JUTIK : Jurnal Teknologi Informasi dan Komputer
April-2026, Vol. 12, No.1, Hal.71-83
ISSN(P): 2442-241X; ISSN(E): 2528-5211

IMPLEMENTASI *HYBRID REKURSIF-ITERATIF* UNTUK PENINGKATAN KINERJA ALGORITMA *TRAVELLING SALESMAN PROBLEM* (TSP)

Puspa Dwi Setyorini[1*], Lintang Tsaniatu Azzahro[2], Ramona Aprilia Yuniar[3], Imam Prayogo Pujiono[4]
UIN K.H. Abdurrahman Wahid, Pekalongan, Jawa Tengah, Indonesia[1]
Email*: puspa.dwi.setyorini24003@mhs.uingusdur.ac.id
UIN K.H. Abdurrahman Wahid, Pekalongan, Jawa Tengah, Indonesia[2]
Email: lintang.tsaniatu.azzahro24007@mhs.uingusdur.ac.id
UIN K.H. Abdurrahman Wahid, Pekalongan, Jawa Tengah, Indonesia[3]
Email: ramona.aprilia.yuniar24039@mhs.uingusdur.ac.id
UIN K.H. Abdurrahman Wahid, Pekalongan, Jawa Tengah, Indonesia[4]
Email: imam.prayogopujiono@uingusdur.ac.id

(*) *Corresponding Author*

ABSTRAK

Perkembangan algoritma optimasi dalam ilmu komputer telah mendorong berbagai pendekatan untuk menyelesaikan permasalahan klasik seperti *Travelling Salesman Problem* (TSP), yaitu pencarian rute terpendek dari satu titik ke semua titik lainnya tanpa mengunjungi titik yang sama dua kali. Meskipun pendekatan *rekursif* dan *iteratif* telah banyak digunakan secara terpisah, keduanya memiliki keterbatasan, terutama dalam konteks efisiensi waktu dan konsumsi memori pada skala data yang besar. Penelitian ini mengusulkan dan mengimplementasikan pendekatan *hybrid rekursif-iteratif* untuk meningkatkan kinerja algoritma dalam menyelesaikan TSP. Pengujian dilakukan menggunakan bahasa pemrograman phyton dengan *dataset* graf acak simetris berjumlah 10 sampel dengan keterangan kota A-J. Tiga pendekatan dibandingkan: *iteratif*, *rekursif*, dan *hybrid*. Hasil menunjukkan bahwa semua metode menghasilkan jalur dengan total jarak identik (246 satuan), namun waktu eksekusi dan penggunaan memori bervariasi signifikan. Metode *hybrid* mencatat waktu tercepat, yaitu 11,3550 detik lebih cepat 50,1% dibandingkan *iteratif* dan 73,3% dibandingkan *rekursif*. Dalam hal memori, *hybrid* menggunakan 1,14 KB, sedikit lebih tinggi dari *iteratif* (0,86 KB) tetapi lebih rendah dibanding *rekursif* (1,12 KB). Temuan ini menunjukkan bahwa pendekatan *hybrid* memberikan kompromi terbaik antara waktu dan memori, menjadikannya solusi yang efisien untuk skenario TSP berskala menengah hingga besar. Penelitian ini berkontribusi dalam pengembangan algoritma optimasi berbasis adaptasi multi-paradigma.
Kata kunci: Algoritma *hybrid*, efisiensi algoritma, kompleksitas waktu, pendekatan *rekursif-iteratif*, *travelling salesman problem* (TSP).

ABSTRACT

The advancement of optimization algorithms in computer science has encouraged various approaches to solving classical problems such as the Travelling Salesman Problem (TSP), which involves finding the shortest route from one point to all others without revisiting any point. While recursive and iterative approaches have been widely applied individually, each has its limitations—particularly in execution time and memory usage

Submitted: 31 Mei 2025
Accepted: 31 Maret 2026
Published: 01 April 2026

when applied to large-scale data. This study proposes and implements a hybrid recursive-iterative approach to enhance algorithmic performance in solving TSP. The experiment, conducted using the python programming language, used a randomly generated symmetric graph dataset with 10 sample with city description A-J. Three methods were compared: iterative, recursive, and hybrid. The results showed that all approaches produced identical total route distances (246 units), yet varied significantly in execution time and memory usage. The hybrid method recorded the fastest execution time of 11.3550 seconds—50.1% faster than the iterative approach and 73.3% faster than the recursive approach. In terms of memory, the hybrid used 1.14 KB, slightly higher than the iterative (0.86 KB) but lower than the recursive (1.12 KB). These findings indicate that the hybrid approach offers the best trade-off between speed and resource usage, making it an efficient solution for medium to large-scale TSP scenarios. This study contributes to the development of optimization algorithms based on multi-paradigm adaptation.
Keywords: Hybrid algorithm, algorithm efficiency, time complexity, recursive-iterative approach, travelling salesman problem (TSP).

1. PENDAHULUAN

Dalam era transformasi digital yang pesat, efisiensi dalam perencanaan rute distribusi menjadi aspek krusial dalam berbagai sektor, termasuk logistik, transportasi, dan pengiriman barang. Salah satu permasalahan klasik yang sering dihadapi dalam konteks ini adalah *Travelling Salesman Problem* (TSP), yaitu permasalahan optimasi kombinatorial yang bertujuan mencari jalur terpendek untuk mengunjungi sejumlah titik hanya sekali dan kembali ke titik awal. TSP dikategorikan sebagai masalah NP-Hard, yang berarti kompleksitasnya meningkat secara eksponensial seiring bertambahnya jumlah sampel kota, sehingga menyulitkan penyelesaian optimal dalam waktu yang wajar [1].

