Perbandingan Implementasi Evolutionary Algorithm (EPO, FHO, dan CFA) pada Kasus Travelling Salesman Problem untuk Tempat Pariwisata di Surabaya

Christian T.S.L. Chen[1], David Cahyadi[1], Jonathan A. Bevan[1], Williandy Takhta[1], Ariel P. Lesmana[1], Christopher D.A. Poernomo[1], dan Widean Nagari[1]
[1]Departemen Informatika, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Christian T.S.L. Chen (e-mail: christian.t20@mhs.istts.ac.id).

ABSTRACT Traveling is a rapidly growing business around the world, and Indonesia is no exception. In Indonesia, especially Surabaya, the travel and tourism industry has been on the rise in recent years, and it is expected to continue to grow in the coming years. With these improvements, route search for tourism must be efficient and fast, one of the popular solutions right now is Evolutionary Algorithms (EA). Evolutionary algorithms are a type of optimization technique that mimics the process of natural evolution to find solutions to complex problems. One such problem that can be effectively solved using evolutionary algorithms is the Traveling Salesman Problem (TSP). This problem involves visiting a specific set of cities and finding the shortest route back to the starting point. Several evolutionary algorithms have been proposed to solve TSP, such as Cuttlefish Algorithm (CFA), Emperor Penguin Optimizer (EPO), and Fire Hawk Optimizer (FHO). The cuttlefish algorithm is based on the behavior of wild cuttlefish, the EPO is inspired by the huddling behavior of emperor penguins, whereas FHO uses the principle of fire propagation. These algorithms have the potential to solve TSP using their own uniqueness. Our conclusion for all those algorithms that are used in this research is that EPO manages to find the best solution while CFA and FHO solutions being slightly below EPO. From our research, EPO is 39.97% better than CFA and 14.75% better than FHO. EPO also has better computation time which is 69.59% faster than CFA and 178.34% faster than FHO.

KEYWORDS Cuttlefish Algorithm, Emperor Penguin Optimizer, Fire Hawk Optimizer, Travelling Salesman Problem

ABSTRAK Traveling merupakan bisnis yang tumbuh pesat di seluruh dunia, dan Indonesia tidak terkecuali. Di Indonesia, khususnya Surabaya, industri pariwisata telah mengalami peningkatan dalam beberapa tahun terakhir, dan diharapkan akan terus tumbuh dalam beberapa tahun ke depan. Dengan peningkatan tersebut, pencarian rute untuk pariwisata harus efisien dan cepat, salah satu solusi yang populer saat ini adalah *Evolutionary Algorithms* (EA). Algoritma evolusi adalah jenis teknik optimisasi yang meniru proses evolusi alami untuk menemukan solusi terhadap masalah yang kompleks. Salah satu permasalahan yang dapat diselesaikan dengan efektif menggunakan algoritma evolusi adalah *Traveling Salesman Problem* (TSP). Permasalahan tersebut melibatkan pengunjungan pada beberapa kota dan menemukan rute terpendek untuk kembali ke titik awal. Beberapa algoritma evolusi telah dicadangkan untuk menyelesaikan TSP, seperti algoritma Cuttlefish (CFA), Emperor Penguin Optimizer (EPO) dan Fire Hawk Optimizer (FHO). Algoritma sotong didasarkan pada perilaku sotong liar, EPO terinspirasi oleh perilaku berkerumun dari penguin kaisar, sedangkan FHO menggunakan prinsip propagasi api. Semua algoritma yang telah disebutkan tadi memiliki potensi untuk menyelesaikan TSP dengan keunikannya masing-masing. Kesimpulan kami untuk semua algoritma yang digunakan dalam penelitian ini adalah bahwa EPO berhasil menemukan solusi terbaik diikuti dengan solusi dari CFA dan FHO. Berdasarkan hasil percobaan kami, didapatkan EPO menghasilkan solusi 39.97% lebih baik dari CFA serta 14.75% lebih baik dari FHO secara rata-rata. Serta EPO juga memiliki waktu komputasi rata-rata lebih cepat (69.59% lebih cepat dari CFA dan 178.34% lebih cepat dari FHO).

