Berbasis Web. *Infotek: Jurnal Informatika dan Teknologi*, 7(1), 160–167. <https://doi.org/10.29408/jit.v7i1.24100>

- Stamadianos, T., Taxidou, A., Marinaki, M., & Marinakis, Y. (2024). Swarm intelligence and nature inspired algorithms for solving vehicle routing problems: A survey. *Operational Research*, 24(3), 47. <https://doi.org/10.1007/s12351-024-00862-5>
- Waruwu, M. (2023). *Pendekatan Penelitian Pendidikan: Metode Penelitian Kualitatif, Metode Penelitian Kuantitatif dan Metode Penelitian Kombinasi*. 7.
- Wijaya, O. (2023). E-Commerce: Perkembangan, Tren, dan Peraturan Perundang-Undangan. *E-Bisnis : Jurnal Ilmiah Ekonomi dan Bisnis*, 16(1), 41–47. <https://doi.org/10.51903/e-bisnis.v16i1.1083>
- Wijayanto, C., & Susetyo, Y. A. (2022). Implementasi Flask Framework Pada Pembangunan Aplikasi Sistem Informasi Helpdesk (SIH). *JIPi (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, 7(3), 858–868. <https://doi.org/10.29100/jipi.v7i3.3161>

LAMPIRAN

Lampiran A

```
ssa.py × requirements.txt
ssa.py > squirrel_search_algorithm

1 import math
2 import random
3 import time
4
5
6 # — Haversine Distance —————
7
8 def haversine(coord1, coord2):
9     R = 6371 # Radius rata-rata bumi (km)
10    lat1, lon1 = math.radians(coord1[0]), math.radians(coord1[1])
11    lat2, lon2 = math.radians(coord2[0]), math.radians(coord2[1])
12
13    dlat = lat2 - lat1
14    dlon = lon2 - lon1
15
16    a = math.sin(dlat / 2) ** 2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon / 2) ** 2
17    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
18    return R * c
19
20
21
22 def total_route_distance(route, coords):
23     dist = 0.0
24     prev = 0 # depot
25     for idx in route:
26         dist += haversine(coords[prev], coords[idx + 1])
27         prev = idx + 1
28     dist += haversine(coords[prev], coords[0]) # kembali ke depot
29     return dist
30
31
32
33 # — Smallest Position Value (SPV) Decoder —————
34
35 def spv_decode(position):
36     return [i for i, _ in sorted(enumerate(position), key=lambda x: x[1])]
37
38
39
40 # — Squirrel Search Algorithm —————
41
42 def squirrel_search_algorithm(coords, params=None):
43     if params is None:
44         params = {}
45
```

```

46
47     N = params.get("N", 30)
48     T = params.get("T", 100)
49     gc = params.get("gc", 1.9)
50     Pdp = params.get("Pdp", 0.1)
51     eps = params.get("eps", 1e-6)
52     na = params.get("na", 3)
53
54     n_customers = len(coords) - 1 # jumlah pelanggan (tidak termasuk depot)
55
56     start_time = time.time()
57     convergence = []
58
59     # — Inisialisasi populasi —————
60     # Setiap tupai = vektor posisi kontinu berdimensi n_customers
61     population = [[random.uniform(0, n_customers) for _ in range(n_customers)]
62                  | for _ in range(N)]
63
64     # Evaluasi fitness awal
65     Windsurf: Refactor | Explain | Generate Docstring | X
66     def fitness(pos):
67         route = spv_decode(pos)
68         return total_route_distance(route, coords)
69
70     fitnesses = [fitness(p) for p in population]
71
72     # Urutkan: indeks fitness terbaik (terkecil) → pohon acorn (hickory)
73     sorted_idx = sorted(range(N), key=lambda i: fitnesses[i])
74     acorn_trees = sorted_idx[:na] # na pohon acorn terbaik
75     hickory_tree = sorted_idx[0] # pohon hickory (solusi global terbaik)
76
77     best_pos = population[hickory_tree][:]
78     best_fitness = fitnesses[hickory_tree]
79     prev_best = best_fitness
80
81     # — Iterasi utama —————
82     for t in range(1, T + 1):
83         for i in range(N):
84             if i == hickory_tree:
85                 continue # pohon hickory tidak bergerak
86
87             r = random.random()
88
89             if i in acorn_trees:
90                 # Perpindahan pohon acorn → hickory (eksploitasi)
91                 Dg = haversine(
92                     [population[i][0], population[i][1] if n_customers > 1 else 0],