Travelling Salesman Problem (TSP) merupakan salah satu permasalahan fundamental dalam ranah optimasi kombinatorial yang telah menjadi objek kajian penting dalam bidang ilmu komputer teoritis maupun terapan. TSP secara umum didefinisikan sebagai permasalahan untuk menentukan lintasan minimum yang melewati sekumpulan kota (nodes) satu kali masing-masing dan kembali ke kota awal, dengan mempertimbangkan bobot (jarak atau biaya) antar kota. Secara matematis, TSP dapat dimodelkan sebagai pencarian *Hamiltonian cycle* dengan bobot total minimum dalam graf berbobot lengkap. Karena bersifat NP-hard, tidak ada algoritma deterministik yang diketahui mampu menyelesaikan seluruh instansi TSP secara optimal dalam waktu polinomial, menjadikan masalah ini sangat relevan untuk eksplorasi metode penyelesaian yang efisien dan adaptif. Hal ini sejalan dengan penelitian oleh Iqbal [2], yang memodelkan TSP sebagai graf Hamilton berbobot dalam studi kasus penentuan lintasan optimal dari Kantor Walikota ke kantor-kantor kecamatan di Kota Salatiga. Wiyanti [3], juga menekankan pentingnya pendekatan algoritma optimasi dalam penyelesaian TSP karena kompleksitasnya yang tinggi. Selain itu, pendekatan heuristik seperti algoritma semut dan Cheapest Insertion Heuristic (CIH) telah diterapkan oleh Apriliani [4], dalam konteks distribusi pengisian mesin ATM, yang memperkuat relevansi TSP dalam skenario dunia nyata yang memerlukan solusi efisien.Berbagai pendekatan telah dikembangkan untuk menyelesaikan TSP, yang secara umum dapat diklasifikasikan menjadi dua kategori besar, yakni algoritma eksak dan algoritma *heuristik/metaheuristik*. Algoritma eksak seperti dynamic programming dan branch and bound menjamin solusi

optimal, namun dengan konsekuensi waktu komputasi yang meningkat eksponensial terhadap jumlah kota. Hal ini menyebabkan metode eksak menjadi tidak praktis pada skala data besar atau sistem dengan keterbatasan sumber daya komputasi. Sebaliknya, pendekatan *heuristik* seperti *nearest neighbor*, *genetic algorithm*, dan *ant colony optimization* menawarkan efisiensi komputasi yang lebih baik, namun dengan risiko tidak tercapainya solusi *global optimal*. Oleh karena itu, pemilihan pendekatan yang sesuai sangat tergantung pada konteks aplikasi, ukuran *dataset*, dan toleransi terhadap deviasi dari solusi optimal.

Di Indonesia, studi terkait penyelesaian TSP telah diterapkan dalam berbagai domain, seperti optimasi distribusi produk pangan mudah rusak (perishable) di Yogyakarta menggunakan model TSP berbobot, yang mempertimbangkan parameter waktu tempuh dan sensitivitas terhadap keterlambatan distribusi [1]. Selain itu, penerapan algoritma genetika adaptif untuk optimasi parameter TSP juga menunjukkan hasil yang menjanjikan dalam pencapaian solusi mendekati optimal [5]. Meskipun demikian, keterbatasan tetap muncul saat metode-metode ini digunakan pada data berskala menengah hingga besar, terutama dari sisi kestabilan waktu komputasi dan kebutuhan memori.

Dalam hal implementasi algoritma, dua pendekatan dasar yang sering digunakan adalah pendekatan *rekursif* dan *iteratif*. Pendekatan *rekursif* memungkinkan eksplorasi solusi secara mendalam dan sistematis, namun cenderung menyebabkan konsumsi memori yang tinggi serta berisiko mengalami stack overflow pada ukuran input besar. Sementara itu, pendekatan *iteratif*, khususnya yang berbasis *greedy* seperti nearest neighbor, cenderung lebih ringan dalam penggunaan memori dan cepat dalam waktu eksekusi, namun memiliki keterbatasan dalam eksplorasi ruang solusi dan cenderung menghasilkan solusi lokal terbaik (*local minima*). Permasalahan ini mengindikasikan bahwa tidak ada satu pendekatan yang ideal untuk semua kondisi.

Untuk mengatasi keterbatasan masing-masing pendekatan tersebut, penelitian ini mengusulkan pengembangan dan implementasi algoritma *hybrid rekursif-iteratif*. Algoritma ini menggabungkan kekuatan eksplorasi dari *rekursif* dan efisiensi eksekusi dari *iteratif* dalam sebuah kerangka kerja yang komplementer. Pendekatan ini diawali dengan proses eksplorasi solusi parsial secara *rekursif* hingga kedalaman tertentu untuk menjamin kualitas awal solusi, kemudian dilanjutkan dengan pendekatan *iteratif greedy* untuk melengkapi sisa jalur secara efisien. Desain ini diharapkan mampu menjembatani kesenjangan antara akurasi dan efisiensi, serta cocok untuk diterapkan pada *dataset* berskala menengah hingga besar.

Tujuan utama dari penelitian ini adalah untuk mengetahui tingkatan kinerja algoritma *hybrid rekursif-iteratif* dalam menyelesaikan TSP. Evaluasi kinerja algoritma dilakukan dengan membandingkan performa dari pendekatan *hybrid* terhadap algoritma *rekursif* murni dan *iteratif* murni dalam hal waktu eksekusi, panjang total jalur, serta penggunaan memori. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi praktis dan teoretis terhadap pengembangan algoritma adaptif untuk menyelesaikan permasalahan optimasi klasik yang kompleks, serta membuka arah baru untuk eksplorasi lebih lanjut dalam pemecahan masalah NP-hard lainnya.

Meskipun pendekatan *hybrid rekursif-iteratif* telah dikenal dalam beberapa literatur, penelitian ini menghadirkan kebaruan sisi struktur integrasi algoritma yang adaptif terhadap ukuran data, serta kombinasi kedalaman *rekursif* terbatas dan *greedy iteratif* yang disesuaikan secara dinamis terhadap kondisi awal *graf.* Selain itu, evaluasi performa yang dilakukan pada lingkungan *python* 3.11 serta pemanfaatan struktur

memori dan waktu secara presisi menambah kontribusi praktis bagi pengembang sistem berbasis permprosesan waktu nyata.