KATA KUNCI *Cuttlefish Algorithm*, *Emperor Penguin Optimizer*, *Fire Hawk Optimizer*, *Travelling Salesman Problem*

I. PENDAHULUAN

Indonesia merupakan negara yang memiliki banyak sekali budaya dan alam. Selain Bali, Jakarta, maupun Yogyakarta, Surabaya juga tidak kalah hebat. Surabaya melalui program non-profit pemerintah yang bergerak dalam bidang branding marketing dengan nama Sparkling Surabaya, melakukan publikasi tentang tempat pariwisata yang terdapat di Surabaya.

Dengan berkembangnya jumlah tempat pariwisata di Surabaya maka proses penemuan jalan yang terbaik dan efisienpun juga perlu untuk diteliti lebih lanjut. Algoritma–algoritma untuk menemukan jalan yang terbaik dan efisien tersebut juga mulai berkembang dengan sangat cepat, salah satu perkembangan dari algoritma–algoritma tersebut adalah Evolutionary Algorithm.

Evolutionary Algorithm merupakan algoritma pencari solusi yang dilakukan secara stokastik yang biasanya ditemukan atau diimplementasikan dari evolusi biologis[1]. Sedangkan menurut Sloss, EA itu merupakan Algoritma Evolusi yang menyediakan kerangka kerja dimana kerangka tersebut dapat digunakan kembali di berbagai domain, mereka sebagian besar merupakan algoritma yang terinspirasi dari biologi yang berada sebagai cabang dari Intelijen Buatan[2]. Bartz-Beielstein mengatakan bahwa Algoritma Evolusi dipahami sebagai algoritma pencarian stokastik berbasis populasi yang dalam beberapa hal menirukan proses evolusi alami yang terjadi di alam. Proses tersebut dapat digunakan untuk menemukan solusi optimal dari suatu masalah yang diberikan. EA dapat digunakan untuk menyelesaikan berbagai masalah yang beragam, mulai dari masalah yang memiliki struktur yang sangat kompleks hingga masalah yang relatif sederhana [3].

Dari pernyataan-pernyataan diatas dapat disimpulkan bahwa Algoritma Evolusi merupakan algoritma pencarian solusi stokastik yang didasarkan pada proses evolusi alami yang terjadi di alam. Terinspirasi dari ilmu biologi, Algoritma Evolusi dapat digunakan untuk menyelesaikan berbagai masalah yang memiliki struktur yang kompleks atau yang relatif sederhana. Dengan menggunakan kerangka kerja tersebut, solusi optimal untuk masalah yang diberikan dapat ditemukan. EA telah banyak digunakan dalam berbagai domain, terutama di bidang Intelijen Buatan.

Evolusi biologis merupakan salah satu hal yang sangat menunjukkan bagaimana mahluk hidup dapat mengoptimasi dirinya dengan perubahan agar dapat menjadi mahluk hidup yang bisa bertahan hidup sampai sekarang. Optimasi–optimasi diimplementasikan dalam sebuah algoritma algoritma tersebut yang diharapkan dapat mencari solusi dari sebuah problem yang lebih kompleks seperti Travelling Salesman Problem.

Travelling Salesman Problem (TSP) merupakan salah satu masalah yang sangat mudah untuk dijelaskan namun sangat susah untuk disesaikan. Problem dari TSP adalah bagaimana cara agar travelling salesman dapat mengunjungi semua kota yang ada pada permasalahan dan dapat kembali ketempat awal dengan jarak terdekat atau dengan efisien[4].

Travelling Salesman Problem menurut Hashim merupakan masalah optimasi kombinatorial klasik yang berurusan dengan menemukan jarak atau waktu terpendek untuk tur tertutup di situasi kota n, di mana setiap kota dikunjungi tepat sekali sebelum kembali ke titik awal[5] dan menurut Nurdiawan bahwa Travelling Salesman Problem itu adalah optimasi kombinatorial[6].

II. PENELITIAN TERDAHULU

Travelling Salesman Problem (TSP) merupakan problem mudah untuk dijelaskan namun sangat sulit untuk diselesaikan. Tujuan dari TSP adalah untuk menemukan rute terpendek atau paling efisien yang mengunjungi setiap kota dalam masalah tersebut tepat sekali dan kembali ke titik awal. Hashim mengatakan bahwa TSP harus memiliki setiap kota dikunjungi sekali sebelum kembali ke titik awal, dan Nurdiawan mengatakan itu adalah optimasi kombinatorial. Masalah yang dialami TSP sangat sulit untuk diselesaikan karena seringkali memerlukan menemukan rute paling efisien dengan jarak terpendek.