```

```

90     Dg = haversine(
91         [population[i][0], population[i][1] if n_customers > 1 else 0],
92         [population[hickory_tree][0], population[hickory_tree][1] if n_customers > 1 else 0]
93     ) if n_customers >= 2 else random.uniform(0.5, 1.5)
94
95     new_pos = [
96         population[i][j] + gc * Dg * (population[hickory_tree][j] - population[i][j])
97         for j in range(n_customers)
98     ]
99     else:
100         if r > Pdp:
101             # Perpindahan ke pohon acorn terdekat (eksplorasi lokal)
102             target = random.choice(acorn_trees)
103             Dg = random.uniform(0.5, 1.5)
104             new_pos = [
105                 population[i][j] + gc * Dg * (population[target][j] - population[i][j])
106                 for j in range(n_customers)
107             ]
108         else:
109             # Perpindahan acak: kehadiran predator (diversifikasi)
110             new_pos = [random.uniform(0, n_customers) for _ in range(n_customers)]
111
112     # Clamp ke range valid
113     new_pos = [max(0.0, min(float(n_customers - 1 + 0.999), v)) for v in new_pos]
114
115     new_fit = fitness(new_pos)
116     if new_fit < fitnesses[i]:
117         population[i] = new_pos
118         fitnesses[i] = new_fit
119
120     # Update hierarki pohon
121     sorted_idx = sorted(range(N), key=lambda i: fitnesses[i])
122     acorn_trees = sorted_idx[:na]
123     hickory_tree = sorted_idx[0]
124
125     if fitnesses[hickory_tree] < best_fitness:
126         best_fitness = fitnesses[hickory_tree]
127         best_pos = population[hickory_tree][:]
128
129     # Seasonal Monitoring Condition (SMC)
130     if abs(prev_best - best_fitness) < eps and t > 10:
131         # Reinisialisasi sebagian populasi untuk hindari premature convergence
132         for k in sorted_idx[na:na + max(1, N // 5)]:
133             population[k] = [random.uniform(0, n_customers) for _ in range(n_customers)]
134             fitnesses[k] = fitness(population[k])
135

```

```

136     prev_best = best_fitness
137     convergence.append(round(best_fitness, 6))
138
139     elapsed_ms = (time.time() - start_time) * 1000
140
141     best_route = spv_decode(best_pos)
142
143     # Bangun detail rute: depot → pelanggan → depot
144     route_detail = []
145     prev_idx = 0
146     cumulative = 0.0
147     for cust_idx in best_route:
148         real_idx = cust_idx + 1
149         d = haversine(coords[prev_idx], coords[real_idx])
150         cumulative += d
151         route_detail.append({
152             "from": prev_idx,
153             "to": real_idx,
154             "dist": round(d, 4)
155         })
156         prev_idx = real_idx
157     # Kembali ke depot
158     d = haversine(coords[prev_idx], coords[0])
159     cumulative += d
160     route_detail.append({
161         "from": prev_idx,
162         "to": 0,
163         "dist": round(d, 4)
164     })
165
166     return {
167         "best_route": best_route,
168         "route_detail": route_detail,
169         "total_distance": round(best_fitness, 4),
170         "execution_time_ms": round(elapsed_ms, 2),
171         "convergence": convergence,
172         "params": {"N": N, "T": T, "gc": gc, "Pdp": Pdp, "na": na},
173         "n_customers": n_customers,
174     }