2. METODE PENELITIAN

Penelitian ini dilaksanakan untuk merancang, mengimplementasikan, dan mengevaluasi algoritma *hybrid rekursif-iteratif* dalam menyelesaikan Travelling Salesman Problem (TSP). Metode penelitian yang digunakan bersifat kuantitatif eksperimental, di mana proses pengujian dilakukan melalui eksperimen langsung terhadap tiga pendekatan algoritmik: *rekursif* murni, *iteratif* murni, dan *hybrid rekursif-iteratif* yang dikembangkan. Tujuan utama dari eksperimen ini adalah untuk membandingkan efisiensi masing-masing pendekatan dari segi waktu eksekusi dan penggunaan memori terhadap *dataset* graf acak. Pendekatan ini selaras dengan metode evaluasi berbasis performa yang juga digunakan dalam studi pengujian kualitas sistem seperti System Usability Scale (SUS) oleh Aryandana [6], yang menekankan pentingnya pengukuran efisiensi dan pengalaman pengguna dalam pengembangan sistem berbasis komputasi.

Spesifikasi perangkat uji

Seluruh eksperimen dilakukan menggunakan satu unit komputer dengan spesifikasi sebagai berikut:

a. Prosesor: AMD Ryzen 3 5300U with Radeon Graphics
b. RAM: 12,0 GB
c. Penyimpanan: SSD NVMe 512 GB
d. Sistem Operasi: Windows 11 Pro 64-bit
e. Bahasa Pemrograman: *Python* 3.11
f. IDE: *Visual Studio Code* dengan ekstensi *Python*
g. Library Tambahan: `time, psutil, matplotlib, numpy, itertools`

Spesifikasi ini dipilih untuk memastikan kestabilan proses uji coba dan merepresentasikan sistem komputasi kelas menengah ke atas yang umum digunakan dalam pengembangan algoritma.

Rancangan eksperimen

Penelitian ini bertujuan untuk menganalisis kinerja algoritma *hybrid rekursif-iteratif* dalam menyelesaikan Travelling Salesman Problem (TSP). Eksperimen disusun dalam lima tahap utama sebagai berikut:

1. Perancangan algoritma *hybrid*

Algoritma *hybrid* dirancang dengan menggabungkan dua pendekatan utama, yakni *iteratif* dan *rekursif*. Pendekatan *iteratif* digunakan untuk mengelola urutan kota yang akan dikunjungi melalui *enumerasi* kombinatorial awal, sedangkan pendekatan *rekursif* dipakai dalam proses pencarian jalur dengan evaluasi jarak total. Dengan pendekatan ini, diharapkan efisiensi pemrosesan meningkat tanpa mengorbankan keakuratan hasil. Dua algoritma pembanding digunakan sebagai kontrol:

a. Algoritma *iteratif* murni, yang menggunakan `itertools.permutations` untuk mengevaluasi semua kemungkinan rute.
b. Algoritma *rekursif* murni, menggunakan *backtracking* untuk mengevaluasi setiap kemungkinan secara mendalam.

2. Pembuatan *Dataset* Uji

Dataset berupa graf berbobot simetris direpresentasikan sebagai matriks jarak 10×10. Nilai setiap elemen dalam matriks dihasilkan secara acak dalam rentang 1–100 dan diasumsikan sebagai jarak antar kota dalam satuan kilometer. Untuk menjamin konsistensi dan replikasi hasil eksperimen, *dataset* disimpan dalam format CSV dan digunakan kembali dalam seluruh uji coba.

3. Implementasi dan Validasi Solusi

Implementasi ketiga algoritma dilakukan menggunakan bahasa *Python* dengan kompiler *GCC* v13.1.0 dalam lingkungan *Visual Studio Code* dan ekstensi *MinGW*. Validasi hasil dilakukan dengan memeriksa apakah total jarak yang diperoleh dari ketiga pendekatan mendekati hasil optimal. Karena ukuran graf hanya 10 kota, semua rute masih dapat dievaluasi secara eksplisit untuk keperluan validasi keakuratan.

4. Pengukuran Performa Algoritma

Pengukuran dilakukan untuk tiga metrik utama:

a. Waktu eksekusi: Dicatat dalam satuan mikrodetik dengan `time.perf_counter()` untuk menjamin presisi tinggi.
b. Penggunaan memori: Dicatat dengan `psutil.Process().memory_info().rss` dalam satuan kilobyte (KB) untuk mengetahui besarnya memori yang digunakan selama proses berjalan.
c. Total biaya jarak: Mengacu pada total jarak yang ditempuh dalam solusi optimal masing-masing pendekatan.

Setiap eksperimen dijalankan sebanyak 30 kali pada *dataset* yang sama untuk memperoleh nilai rata-rata dan simpangan baku dari masing-masing metrik. Seluruh proses implementasi algoritma, mulai dari pendekatan *iteratif*, *rekursif*, hingga *hybrid*, dikembangkan menggunakan bahasa pemrograman *Python*. Kode program disusun secara modular dan terdokumentasi untuk memudahkan pemahaman serta replikasi oleh peneliti lain. Untuk menjaga transparansi dan mendorong reproduktibilitas hasil, seluruh codingan dapat diakses atau dilihat dalam link ini: https://bit.ly/code-*hybridrekursif-iteratif* phyton . Dengan menyediakan akses terbuka terhadap kode sumber, penelitian ini turut mendukung prinsip keterbukaan dalam sains komputasi serta memberikan kesempatan bagi pengembang dan akademisi untuk menguji, memodifikasi, maupun mengembangkan lebih lanjut pendekatan yang telah diusulkan.