Beberapa penelitian mengenai TSP telah dilakukan oleh beberapa ahli seperti Hashim yang melakukan penelitian TSP untuk melakukan optimasi kepada perjalanan untuk mengunjungi tempat wisata di Langkawi, salah satu distrik/pulau di Malaysia. Hashim menggunakan software LINGO versi 12.0 yang merupakan software matematika yang dapat membantu menyelesaikan problem TSP[5].

Nurdiawan et al., sebelumnya juga pernah melakukan penelitian mengenai permasalahan TSP untuk melakukan proses penjadwalan paket touring menggunakan Genetic Algorithm. Nurdiawan menggunakan Genetic Algorithm untuk menyelesaikan permasalahan TSP[6]. Penelitian tersebut berhasil menemukan rute paling optimal untuk pariwisata antar kota Indramayu, Cirebon, Majalengka, dan Kuningan. Nurdiawan et al. kemudian menyimpulkan bahwa Genetic Algorithm merupakan algoritma yang sangat cocok untuk menyelesaikan TSP, namun sangat dependen terhadap mekanisme persilangan dan mutasi.

Penelitian lain yang menggunakan Genetic Algorithm untuk menyelesaikan TSP pernah dilakukan oleh Hanif Khan et al.[7]. Permasalahan TSP dengan Time Windows merupakan salah satu bentuk pengembangan masalah TSP, di

mana pada permasalahan tersebut ditambahkan beberapa hal yang menyusahkan penyelesaian masalah, yaitu unsur waktu. Contohnya adalah ditambahkannya waktu pergi dari 1 tempat ke tempat lainnya dan waktu dari tempat wisata yang akan dikunjungi. Unsur waktu juga harus menjadi faktor pertimbangan komputasi GA untuk mendapatkan hasil terbaik. Juwairah et al[8]. menyimpulkan bahwa permasalahan TSP yang dimodifikasi dengan penambahan Time Windows pada rute tempat wisata D.I. Yogyakarta masih dapat diselesaikan menggunakan GA.

Sembiring et al., mengimplementasi TSP dengan menggunakan Ant Colony Optimization (ACO) [9]. Implementasi TSP tersebut digunakan untuk mendapatkan rute terpendek transportasi bahan mentah. ACO menggunakan konsep dari semut ketika mencari makanan dalam suatu colony. Salah satu yang mempengaruhi semut adalah pheromone yang juga digunakan sebagai parameter dalam algoritma ACO. Algoritma ACO terbukti untuk membuat rute terpendek sehingga menghemat jarak tempuh sebesar 37%.

Beberapa penelitian diatas dapat menunjukkan bahwa penelitian sejenis TSP belum pernah menggunakan dengan tiga metode EA yang akan dibahas. Kasus pada tempat wisata Surabaya sangatlah penting untuk diselesaikan agar turis dapat memiliki . Salah satu cara untuk mendapatkan hasil yang efisien dalam permasalahan TSP yaitu adalah dengan menggunakan Evolutionary Algorithm. Penelitian ini akan menggunakan tiga algoritma EA yaitu Cuttlefish Algorithm (CFA)[10], Emperor Penguin Optimizer (EPO)[11], dan Fire Hawk Optimizer (FHO)[12].

Tujuan dari penelitian ini adalah membandingkan tiga algoritma tersebut dan mencari algoritma terbaik yang dapat menghasilkan solusi terbaik untuk Travelling Salesman Problem (TSP) yang diimplementasikan kepada tempat Pariwisata di Surabaya.