```

```

index.html X
index.html > html > head > style > Ad.stbl.jh

1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4 <meta charset="UTF-8"/>
5 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
6 <title>Eksperimen SSA</title>
7 <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"/>
8 <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
9 <style>
10 * { box-sizing: border-box; margin: 0; padding: 0; }
11 body { font-family: Arial, sans-serif; font-size: 13px; background: #f0f2f5; height: 100vh; overflow: hidden; }
12 #app { display: flex; height: 100vh; }
13
14 /* --- PANEL --- */
15 #panel {
16   width: 230px; min-width: 230px;
17   background: #fff; border-right: 1px solid #ddd;
18   display: flex; flex-direction: column;
19 }
20 #panel-header { padding: 10px 12px; background: #1a56db; color: #fff; }
21 #panel-header h1 { font-size: 12px; font-weight: 700; }
22 #panel-header p { font-size: 10px; opacity: .8; margin-top: 1px; }
23 #panel-body { padding: 10px 12px; display: flex; flex-direction: column; gap: 12px; }
24
25 .lbl { font-size: 10px; font-weight: 700; color: #666; text-transform: uppercase; letter-spacing: .04em; margin-bottom: 5px; }
26
27 /* template */
28 .tpl-grid { display: grid; grid-template-columns: repeat(5,1fr); gap: 4px; }
29 .btn-tpl {
30   padding: 5px 2px; font-size: 11px; font-weight: 700;
31   border: 1px solid #1a56db; background: #fff; color: #1a56db;
32   border-radius: 4px; cursor: pointer; text-align: center; line-height: 1.3;
33 }
34 .btn-tpl span { display: block; font-size: 9px; color: #999; font-weight: 400; }
35 .btn-tpl:hover { background: #e8f0fd; }
36 .btn-tpl.active { background: #1a56db; color: #fff; }
37 .btn-tpl.active span { color: #b3caff; }
38
39 /* titik */
40 #coord-list { max-height: 140px; overflow-y: auto; border: 1px solid #eee; border-radius: 4px; }
41 .ci { display: flex; align-items: center; gap: 5px; padding: 3px 6px; border-bottom: 1px solid #f5f5f5; font-size: 10px; color: #444; }
42 .ci:last-child { border-bottom: none; }
43 .badge { width: 16px; height: 16px; border-radius: 50%; font-size: 8px; font-weight: 700; color: #fff; display: flex; align-items: center; justify-content: center; flex-shrink: 0; }
44 .bd { background: #e74c3c; } .bc { background: #2980b9; }
45 .del { margin-left: auto; background: none; border: none; color: #ccc; cursor: pointer; font-size: 12px; line-height: 1; }
46 .del:hover { color: #e74c3c; }
47 #empty { padding: 8px; text-align: center; color: #bbb; font-size: 10px; }
48 #cnt { font-size: 10px; color: #888; margin-bottom: 4px; }

50 /* aksi */
51 .grp { display: flex; flex-direction: column; gap: 5px; }
52 .brun { width: 100%; padding: 8px; background: #1a56db; color: #fff; border: none; border-radius: 5px; font-size: 12px; font-weight: 700; cursor: pointer; }
53 .brun:hover:not(:disabled) { background: #1648c0; }
54 .brun:disabled { background: #aaa; cursor: not-allowed; }
55 .bsec { width: 100%; padding: 7px; background: #fff; color: #1a56db; border: 1px solid #1a56db; border-radius: 5px; font-size: 11px; font-weight: 600; cursor: pointer; }
56 .bsec:hover { background: #e8f0fd; }
57 .bclr { width: 100%; padding: 6px; background: #fff; color: #c0392b; border: 1px solid #e0b0ab; border-radius: 5px; font-size: 11px; cursor: pointer; }
58 .bclr:hover { background: #fdf0ee; }
59
60 /* hasil */
61 #rbox { display: none; }
62 .rcards { display: grid; grid-template-columns: 1fr 1fr; gap: 6px; margin-bottom: 8px; }
63 .rcard { background: #f0f4ff; border: 1px solid #c5d5f0; border-radius: 5px; padding: 7px; text-align: center; }
64 .rval { font-size: 14px; font-weight: 700; color: #1a56db; }
65 .rlbl { font-size: 9px; color: #777; margin-top: 1px; }
66 .stbl { width: 100%; border-collapse: collapse; font-size: 10px; }
67 .stbl th { background: #f0f4ff; padding: 3px 5px; text-align: left; color: #555; border: 1px solid #dde4f5; font-weight: 600; }
68 .stbl td { padding: 3px 5px; border: 1px solid #dde4f5; text-align: right; }
69 .stbl td:first-child { text-align: left; color: #666; }
70 .stbl .hi { color: #1a56db; font-weight: 700; }
71
72 /* peta */
73 #mwrap { flex: 1; position: relative; }
74 #map { width: 100%; height: 100%; }
75 #mbadge {
76   position: absolute; top: 10px; left: 10px; z-index: 100;
77   background: rgba(255, 255, 255, .93); border: 1px solid #ddd; border-radius: 5px;
78   padding: 6px 10px; font-size: 11px; line-height: 1.8; box-shadow: 0 1px 4px rgba(0, 0, 0, .1);
79 }
80 #mbadge b { color: #1a56db; }
81 #loading {
82   display: none; position: absolute; inset: 0; background: rgba(255, 255, 255, .65);
83   z-index: 200; align-items: center; justify-content: center; flex-direction: column; gap: 8px;
84 }
85 #loading.on { display: flex; }
86 .spin { width: 28px; height: 28px; border: 3px solid #ddd; border-top-color: #1a56db; border-radius: 50%; animation: sp .7s linear infinite; }
87 @keyframes sp { to { transform: rotate(360deg); } }
88 #ltx { font-size: 11px; font-weight: 600; color: #1a56db; }

```

```

90  /* toast */
91  #toast { position:fixed;bottom:14px;right:14px;padding:7px 13px;border-radius:4px;font-size:11px;z-index:999;opacity:0;transition:opacity .3s;pointer-events:none;background:#222;color:#fff; }
92  #toast.on { opacity:1; }
93  #toast.ok { background:#1a7a4a; }
94  #toast.err { background:#c0392b; }
95  </style>
96  </head>
97  <body>
98  <div id="app">
99
100    <div id="panel">
101      <div id="panel-header">
102        <h1>🐿 Eksperimen SSA</h1>
103        <p>Squirrel Search Algorithm • VRP</p>
104      </div>
105
106      <div id="panel-body">
107
108        <div>
109          <div class="lbl">Skenario Penelitian</div>
110          <div class="tpl-grid">
111            <button class="btn-tpl" id="tA" onclick="load('A')">A<span>n=5</span></button>
112            <button class="btn-tpl" id="tB" onclick="load('B')">B<span>n=10</span></button>
113            <button class="btn-tpl" id="tC" onclick="load('C')">C<span>n=15</span></button>
114            <button class="btn-tpl" id="tD" onclick="load('D')">D<span>n=20</span></button>
115            <button class="btn-tpl" id="tE" onclick="load('E')">E<span>n=25</span></button>
116          </div>
117        </div>
118
119        <div>
120          <div class="lbl">Titik - <span id="cnt">0</span> titik</div>
121          <div id="coord-list"><div id="empty">Pilih skenario atau klik peta.</div></div>
122        </div>
123
124        <div class="grp">
125          <button class="brun" id="btn-run" onclick="run(false)" disabled>▶ Jalankan SSA</button>
126          <button class="bsec" onclick="run(true)">🔄 Batch 10 Replikasi</button>
127          <button class="bclr" onclick="clearAll()">✕ Hapus Semua</button>
128        </div>
129
130        <div id="rbox">
131          <div class="lbl" id="rtitle">Hasil</div>
132          <div class="rcards">
133            <div class="rcard"><div class="rval" id="rdist"></div><div class="rlbl">JARAK (KM)</div></div>
134            <div class="rcard"><div class="rval" id="rtime"></div><div class="rlbl">WAKTU (MS)</div></div>