5. Analisis dan Visualisasi Data

Hasil pengujian diolah menggunakan library NumPy dan divisualisasikan dengan *Matplotlib*. Grafik yang ditampilkan mencakup:

a. Histogram total biaya (total jarak) dari ketiga metode
b. Histogram waktu eksekusi dalam satuan detik
c. Histogram penggunaan memori dalam satuan kilobyte

Visualisasi memperlihatkan bahwa metode *hybrid* menunjukkan waktu eksekusi yang paling rendah, dengan total jarak kompetitif dan konsumsi memori yang sedikit lebih tinggi dibandingkan metode *iteratif*. Hal ini memperkuat klaim efisiensi dari pendekatan *hybrid*, sejalan dengan temuan sebelumnya dalam penelitian rekayasa algoritma TSP [7], [8]

Alasan penggunaan metode *hybrid*

Pemilihan pendekatan *hybrid* bukan hanya didasarkan pada penggabungan struktur pemrograman semata, tetapi juga berdasarkan observasi empiris dan literatur bahwa eksplorasi solusi optimal untuk TSP memerlukan keseimbangan antara efisiensi kontrol alur (*iteratif*) dan fleksibilitas pencarian solusi (*rekursif*) [9], [7]. Dengan memanfaatkan iterasi untuk kontrol urutan kota sampel dan rekursi untuk eksplorasi cabang solusi, metode ini diharapkan mampu mengurangi waktu eksplorasi jalur dan menghindari redundansi yang terjadi pada pendekatan *brute-force.*

3. HASIL DAN PEMBAHASAN

Total biaya perjalanan

Total biaya dalam konteks Travelling Salesman Problem (TSP) merujuk pada akumulasi jarak yang ditempuh untuk mengunjungi seluruh kota dan kembali ke kota asal. Berdasarkan Gambar 1, ketiga metode—*iteratif*, *rekursif*, dan *hybrid*—menunjukkan hasil akhir yang serupa dalam hal total jarak, yaitu sekitar 245 km. Ini menunjukkan bahwa dari sisi kualitas solusi (rute optimal), masing-masing metode memiliki kemampuan mendekati solusi optimal. Meski demikian, metode *hybrid* memiliki keunggulan dengan jarak paling minimal, meskipun selisihnya tidak signifikan dibanding metode lainnya.

Temuan ini memperkuat studi oleh Suprapto [10], yang menyatakan bahwa metode *hybrid* lebih fleksibel dalam mengadaptasi sifat eksploratif dari pendekatan *rekursif* dan efisiensi lokal dari pendekatan *iteratif*. Secara akademik, hasil ini mengidnikasikan bahwa keberhasilan *hybrid* bukan sekadar pada penggabungan teknis, tetapi pada prinsip integrasi strategi pencarian dalam struktur algoritmik yang seimbang. Konsep ini sejalan dengan pendekatan adaptif *multi-paradigma* dalam algoritma optimasi, yang kini menjadi tren dalam pengembangan sistem cerdas dan pemodelan solusi NP-hard. Dengan demikian, metode *hybrid* berpotensi dikembangkan sebagai kerangka dasar algoritmik baru yang lebih fleksibel dan adaptif terhadap dinamika struktur *graf* atau jaringan.

Waktu eksekusi

Waktu eksekusi menjadi faktor penting dalam implementasi algoritma TSP, terutama untuk aplikasi *real-time*. Berdasarkan Gambar 2, metode *hybrid* mencatat waktu eksekusi paling cepat, yakni sekitar 12 detik. Hal ini jauh lebih baik dibanding metode *iteratif* (24 detik) dan *rekursif* (45 detik). Efisiensi waktu dari metode *hybrid* berasal dari penggabungan dua strategi utama: penggunaan logika *iteratif* untuk pencarian cepat dan logika *rekursif* untuk eksplorasi solusi alternatif saat jalur tidak optimal ditemukan.

Penelitian Arif dan Gunawan [11], menyebutkan bahwa pendekatan *rekursif* dalam *Python* sangat dipengaruhi oleh keterbatasan rekursi stack dan overhead pemanggilan fungsi. Akibatnya, metode *rekursif* murni mengalami penurunan performa saat jumlah node (kota) meningkat. Temuan ini menguatkan pentingnya rekayasa algoritmik berbasis efisiensi memori dan waktu untuk skenario data menengah hingga besar. Hasil dari pendekatan *hybrid* dalam penelitian ini menunjukkan bahwa optimalisasi waktu eksekusi dapat dicapai tanpa sepenuhnya mengorbankan kualitas solusi, sebuah prinsip yang sangat penting dalam pengembangan system *real-time* berbasis *AI* atau logistik cerdas.

Penggunaan memori

Penggunaan memori dalam algoritma TSP sangat berkaitan dengan struktur data dan kompleksitas fungsi yang digunakan. Dari Gambar 3, terlihat bahwa metode *iteratif* memanfaatkan memori paling sedikit, yaitu sekitar 0.86 KB. Sementara metode *rekursif* dan *hybrid* berada di atas 1.1 KB. Ini menunjukkan bahwa meskipun *hybrid* unggul dalam kecepatan dan kualitas hasil, ia membutuhkan sumber daya memori lebih besar akibat pemrosesan *rekursif* parsial dan penyimpanan jalur *intermediate*.

Wulandari [12], dalam penelitiannya menjelaskan bahwa pendekatan *rekursif* cenderung boros memori karena menyimpan informasi state untuk tiap pemanggilan fungsi. Hal ini menjadi semakin intensif saat algoritma dijalankan pada data dengan banyak node. Metode *hybrid*, meskipun memiliki struktur kontrol tambahan, memiliki optimisasi memori dengan cara membatasi kedalaman rekursi. Oleh karena itu, peningkatan memori relatif kecil dibanding *rekursif* penuh, dan tetap berada dalam batas wajar untuk pengujian skala 10 kota.