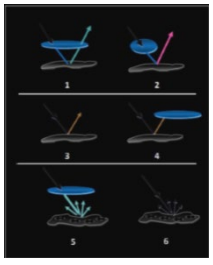
III. ALGORITMA CUTTLEFISH (CFO)

Cuttlefish atau sotong merupakan hewan yang hidup di perairan seperti danau, sungai, dan juga laut[13]. Binatang tersebut memiliki kemampuan untuk mengubah warna dari kulitnya agar dapat melakukan kamuflase dengan lingkungan yang ada disekitarnya[14]. Algoritma cuttlefish dibuat dengan inspirasi bagaimana sebuah sotong melakukan proses mimikri atau mengubah warna kulitnya sesuai dengan tempat atau lingkungannya[15]. Sotong akan melakukan mimikri dengan cara menyesuaikan cahaya yang masuk ke dalam tubuhnya lalu merefleksikannya agar sesuai dengan lingkungannya. Algoritma cuttlefish ditemukan oleh ilmuan bernama Adel Sabry Eesa, Adnan Moshin Abdulazeez Brifcani dan Zeynep Orman[10].

GAMBAR 1. Sotong

Proses kerja dari algoritma mengikuti proses kerja dari sel-sel yang dimiliki oleh kulit sotong, sel-sel yang terdapat pada kulit sotong tersebut ada 3 yaitu Chromatophores, Iridophores, Leucophores dimana sel-sel tersebut akan bekerja secara independen untuk melakukan proses perubahan warna sesuai dengan lingkungannya atau mimikri tersebut, kombinasi dari ketiga sel tersebut akan merefleksikan cahaya yang masuk ke dalam tubuh sotong tersebut. Proses tersebut dapat dinamakan Global Optimum Solution[14].

Algoritma cuttlefish memiliki tujuan untuk mendapatkan bentuk dengan melihat mekanisme perubahan warna kulit yang dilakukan oleh sotong. Algoritma cuttlefish diimplementasikan dengan cara dibagi menjadi 6 bagian yang disebut kasus yang akan dikumpulkan dalam 4 grup. Kasus tersebut adalah perwakilan dari mekanisme perubahan warna kulit dari hewan sotong. Kasus tersebut memiliki 2 proses penting, yaitu reflection dan visibility. Reflection merupakan proses representasi dari mekanisme yang digunakan sotong untuk merefleksikan cahaya yang masuk. Sedangkan visibility adalah representasi dari hasil simulasi bentuk dan pola yang ditunjukkan ke dalam lingkungannya.



GAMBAR 2. Kasus pada Algoritma Cuttlefish

Inisialisasi awal adalah langkah yang harus dilakukan pada algoritma Cuttlefish, inisialisasi yang harus dilakukan adalah mengisi populasi dengan solusi acak, dan juga melakukan input nilai pada variabel r1, r2, v1, dan v2. Perhitungan untuk mendapatkan solusi baru akan dilakukan setelah proses input nilai variabel yang telah dilakukan, lalu populasi-populasi tersebut akan dibagi menjadi 4 grup sesuai dengan kasus-kasus yang terdapat pada Gambar 2. Rumus yang digunakan untuk menemukan solusi baru ($newp$) adalah dengan

menjumlahkan hasil dari $reflection$ dan $visibility$ seperti pada (1).

$$newp = reflection + visibility \quad (1)$$

Grup 1 (kasus 1 dan kasus 2) adalah bagian awal perjalanan algoritma, grup 1 adalah proses interaksi antara sel chromatophores dan sel iridophores yang melakukan proses relaksasi dan kontraksi agar dapat melakukan proses refleksi dan relaksasi sesuai dengan lingkungan yang ada di sekitarnya, proses akan direpresentasikan dalam bentuk rumus sesuai dengan (2) dan (3).

$$reflection = R * G_1[i].Points[j] \quad (2)$$

$$visibility_j = V * Best.Points[j] - G_1[i].Points[j] \quad (3)$$

Chromatophores adalah sel yang disimulasikan pada (2) dan (3). R pada (2) merepresentasikan untuk menghitung jarak dari peregangan yang dilakukan oleh otot pada sel tersebut dalam kontraksi atau relaksasi dan V pada (3) merepresentasikan derajat terlihatnya pola tersebut pada hasil akhirnya. Nilai dari R dan V adalah sebagai berikut:

$$R = random() * (r_1 - r_2) + r_2 \quad (4)$$

$$V = random() * (v_1 - v_2) + v_2 \quad (5)$$

Random di sini adalah angka acak antara (0,1). Variabel r_1 dan r_2 merupakan konstanta yang digunakan untuk menentukan interval regangan dari sel chromatophors, sedangkan v_1 dan v_2 adalah konstanta yang digunakan untuk menentukan derajat dari hasil akhir pola yang dimiliki. Nilai dari R atau V pada (4) dan (5) dapat langsung bernilai 1 atau dapat dikalkulasikan.