```

```

135      </div>
136      <div id="swrap" style="display:none">
137        <table class="stbl">
138          <thead><tr><th>Metrik</th><th>Mean</th><th>Min</th><th>Max</th><th>StDev</th></tr></thead>
139          <tbody id="sbody"></tbody>
140        </table>
141      </div>
142    </div>
143  </div>
144  </div>
145  </div>
146
147  <div id="mmap">
148    <div id="map"></div>
149    <div id="mbadge">
150      Titik: <b id="opts">0</b> &nbsp;&nbsp;&nbsp; Skenario: <b id="oscen"></b> &nbsp;&nbsp;&nbsp; Jarak: <b id="odist"></b>
151    </div>
152    <div id="loading"><div class="spin"></div><div id="ltxt">Menjalankan SSA...</div></div>
153  </div>
154
155  </div>
156  <div id="toast"></div>
157
158  <script>
159  let coords=[], markers=[], route=null, scen='-';
160
161  const map = L.map('map').setView([-7.797,110.370],13);
162  L.tilelayer("https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",{attribution:'© OpenStreetMap',maxZoom:19}).addTo(map);
163  map.on('click',e=>add(e.latlng.lat,e.latlng.lng));
164
165  Windsurf: Refactor | Explain | Generate Function Comment | ✕
166  function icon(l,c){return L.divIcon({className:'',html:`<div style="width:20px;height:20px;background:${c};border-radius:50%;border:2px solid #fff;display:
167  flex;align-items:center;justify-content:center;font-size:8px;font-weight:700;color:#fff;box-shadow:0 1px 4px rgba(0,0,0,.3)">${l}</div>`,iconSize:[20,20],iconAnchor:[10,10]});}
168
169  Windsurf: Refactor | Explain | Generate Function Comment | ✕
170  function add(lat,lon){
171    coords.push([lat,lon]);
172    const i=coords.length-1, d=i==0;
173    markers.push(L.marker([lat,lon],{icon:icon(d?'D':'I,d?'#74c3c':'#2980b9')})).addTo(map));
174    render(); sync();
175    document.getElementById('btn-run').disabled=coords.length<3;
176  }