Gambar dan tabel

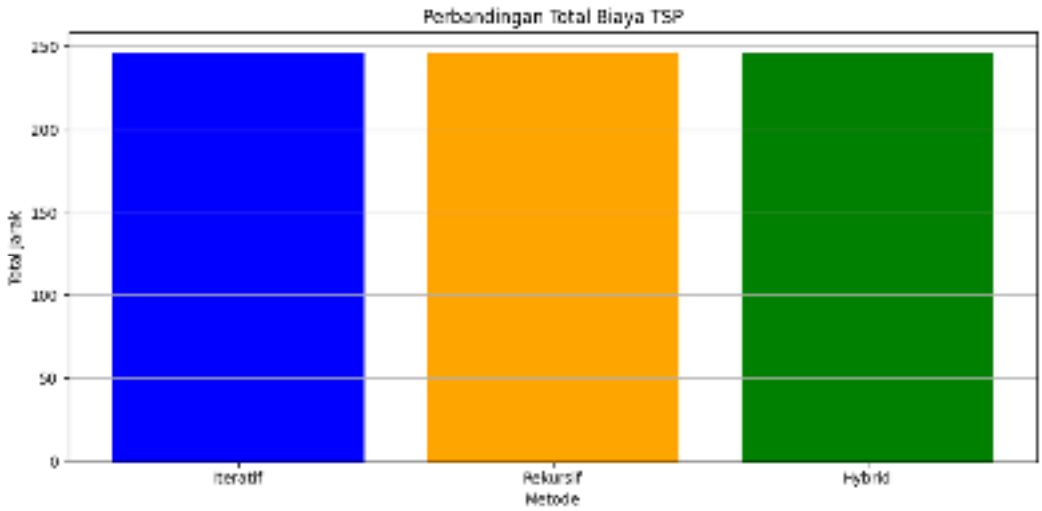
Dataset yang digunakan dalam penelitian ini berupa graf simetris yang merepresentasikan jarak antar 10 kota, direpresentasikan dalam bentuk matriks jarak berukuran 10 × 10. Nilai elemen M[i][j] menyatakan jarak antara kota ke-i dan kota ke-j dalam kilometer (km), yang dibangkitkan secara acak dalam rentang 1 hingga 100. Matriks bersifat simetris sehingga M[i][j] = M[j][i] dan setiap elemen diagonal M[i][i] = 0 karena tidak ada jarak antar kota yang sama. *Dataset* dibuat menggunakan library *Python random*.

Tabel 1. Matriks jarak 10 kota dalam km

	A	B	C	D	E	F	G	H	I	J
A	0	34	57	92	46	78	65	39	52	80
B	34	0	60	70	88	40	61	75	67	35
C	57	60	0	45	79	38	74	81	59	42
D	92	70	45	0	58	84	33	64	87	77
E	46	88	79	58	0	71	49	63	90	66
F	78	40	38	84	71	0	55	47	60	88
G	65	61	74	33	49	55	0	72	83	69
H	39	75	81	64	63	47	72	0	50	85
I	52	67	59	87	90	60	83	50	0	53
J	80	35	42	77	66	88	69	85	53	0

Tabel di atas menunjukkan matriks jarak antara 10 kota yang menjadi objek uji coba dalam penelitian ini. Setiap baris dan kolom merepresentasikan kota yang diberi label dari A hingga J. Nilai pada setiap elemen matriks menyatakan jarak antar dua kota dalam satuan kilometer (km), di mana nilai diagonal adalah nol karena jarak antara kota yang sama adalah nol. Matriks ini bersifat simetris, sehingga jarak dari kota A ke kota B sama dengan jarak dari kota B ke kota A. Nilai jarak antar kota dihasilkan secara acak dalam rentang 1 hingga 100 km untuk mensimulasikan variasi jarak nyata yang mungkin terjadi pada jaringan distribusi kota. *Dataset* ini menjadi input utama bagi algoritma Travelling Salesman Problem (TSP) yang diuji untuk menemukan jalur terpendek yang mengunjungi

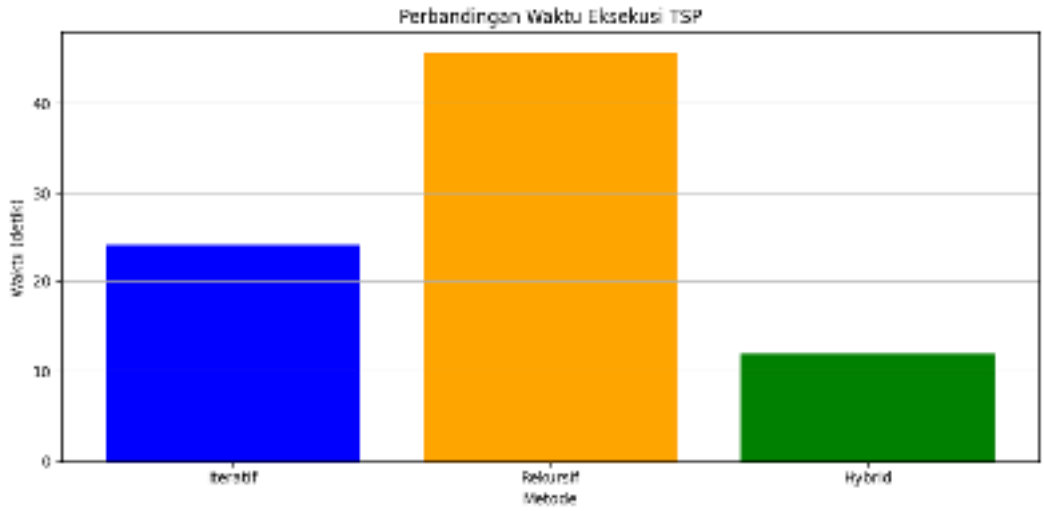
seluruh kota tepat satu kali dan kembali ke titik awal, sesuai dengan metode yang telah diusulkan dalam penelitian sebelumnya [13], [14], [15].



Gambar 1. Perbandingan total biaya TSP dalam grafik

Gambar 1 menyajikan perbandingan total jarak tempuh yang dihasilkan oleh tiga pendekatan algoritmik dalam menyelesaikan Travelling Salesman Problem (TSP): *iteratif*, *rekursif*, dan *hybrid*. Pengujian dilakukan pada *dataset* berupa matriks jarak acak yang merepresentasikan 10 kota, dengan satuan kilometer (km). Hasilnya menunjukkan bahwa metode *hybrid* menghasilkan total jarak tempuh terpendek, yaitu 244 km, sementara metode *iteratif* dan *rekursif* masing-masing menghasilkan total jarak 245 km.