Grup 2 (kasus 3 dan kasus 4) adalah simulasi dari iridophores yang akan melakukan refleksi cahaya dari luar, dan akan merefleksikan warna tertentu. Iridophores membantu proses menutupi oragan dan direpresentasikan sebagai best solution. Rumus yang digunakan pada grup 2 memiliki perubahan pada reflection dimana rumus tersebut dituliskan menjadi seperti berikut:

$$reflection_j = R * Best.Point[j] \quad (6)$$

Rumus R akan menjadi 1, sedangkan V akan dihitung ulang. Grup 2 akan melakukan proses local search dan menggunakan perbedaan dari best solution dan hasil terbaik sekarang untuk menemukan interval disekitar solusi terbaik sebagai area baru yang terjadi setelah rumus (6).

Grup 3 (kasus 5) adalah representasi dari sel leucophores, sel akan bekerja sebagai cermin. Sel Kemudian akan melakukan refleksi pada cahaya lingkungannya yang masuk ke dalam sel. Untuk menemukan warna yang akan direfleksikan, maka dapat diasumsikan bahwa warna yang datang adalah solusi terbaik (best). Interval yang digunakan adalah interval diantara best untuk menentukan visibility. Rumus yang digunakan pada grup 3 merupakan hasil modifikasi dari (2) dan (3) yaitu sebagai berikut:

$$reflection_j = R * Best.Points[j] \quad (7)$$

$$visibility = V * (Best.Points[j] - AV_{Best}) \quad (8)$$

AV_{Best} merupakan nilai dari rata-rata best points. Nilai dari R pada rumus adalah 1, sedangkan V akan dilakukan kalkulasi ulang. Hasil rumus modifikasi dari (2) adalah (7) dan hasil modifikasi dari (3) adalah (8).

Grup 4 (kasus 6) adalah grup terakhir dari algoritma cuttlefish yaitu sel leucophores akan melakukan refleksi cahaya dari lingkungannya. Hal ini akan membantu proses sotong untuk mengubah warna dari kulitnya agar sesuai dengan lingkungannya. Simulasi yang dapat diasumsi dari kasus 4 adalah representasi dari solusi acak yang baru. Inisialisasi nilai acak akan digunakan pada kasus 4.

IV. ALGORITMA EMPEROR PENGUIN OPTIMIZER (EPO)

Emperor Penguin Optimizer adalah sebuah algoritma optimasi yang terinspirasi dari perilaku emperor penguin [11]. Algoritma EPO menggunakan kombinasi dari eksplorasi dan eksploitasi untuk melakukan pencarian solusi optimal pada sebuah masalah yang sedang dihadapi. Algoritma ini menggunakan sebuah grup artificial penguin, yang setiap penguinnya adalah hasil representasi dari sebuah potensi dari solusi masalah tersebut. Simulasi perilaku dari penguin dilakukan dengan meniru etika penguin dalam melakukan pencarian makanan pada kondisi ekstrim Antartika. Algoritma EPO dapat melakukan adaptasi pada kondisi dan juga dapat melakukan proses hingga konvergen untuk mendapatkan solusi terbaik dengan lebih cepat. Sehingga mendapatkan posisi penguin (solusi) yang sesuai dengan memiliki nilai fitness yang tinggi.

GAMBAR 3. Emperor Penguin

Algoritma EPO akan dibagi menggunakan fase-fase yang disesuaikan dengan perilaku emperor penguin saat melakukan proses bertahan hidup di kondisi ekstrim yang terdapat pada pedalaman antartika. Fase-fase yang akan dilakukan saat melakukan kerumunan adalah membuat dan menentukan

batasannya, lalu melakukan kalkulasi dari batasannya, menentukan jarak antara penguin, dan mengubah posisi dari effective mover. Fitur yang penting dalam proses perilaku gerombolan adalah setiap penguin mendapatkan kesempatan yang sama untuk mendapatkan kehangatan.