```

```

176 function del(idx){
177   const c2=coords.filter((_,i)>>i!==(idx));
178   clearM(); coords=[];
179   c2.forEach(c=>add(c[0],c[1])); clearR();
180 }
181
182 function clearM(){ markers.forEach(m=>map.removeLayer(m)); markers=[]; }
183 function clearR(){ if(route){map.removeLayer(route);route=null;} }
184
Windsurf Refactor | Explain | Generate Function Comment | X
185 function clearAll(){
186   clearM(); clearR(); coords=[]; scen='';
187   render(); sync();
188   document.getElementById('btn-run').disabled=true;
189   document.getElementById('rbox').style.display='none';
190   document.querySelectorAll('.btn-tpl').forEach(b=>b.classList.remove('active'));
191 }
192
Windsurf Refactor | Explain | Generate Function Comment | X
193 function render(){
194   const el=document.getElementById('coord-list');
195   document.getElementById('cnt').textContent=coords.length;
196   if(!coords.length){el.innerHTML='<div id="empty">Pilih skenario atau klik peta.</div>';return;}
197   el.innerHTML=coords.map((c,i)>>{
198     const d=i==0;
199     return`<div class="ci"><div class="badge ${d?'bd':'bc'}">${d?'D':'i'}</div><span>${d?'Depot':'P'+i}<span style="color:#bbb">${c[0].toFixed(4)},${c[1].toFixed(4)}</span></span><button class="del" on
200   }).join(' ');
201 }
202
Windsurf Refactor | Explain | Generate Function Comment | X
203 function sync(){
204   document.getElementById('opts').textContent=coords.length;
205   document.getElementById('oscen').textContent=scen;
206 }
207
208 const T={
209   A:[n:5, c:[[-7.7956,110.3695],[-7.8014,110.3647],[-7.7900,110.3750],[-7.8100,110.3800],[-7.7800,110.3600],[-7.8050,110.3550]],
210   B:[n:10, c:[[-7.7956,110.3695],[-7.8014,110.3647],[-7.7900,110.3750],[-7.8100,110.3800],[-7.7800,110.3600],[-7.8050,110.3550],
211     [-7.7750,110.3900],[-7.8200,110.3700],[-7.7850,110.3500],[-7.8150,110.3850],[-7.7700,110.3700]]],
212   C:[n:15, c:[[-7.7956,110.3695],[-7.8014,110.3647],[-7.7900,110.3750],[-7.8100,110.3800],[-7.7800,110.3600],[-7.8050,110.3550],
213     [-7.7750,110.3900],[-7.8200,110.3700],[-7.7850,110.3500],[-7.8150,110.3850],[-7.7700,110.3700],[-7.8250,110.3600],[-7.7650,110.3650],[-7.8300,110.3750],[-7.7600,110.3820],[-7.8350,110.3900]]],
214   D:[n:20, c:[[-7.7956,110.3695],[-7.8014,110.3647],[-7.7900,110.3750],[-7.8100,110.3800],[-7.7800,110.3600],[-7.8050,110.3550],
215     [-7.8050,110.3550],[-7.7750,110.3900],[-7.8200,110.3700],[-7.7850,110.3500],[-7.8150,110.3850],[-7.7700,110.3700],
216     [-7.8250,110.3600],[-7.7650,110.3650],[-7.8300,110.3750],[-7.7600,110.3820],[-7.8350,110.3900],[-7.7550,110.3700],[-7.8400,110.3550],[-7.7500,110.3850],[-7.8450,110.3800],[-7.7480,110.3600]]],
217
218   E:[n:25, c:[[-7.7956,110.3695],[-7.8014,110.3647],[-7.7900,110.3750],[-7.8100,110.3800],[-7.7800,110.3600],
219     [-7.8050,110.3550],[-7.7750,110.3900],[-7.8200,110.3700],[-7.7850,110.3500],[-7.8150,110.3850],[-7.7700,110.3700],
220     [-7.8250,110.3600],[-7.7650,110.3650],[-7.8300,110.3750],[-7.7600,110.3820],[-7.8350,110.3900],[-7.7550,110.3700],[-7.8400,110.3550],[-7.7500,110.3850],[-7.8450,110.3800],[-7.7480,110.3600],
221     [-7.8500,110.3650],[-7.7430,110.3900],[-7.8550,110.3900],[-7.7380,110.3750],[-7.8600,110.3750]]],
222   };
223
Windsurf Refactor | Explain | Generate Function Comment | X
223 function load(s){
224   clearAll();
225   T[s].c.forEach(c=>add(c[0],c[1]));
226   scen=`${s} (n=${T[s].n})`; sync();
227   map.fitBounds(markers.map(m=>m.getLatLng()));
228   document.getElementById('t'+s).classList.add('active');
229   toast(`Skenario ${s} dimuat, 'ok');
230 }
231
Windsurf Refactor | Explain | Generate Function Comment | X
232 async function run(batch=false){
233   if(coords.length<3){toast('Minimal 2 pelanggan + 1 depot.','err');return;}
234   loading(true,batch?'Batch 100 replikasi...':'Menjalankan SSA...');
235   try{
236     const ep=batch?'/api/batch':'/api/optimize';
237     const body={coords:params:{N:30,T:100,gc:1.9,Pdp:0.1,na:3,eps:0.000001}};
238     if(batch)body.n_runs=10;
239     const r=await fetch(ep,{method:'POST',headers:{'Content-Type':'application/json'},body:JSON.stringify(body)});
240     const d=await r.json();
241     if(d.error)throw new Error(d.error);
242     batch?showBatch(d):showSingle(d.result);
243     toast('Selesai /','ok');
244   }catch(e){toast('Error: '+e.message,'err');}
245   finally{loading(false);}
246 }
247
Windsurf Refactor | Explain | Generate Function Comment | X
248 function showSingle(r){
249   document.getElementById('rbox').style.display='block';
250   document.getElementById('rtitle').textContent='Hasil SSA';
251   document.getElementById('rdist').textContent=r.total_distance+' km';
252   document.getElementById('rtime').textContent=r.execution_time_ms+' ms';
253   document.getElementById('swrap').style.display='none';
254   document.getElementById('odist').textContent=r.total_distance+' km';
255   drawRoute(r);
256 }