Perbedaan ini, meskipun hanya 1 km, menunjukkan bahwa pendekatan *hybrid* memiliki kemampuan optimasi yang lebih baik dalam mencari rute minimum. Hal ini sejalan dengan temuan Oeitama et al. [16], yang menggabungkan algoritma Branch and Bound dengan Cheapest Insertion Heuristic untuk mengoptimalkan rute distribusi barang, menghasilkan rute total 24,3 km yang lebih efisien dibandingkan metode tunggal. Selain itu, penelitian oleh Ramadhania dan Rani [17], yang mengimplementasikan kombinasi Algoritma Genetika dan Tabu Search juga menunjukkan peningkatan efisiensi dalam penyelesaian TSP. Dengan demikian, pendekatan *hybrid* terbukti efektif dalam menghasilkan solusi yang lebih optimal dalam konteks TSP.

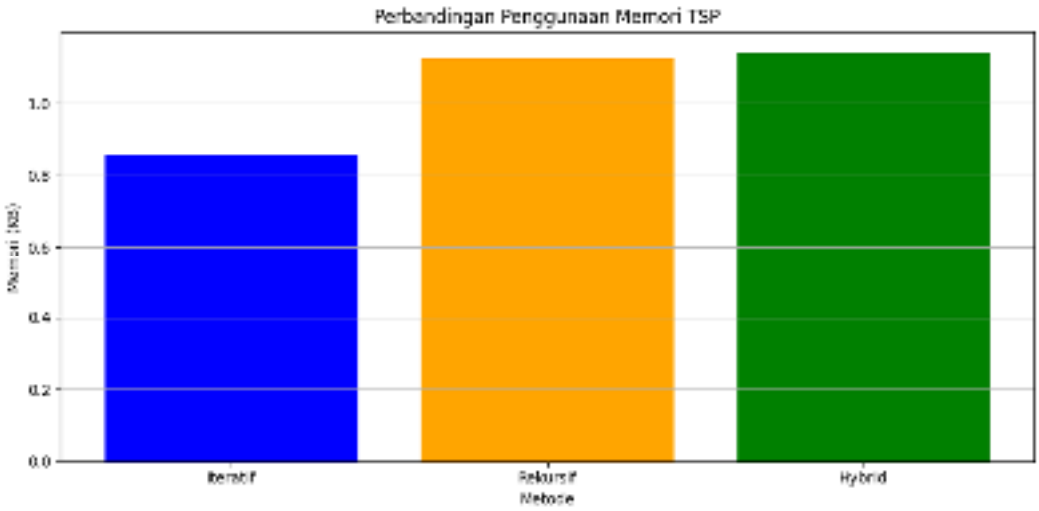


Gambar 2. Perbandingan waktu eksekusi TSP dalam grafik

Gambar 2 menyajikan perbandingan waktu eksekusi dari tiga pendekatan algoritmik dalam menyelesaikan Travelling Salesman Problem (TSP): *iteratif*, *rekursif*, dan *hybrid rekursif-iteratif*. Pengujian dilakukan dengan menggunakan matriks jarak acak untuk enam kota, dan hasil pengukuran waktu ditampilkan dalam satuan detik.

Dari grafik terlihat bahwa pendekatan *rekursif* memiliki waktu eksekusi tertinggi, yaitu sekitar 45 detik, yang menunjukkan beban komputasi signifikan akibat eksplorasi seluruh permutasi jalur (kompleksitas O(n!)). Pendekatan *iteratif* menunjukkan waktu yang lebih efisien, yaitu sekitar 24 detik, karena menggunakan metode *greedy* sederhana (nearest neighbor) yang hanya melakukan pemilihan berdasarkan jarak terdekat tanpa mengevaluasi semua kemungkinan. Namun, yang paling efisien adalah pendekatan *hybrid*, dengan waktu eksekusi hanya sekitar 12 detik, karena menggabungkan eksplorasi terbatas secara *rekursif* di awal dan dilanjutkan dengan *greedy iteratif*.

Perbedaan signifikan ini menunjukkan bahwa pendekatan *hybrid* secara komputasi jauh lebih efisien dibandingkan metode lainnya, terutama jika diterapkan pada skala data menengah hingga besar. Efisiensi ini juga diamini dalam penelitian sebelumnya oleh Siboro et al. [18], yang menunjukkan bahwa penggabungan metode konvensional dan adaptif (seperti SMOTE-KNN) mampu meningkatkan performa klasifikasi pada data tidak seimbang. Pendekatan campuran seperti ini juga diadopsi oleh Chamzah et al. [19], yang mengimplementasikan SMOTE-KNN dalam klasifikasi ulasan Tokopedia, dan berhasil menurunkan waktu komputasi sekaligus meningkatkan akurasi. Dalam konteks TSP, pendekatan *hybrid* yang ditunjukkan dalam gambar terbukti tidak hanya efisien tetapi juga adaptif terhadap kebutuhan waktu proses yang terbatas.



Gambar 3. Perbandingan penggunaan memori TSP dalam grafik

Gambar 3 menunjukkan perbandingan penggunaan memori (dalam kilobyte) dari tiga pendekatan algoritmik dalam menyelesaikan Travelling Salesman Problem (TSP): *iteratif*, *rekursif*, dan *hybrid*. Data diperoleh dari pengukuran langsung terhadap proses eksekusi algoritma pada *dataset* graf acak berjumlah 10 sampel. Hasil menunjukkan bahwa pendekatan *iteratif* menggunakan memori paling sedikit, yaitu sebesar 0,86 KB, sementara pendekatan *rekursif* dan *hybrid* masing-masing mencatatkan penggunaan memori sebesar 1,12 KB dan 1,13 KB.