```

```

Windsurf Refactor | Explain | Generate Function Comment | X
258 function showBatch(data){
259   const{stats,best_run}=data;
260   document.getElementById('rbox').style.display='block';
261   document.getElementById('rtile').textContent='Hasil Batch (100)';
262   document.getElementById('rdist').textContent=stats.distance.mean+' km';
263   document.getElementById('rtime').textContent=stats.time_ms.mean+' ms';
264   document.getElementById('odist').textContent=stats.distance.mean+' km';
265   document.getElementById('swrap').style.display='block';
266   document.getElementById('tbody').innerHTML=
267     <tr><td>Jarak (km)</td><td class="hi">${stats.distance.mean}</td><td>${stats.distance.min}</td><td>${stats.distance.max}</td><td>${stats.distance.stdev}</td></tr>
268     <tr><td>Waktu (ms)</td><td class="hi">${stats.time_ms.mean}</td><td>${stats.time_ms.min}</td><td>${stats.time_ms.max}</td><td>${stats.time_ms.stdev}</td></tr>;
269   drawRoute(best_run);
270 }
271
Windsurf Refactor | Explain | Generate Function Comment | X
272 function drawRoute(r){
273   clearR();
274   const pts=[0,...r.best_route.map(i=>i+1),0].map(i=>L.latlng(coords[i][0],coords[i][1]));
275   route=L.polyline(pts,{color:'#1a56db',weight:3,opacity:.8}).addTo(map);
276   map.fitBounds(route.getBounds(),{padding:[30,30]});
277 }
278
Windsurf Refactor | Explain | Generate Function Comment | X
279 function loading(on,msg=''){
280   document.getElementById('loading').className=on?'on':'';
281   document.getElementById('ltxt').textContent=msg;
282 }
283
284 let _t;
Windsurf Refactor | Explain | Generate Function Comment | X
285 function toast(msg,type=''){
286   const el=document.getElementById('toast');
287   el.textContent=msg; el.className='on '+ (type|| '');
288   clearTimeout(_t); _t=setTimeout(()=>el.className='',2500);
289 }
290
291 render();
292 </script>
293 </body>
294 </html>

```

app.py X

app.py > batch_optimize

```
1 from flask import Flask, render_template, request, jsonify
2 from ssa import squirrel_search_algorithm
3 import statistics
4
5 app = Flask(__name__)
6
7
8 @app.route("/")
9 def index():
10     return render_template("index.html")
11
12
13 @app.route("/api/optimize", methods=["POST"])
14 def optimize():
15
16     data = request.get_json()
17     if not data or "coords" not in data:
18         return jsonify({"error": "Parameter 'coords' diperlukan."}), 400
19
20     coords = data["coords"]
21     if len(coords) < 3:
22         return jsonify({"error": "Minimal 2 pelanggan + 1 depot (3 koordinat)."}), 400
23
24     params = data.get("params", {})
25
26     try:
27         result = squirrel_search_algorithm(coords, params)
28         return jsonify({"status": "ok", "result": result})
29     except Exception as e:
30         return jsonify({"error": str(e)}), 500
31
32
33 @app.route("/api/batch", methods=["POST"])
34 def batch_optimize():
35
36     data = request.get_json()
37     if not data or "coords" not in data:
38         return jsonify({"error": "Parameter 'coords' diperlukan."}), 400
39
40     coords = data["coords"]
41     params = data.get("params", {})
42     n_runs = int(data.get("n_runs", 10))
43
44     distances = []
45     times = []
46     best_run = None
```

```

44     distances = []
45     times     = []
46     best_run  = None
47
48     for _ in range(n_runs):
49         r = squirrel_search_algorithm(coords, params)
50         distances.append(r["total_distance"])
51         times.append(r["execution_time_ms"])
52         if best_run is None or r["total_distance"] < best_run["total_distance"]:
53             best_run = r
54
55         Windsurf: Refactor | Explain | Generate Docstring | X
56     def safe_stdev(lst):
57         return round(statistics.stdev(lst), 4) if len(lst) > 1 else 0.0
58
59     stats = {
60         "distance": {
61             "mean": round(statistics.mean(distances), 4),
62             "min": round(min(distances), 4),
63             "max": round(max(distances), 4),
64             "stdev": safe_stdev(distances),
65         },
66         "time_ms": {
67             "mean": round(statistics.mean(times), 2),
68             "min": round(min(times), 2),
69             "max": round(max(times), 2),
70             "stdev": safe_stdev(times),
71         },
72         "n_runs": n_runs,
73         "n_customers": len(coords) - 1,
74     }
75
76     return jsonify({"status": "ok", "stats": stats, "best_run": best_run})
77
78 if __name__ == "__main__":
79     app.run(debug=True, port=5000)
80

```

Lampiran B

Tangkapan layar diambil pada eksekusi terpisah dari data pengujian utama. Perbedaan angka merupakan hal yang wajar karena SSA menggunakan inisialisasi populasi acak pada setiap eksekusi.

Skenario A (n=5)

10	"convergence": [	61	10.748928,
11	10.748928,	62	10.748928,
12	10.748928,	63	10.748928,
13	10.748928,	64	10.748928,
14	10.748928,	65	10.748928,
15	10.748928,	66	10.748928,
16	10.748928,	67	10.748928,
17	10.748928,	68	10.748928,
18	10.748928,	69	10.748928,
19	10.748928,	70	10.748928,
20	10.748928,	71	10.748928,
21	10.748928,	72	10.748928,
22	10.748928,	73	10.748928,
23	10.748928,	74	10.748928,
24	10.748928,	75	10.748928,
25	10.748928,	76	10.748928,
26	10.748928,	77	10.748928,
27	10.748928,	78	10.748928,
28	10.748928,	79	10.748928,
29	10.748928,	80	10.748928,
30	10.748928,	81	10.748928,
31	10.748928,	82	10.748928,
32	10.748928,	83	10.748928,
33	10.748928,	84	10.748928,
34	10.748928,	85	10.748928,
35	10.748928,	86	10.748928,
36	10.748928,	87	10.748928,
37	10.748928,	88	10.748928,
38	10.748928,	89	10.748928,
39	10.748928,	90	10.748928,
40	10.748928,	91	10.748928,
41	10.748928,	92	10.748928,
42	10.748928,	93	10.748928,
43	10.748928,	94	10.748928,
44	10.748928,	95	10.748928,
45	10.748928,	96	10.748928,
46	10.748928,	97	10.748928,
47	10.748928,	98	10.748928,
48	10.748928,	99	10.748928,
49	10.748928,	100	10.748928,
50	10.748928,	101	10.748928,
51	10.748928,	102	10.748928,
52	10.748928,	103	10.748928,
53	10.748928,	104	10.748928,
54	10.748928,	105	10.748928,
55	10.748928,	106	10.748928,
56	10.748928,	107	10.748928,
57	10.748928,	108	10.748928,
58	10.748928,	109	10.748928,
59	10.748928,	110	10.748928

Skenario B (n=10)

15 ▼	"convergence": [	66	19.941678,
16	24.421227,	67	19.941678,
17	21.382467,	68	19.941678,
18	19.941678,	69	19.941678,
19	19.941678,	70	19.941678,
20	19.941678,	71	19.941678,
21	19.941678,	72	19.941678,
22	19.941678,	73	19.941678,
23	19.941678,	74	19.941678,
24	19.941678,	75	19.941678,
25	19.941678,	76	19.941678,
26	19.941678,	77	19.941678,
27	19.941678,	78	19.941678,
28	19.941678,	79	19.941678,
29	19.941678,	80	19.941678,
30	19.941678,	81	19.941678,
31	19.941678,	82	19.941678,
32	19.941678,	83	19.941678,
33	19.941678,	84	19.941678,
34	19.941678,	85	19.941678,
35	19.941678,	86	19.941678,
36	19.941678,	87	19.941678,
37	19.941678,	88	19.941678,
38	19.941678,	89	19.941678,
39	19.941678,	90	19.941678,
40	19.941678,	91	19.941678,
41	19.941678,	92	19.941678,
42	19.941678,	93	19.941678,
43	19.941678,	94	19.941678,
44	19.941678,	95	19.941678,
45	19.941678,	96	19.941678,
46	19.941678,	97	19.941678,
47	19.941678,	98	19.941678,
48	19.941678,	99	19.941678,
49	19.941678,	100	19.941678,
50	19.941678,	101	19.941678,
51	19.941678,	102	19.941678,
52	19.941678,	103	19.941678,
53	19.941678,	104	18.095609,
54	19.941678,	105	18.095609,
55	19.941678,	106	18.095609,
56	19.941678,	107	18.095609,
57	19.941678,	108	18.095609,
58	19.941678,	109	18.095609,
59	19.941678,	110	18.095609,
60	19.941678,	111	18.095609,
61	19.941678,	112	18.095609,
62	19.941678,	113	18.095609,
63	19.941678,	114	18.095609,
		115	18.095609

Skenario C (n=15)

20	"convergence": [	70	29.072181,
21	41.731993,	71	29.072181,
22	41.357691,	72	29.072181,
23	41.357691,	73	29.072181,
24	40.331549,	74	29.072181,
25	40.331549,	75	29.072181,
26	40.331549,	76	29.072181,
27	40.331549,	77	29.072181,
28	40.331549,	78	29.072181,
29	40.331549,	79	29.072181,
30	40.331549,	80	29.072181,
31	40.331549,	81	29.072181,
32	40.331549,	82	29.072181,
33	40.331549,	83	29.072181,
34	40.331549,	84	29.072181,
35	35.561795,	85	29.072181,
36	35.561795,	86	29.072181,
37	35.208874,	87	29.072181,
38	35.208874,	88	29.072181,
39	35.208874,	89	29.072181,
40	35.208874,	90	29.072181,
41	35.208874,	91	29.072181,
42	34.021189,	92	29.072181,
43	34.021189,	93	29.072181,
44	32.8985,	94	29.072181,
45	32.8985,	95	29.072181,
46	32.8985,	96	29.072181,
47	30.728385,	97	29.072181,
48	30.728385,	98	29.072181,
49	30.728385,	99	29.072181,
50	30.728385,	100	29.072181,
51	29.453676,	101	29.072181,
52	29.453676,	102	29.072181,
53	29.418789,	103	29.072181,
54	29.418789,	104	29.072181,
55	29.072181,	105	29.072181,
56	29.072181,	106	29.072181,
57	29.072181,	107	29.072181,
58	29.072181,	108	29.072181,
59	29.072181,	109	29.072181,
60	29.072181,	110	29.072181,
61	29.072181,	111	29.072181,
62	29.072181,	112	29.072181,
63	29.072181,	113	29.072181,
64	29.072181,	114	29.072181,
65	29.072181,	115	29.072181,
66	29.072181,	116	29.072181,
67	29.072181,	117	29.072181,
68	29.072181,	118	29.072181,
69	29.072181,	119	29.072181,