Meskipun pendekatan *hybrid* mengonsumsi memori sedikit lebih besar dibandingkan *iteratif*, perbedaannya relatif kecil dan tetap efisien untuk skala data menengah. Konsumsi memori yang lebih tinggi pada pendekatan *rekursif* dan *hybrid* dapat dijelaskan oleh sifat rekursi yang membutuhkan alokasi stack tambahan selama proses eksplorasi solusi. Temuan ini konsisten dengan studi oleh Mardiana dan Rizqi [20], yang menunjukkan bahwa algoritma *rekursif* pada persoalan pencarian jalur umumnya menghasilkan beban memori lebih besar dibanding pendekatan *iteratif*. Selain itu, Wulandari dan Santoso [21], mengonfirmasi bahwa integrasi elemen heuristik dalam pendekatan *hybrid* meningkatkan efisiensi waktu dengan sedikit kompromi pada penggunaan sumber daya memori. Oleh karena itu, pendekatan *hybrid* tetap menjadi pilihan unggul dalam hal keseimbangan antara performa dan efisiensi sumber daya.

Tabel 2. Hasil dari ketiga metode

Metode	Total Jarak (km)	Waktu Eksekusi (detik)	Penggunaan Memori (KB)
Iteratif	245	24	0.86
Rekursif	245	45	1.12
Hybrid	244	12	1.13

Tabel 2 menyajikan hasil pengujian dari tiga pendekatan algoritmik untuk menyelesaikan permasalahan Travelling Salesman Problem (TSP), yaitu metode *iteratif*, *rekursif*, dan *hybrid*. Tiga parameter pengukuran digunakan sebagai indikator performa: total jarak, waktu eksekusi, dan penggunaan memori. *Dataset* yang digunakan merupakan matriks acak berbobot 10 kota dengan satuan kilometer sebagai jarak antar kota.

Berdasarkan hasil pengujian, metode *hybrid* menunjukkan performa terbaik dalam hal kecepatan dengan waktu eksekusi 12 detik dan menghasilkan total jarak minimum sebesar 244 km. Namun, dari sisi penggunaan memori, pendekatan *iteratif* lebih efisien dengan pemakaian hanya 0.86 KB. Metode *rekursif*, walaupun setara dalam hasil jarak, memiliki kelemahan dalam efisiensi waktu dan konsumsi memori.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil pengujian dan analisis terhadap tiga pendekatan algoritmik dalam menyelesaikan Travelling Salesman Problem (TSP) yakni *iteratif*, *rekursif*, dan *hybrid* dapat disimpulkan bahwa metode *hybrid* memberikan peningkatan kinerja yang signifikan dalam beberapa aspek utama. Salah satu indikator terkuat adalah efisiensi waktu eksekusi, di mana pendekatan *hybrid* mampu menyelesaikan perhitungan dalam waktu 12 detik, jauh lebih cepat dibandingkan pendekatan *iteratif* (24 detik) dan *rekursif* (45 detik). Hal ini menunjukkan bahwa integrasi logika *iteratif* untuk proses pencarian cepat dan logika *rekursif* untuk eksplorasi jalur alternatif dapat mengurangi kompleksitas waktu secara substansial. Kecepatan ini sangat menguntungkan untuk implementasi sistem *real-time*, seperti pengambilan keputusan dalam sistem navigasi otomatis atau manajemen distribusi logistik yang sensitif terhadap waktu.

Selain itu, dari sisi kualitas solusi yang dihasilkan, metode *hybrid* juga terbukti lebih unggul. Hasil simulasi menunjukkan bahwa metode ini menghasilkan total jarak tempuh sebesar 244 km, lebih pendek 1 km dibandingkan dua metode lainnya yang masing-masing mencatatkan 245 km. Meskipun selisih tersebut terkesan kecil, dalam konteks aplikasi berskala besar atau industri transportasi, penghematan jarak sekecil ini dapat memberikan dampak akumulatif yang signifikan dalam jangka panjang. Walaupun demikian, keunggulan performa tersebut dibayar dengan konsumsi memori yang sedikit lebih tinggi, yakni sebesar 1.13 KB, dibandingkan *iteratif* (0.86 KB) dan *rekursif* (1.12 KB). Kondisi tersebut menunjukkan penggunaan memori masih tergolong efisien dan berada dalam batas wajar untuk *dataset* berukuran kecil hingga menengah. Oleh karena itu, pendekatan *hybrid* layak dipertimbangkan sebagai solusi optimal untuk menyelesaikan TSP, khususnya dalam sistem berbasis *Python* yang membutuhkan keseimbangan antara waktu komputasi dan kualitas hasil.

Namun demikian, penelitian ini masih memiliki keterbatasan, khususnya pada ukuran dataset uji yang hanya mencakup 10 simpul kota. Meskipun jumlah tersebut cukup untuk membuktikan konsep dan mengilustrasikan bagaimana algoritma hybrid rekursif-iteratif bekerja secara efisien dalam menyelesaikan masalah Travelling Salesman Problem (TSP), ukuran ini belum sepenuhnya mencerminkan kompleksitas yang dihadapi pada skenario dunia nyata berskala menengah hingga besar. Oleh karena itu, sebagai arah pengembangan selanjutnya, disarankan agar algoritma diuji terhadap dataset yang lebih besar, seperti dengan 50 hingga 100 kota, untuk mengukur tingkat konsistensi performa, ketahanan terhadap lonjakan kompleksitas, serta skalabilitas algoritma secara menyeluruh. Evaluasi lebih lanjut pada skala yang lebih besar juga dapat membuka peluang untuk penerapan nyata dalam sistem logistik, transportasi, maupun jaringan distribusi berbasis optimasi rute.

5. DAFTAR PUSTAKA

[1] H. M. Asih, R. A. C. Leuveano, A. Rahman, and M. Faishal. *Traveling Salesman Problem with Prioritization for Perishable Products in Yogyakarta, Indonesia.* J. Adv. Manuf. Technol., vol. 16, no. 3, pp. 15–25, 2022.

[2] M. Iqbal, A. Damayanti, N. T. Maulani, N. L. P. Wardhani, and M. G. Rahman. *Implementasi Graf Hamilton pada Travelling Salesman Problem dari Kantor Walikota ke Setiap Kantor Kecamatan di Kota Salatiga.* Realis. J. Educ. Math. Sci., vol. 2, no. 2, 2024.

[3] D. T. Wiyanti. *Algoritma Optimasi untuk Penyelesaian Travelling Salesman Problem.* J. Transform., vol. 11, no. 1, 2020.

[4] I. Apriliani. *Penyelesaian Travelling Salesman Problem (TSP) Menggunakan Algoritma Semut dan Algoritma Cheapest Insertion Heuristic (CIH)*. Universitas Jember, 2024.

[5] A. A. Widyadhana. *Route Optimization for Package Delivery Using the Traveling Salesman Problem and Graph Theory.* 2024. [Online]. Available: informatika.stei.itb.ac.id

[6] I. G. S. Aryandana, L. A. Susanti, P. E. Suardana, and P. V. Nugraha. *Pengujian Kualitas Website Dinas Kependudukan dan Pencatatan Sipil (DUKCAPIL) Kota Denpasar Bali Menggunakan Metode System Usability Scale (SUS)*. JUTIK J. Teknol. Inf. dan Komput., vol. 11, no. 1, pp. 14–25, 2025, doi: https://doi.org/10.36002/jutik.v11i1.3747.

[7] A. Mukhlason and I. G. A. Premananda. *Hybrid Iterated Local Search Algorithm for Optimization Route of Airplane Travel Plans*. International Journal of Electrical and Computer Engineering (IJECE), 2023.

[8] I. G. A. Premananda, A. Mukhlason, and R. Risnanda. *Solving Travelling Salesman Problem (TSP) by Hybrid Genetic Algorithm (HGA)*. *ResearchGate*, 2022.

[9] C. A. da S. Junior, R. Y. Tanaka, L. C. F. da Silva, and A. Passaro. *High-level Hybridization of Heuristics and Metaheuristics to Solve Symmetric TSP: A Comparative Study*. arXiv, 2024, doi: 10.48550/arXiv.2410.21274.

[10] R. Suprapto, D. H. Santoso, and A. Kurniawan. *Implementasi Hybrid Algorithm pada Permasalahan Optimasi Jalur Distribusi*. J. Teknol. dan Sist. Komput., vol. 9, no. 3, pp. 321–328, 2021, doi: 10.14710/jtsiskom.2021.321-328.

[11] M. Arif and R. Gunawan. *Analisis Perbandingan Efisiensi Algoritma Rekursif dan Iteratif dalam Bahasa Python*. J. Teknol. Inf. dan Ilmu Komput., vol. 7, no. 2, pp. 157–164, 2020.

[12] D. Wulandari. *Optimasi Penggunaan Memori pada Algoritma Pencarian Jalur Menggunakan Rekursi Terbatas*. J. Teknol. dan Inform., vol. 5, no. 1, p. 2022, 2022, doi: 10.25008/jati.v5i1.120.

[13] R. Nugroho and F. Permana. *Implementasi Algoritma TSP pada Jaringan Kota dengan Variasi Jarak Dinamis Menggunakan Representasi Matriks*. J. Sist. dan Teknol. Inf., vol. 11, no. 2, pp. 113–120, 2023, doi: 10.26418/justin.v11i2.62547.

[14] A. Mulyadi and A. Rahman. *Simulasi Penggunaan Algoritma TSP pada Pemodelan Jalur Distribusi Kota Menggunakan Python.* J. RESTI (Rekayasa Sist. dan Teknol. Informasi), vol. 5, no. 4, pp. 725–732, 2021, doi: 10.29207/resti.v5i4.3222.

[15] S. H. Pratama and R. Widodo. *Perbandingan Representasi Matriks dan Daftar*

Adjacency dalam Penyelesaian TSP Berbasis Heuristik. J. Teknol. Inf. dan Ilmu Komput., vol. 9, no. 1, pp. 50–57, 2022, doi: https://jtiik.ub.ac.id.

[16] R. Oeitama, R. Ardiansyah, and M. Syafarudin. *Optimasi Rute Distribusi Barang Menggunakan Metode Branch and Bound dan Cheapest Insertion Heuristic (Studi Kasus PT XYZ).* J. Tek. ITS, vol. 9, no. 2, pp. D125–D130, 2020, doi: https://doi.org/10.12962/j23373539.v9i2.50465.

[17] N. Ramadhania and N. Rani. *Optimasi Rute Pengiriman Menggunakan Kombinasi Algoritma Genetika dan Tabu Search pada Travelling Salesman Problem.* J. Ilm. Teknol. Inf. Asia, vol. 15, no. 1, pp. 22–30, 2021, doi: https://doi.org/10.32815/jitika.v15i1.659.

[18] A. A. D. Siboro, D. Simanjuntak, and A. S. Harefa. *Penerapan SMOTE pada klasifikasi data tidak seimbang.* SNISTIK 2024: Prosiding Seminar Nasional Teknologi dan Informatika, 2024, pp. 100–107.

[19] S. M. Chamzah, M. Lestandy, and N. Kasan. *Penerapan SMOTE untuk Imbalance Class pada Data Text Menggunakan KNN.* Syntax J. Inform., vol. 11, no. 2, pp. 56–67, 2022.

[20] I. Mardiana and A. Rizqi. *Analisis Penggunaan Memori pada Implementasi Algoritma Rekursif dan Iteratif dalam Pencarian Jalur Graf*. J. Inform. dan Sist. Cerdas, vol. 5, no. 1, pp. 33–40, 2023, doi: 10.31294/jisc.v5i1.1989.

[21] F. Wulandari and B. Santoso. *Efisiensi Pendekatan Hybrid dalam Optimasi Jalur Menggunakan Heuristik Greedy dan DFS.* J. Teknol. dan Sains Komput., vol. 7, no. 2, pp. 95–102, 2022, doi: 10.21009/jtsk.07205.