Skenario D (n=20)

25	"convergence": [	76	53.390292,
26	69.597643,	77	53.390292,
27	69.561927,	78	53.390292,
28	68.297184,	79	53.390292,
29	67.476686,	80	53.390292,
30	66.211943,	81	53.390292,
31	66.211943,	82	52.870511,
32	63.748899,	83	52.870511,
33	63.748899,	84	52.870511,
34	63.748899,	85	52.870511,
35	63.748899,	86	51.413747,
36	63.748899,	87	51.413747,
37	61.321092,	88	51.413747,
38	61.321092,	89	51.413747,
39	61.321092,	90	51.413747,
40	61.321092,	91	49.375388,
41	61.321092,	92	48.021101,
42	61.321092,	93	48.021101,
43	61.321092,	94	48.021101,
44	61.321092,	95	47.804568,
45	61.321092,	96	47.804568,
46	61.321092,	97	47.804568,
47	61.321092,	98	47.804568,
48	61.321092,	99	47.804568,
49	61.321092,	100	47.804568,
50	61.321092,	101	47.804568,
51	61.321092,	102	47.804568,
52	61.321092,	103	47.804568,
53	61.321092,	104	47.804568,
54	61.321092,	105	47.804568,
55	61.321092,	106	47.804568,
56	60.837565,	107	47.804568,
57	60.837565,	108	47.614195,
58	60.837565,	109	47.614195,
59	59.67041,	110	47.614195,
60	59.67041,	111	47.614195,
61	56.027486,	112	47.614195,
62	56.027486,	113	47.614195,
63	56.027486,	114	47.614195,
64	56.027486,	115	47.614195,
65	56.027486,	116	47.614195,
66	56.027486,	117	47.614195,
67	56.027486,	118	47.614195,
68	56.027486,	119	47.614195,
69	56.027486,	120	47.614195,
70	56.027486,	121	47.614195,
71	56.027486,	122	47.614195,
72	56.027486,	123	47.614195,
73	56.027486,	124	47.614195,
74	56.027486,	125	47.614195

Skenario E (n=25)

30	"convergence": [	81	80.386164,
31	93.629962,	82	80.386164,
32	91.43295,	83	80.386164,
33	91.43295,	84	80.386164,
34	91.43295,	85	80.386164,
35	91.43295,	86	80.386164,
36	91.431251,	87	80.386164,
37	91.203247,	88	80.386164,
38	91.203247,	89	80.386164,
39	91.203247,	90	80.386164,
40	88.61042,	91	80.386164,
41	88.00215,	92	80.386164,
42	88.00215,	93	80.386164,
43	88.00215,	94	80.386164,
44	88.00215,	95	80.386164,
45	88.00215,	96	80.386164,
46	88.00215,	97	80.386164,
47	87.317876,	98	80.386164,
48	87.155849,	99	76.215054,
49	87.155849,	100	76.215054,
50	86.186352,	101	74.277435,
51	86.186352,	102	74.277435,
52	86.186352,	103	74.277435,
53	83.765453,	104	74.277435,
54	83.765453,	105	73.825846,
55	83.765453,	106	73.013257,
56	83.765453,	107	73.013257,
57	83.765453,	108	73.013257,
58	83.765453,	109	73.013257,
59	83.765453,	110	73.013257,
60	83.765453,	111	72.008566,
61	83.765453,	112	72.008566,
62	83.765453,	113	72.008566,
63	81.877705,	114	72.008566,
64	81.877705,	115	72.008566,
65	81.877705,	116	72.008566,
66	81.877705,	117	72.008566,
67	81.877705,	118	72.008566,
68	81.877705,	119	70.478502,
69	81.877705,	120	70.478502,
70	81.877705,	121	69.736927,
71	81.877705,	122	69.736927,
72	81.710956,	123	69.736927,
73	80.552913,	124	69.736927,
74	80.552913,	125	69.736927,
75	80.386164,	126	69.736927,
76	80.386164,	127	69.736927,
77	80.386164,	128	69.484059,
78	80.386164,	129	69.484059,
79	80.386164,	130	69.484059

The screenshot displays the Ekspresimen S5A application interface. On the left, a sidebar shows a list of nodes (A, B, C, D, E) and a search bar. The main area features a map of Yogyakarta with a blue network overlay. The network consists of nodes (circles) and edges (lines) connecting them. A red dot on the map indicates a specific location. On the right, a panel shows a detailed view of the network structure, including a list of nodes and their connections. The nodes are labeled with IDs and names, and the connections are represented by lines. The application also includes a top bar with navigation and search options.