Algoritma EPO adalah salah satu implementasi dari hasil model matematika pada perilaku gerombolan emperor penguin. Pertama, para emperor penguin akan melakukan inisialisasi dari jumlah gerombolan secara acak. Lalu, temperatur dari gerombolan tersebut akan dihitung. Jarak antara emperor penguin juga akan dikalkulasikan agar dapat membantu proses eksplorasi dan juga eksploitasinya. Effective mover akan didapatkan setelah semua proses tersebut telah dijalankan dan optimal solution tersebut ditemukan dan akan melakukan proses perhitungan ulang dari batasan gerombolan dan juga posisi dari emperor penguin yang sudah diperbaharui apabila memiliki kehangatan yang lebih baik dari sebelumnya. Proses-proses implementasi mengenai EPO akan dibahas lebih jauh pada subbab berikutnya.

Emperor penguin melakukan proses bergerombol agar dapat menyimpan energi dan memaksimalkan temperatur dari gerombolan tersebut. Implementasi model matematis dari proses adalah melakukan asumsi bahwa temperatur $T = 0$ saat radius dari poligon $R > 0.5$ dan temperatur $T = 1$ saat radius menjadi $R \leq 0.5$. Temperatur T mengatur proses eksplorasi dan ekploitasi pada emperor penguin saat lokasinya berbeda. Rumus temperatur saat berada di dalam gerombolan T'adalah sebagai berikut:

$$T' = \left(T - \frac{Max_{iteration}}{x - Max_{iteration}}\right) \quad (9)$$

$$T = \begin{Bmatrix} 0, \ if \ R > 0.5 \\ 1, \ if \ R <= 0.5 \end{Bmatrix}$$

Di mana x pada (9) adalah iterasi saat ini, $Max_{iteration}$ merupakan nilai iterasi maksimal, R adalah radius, dan T adalah waktu untuk menentukan solusi optimal dalam search space.

Perhitungan jarak antar penguin dan perhitungan untuk mendapatkan optimal solusi terbaik dilakukan setelah mendapatkan batasan dari gerombolan. Solusi optimal saat ini adalah solusi dari nilai fitness yang paling tinggi atau posisi dengan temperatur yang paling hangat. Emperor penguin yang lain akan melakukan proses perpindahan sesuai dengan solusi optimal terbaik dengan rumus berikut:

$$\vec{D}_{ep} = Abs\left(S(\vec{A}).\vec{P}(x) - \vec{C}.\vec{P}_{ep}(x)\right) \quad (10)$$

$\vec{D}_{ep}$ adalah representasi dari jarak antara penguin dan penguin yang memiliki nilai fitness terbaik, sedangkan x pada (10) adalah iterasi saat ini. Vektor $\vec{A}$ dan $\vec{C}$ digunakan untuk menghindari tabrakan antara sesama penguin. Vektor $\vec{P}$ menotasikan emperor penguin dengan solusi paling optimal (fitness tertinggi). Terakhir, vektor $\vec{P}_{ep}$ merupakan vektor posisi pada iterasi x dari setiap emperor penguin. Fungsi$S()$ merepresentrasikan nilai dorongan yang dimiliki oleh emperor penguin untuk bergerak mendekati emperor penguin yang memiliki solusi terbaik. Fungsi S() dapat dihitung dengan rumus (14). Vektor$\vec{A}$ dan $\vec{C}$ dapat dihitung dengan rumus berikut:

$$\vec{A} = \left(M * \left(T' + P_{grid}(Accuracy)\right) * Rand()\right) - T' \quad (11)$$

$$P_{grid}(Accuracy) = Abs\left(\vec{P} - \vec{P}_{ep}\right) \quad (12)$$

$$\vec{C} = Rand() \quad (13)$$

Pada rumus (11), M adalah parameter yang menggambarkan nilai pergerakan untuk menjaga jarak antara penguin agar tidak terjadi tabrakan. Nilai M akan diisi sebesar 2. T' adalah nilai temperatur di sekitar gerombolan. $P_{grid}(Accuracy)$ adalah akurasi dari poligon yang dapat dihitung dengan melakukan perbandingan lokasi setiap emperor penguin dengan emperor penguin paling optimal. Perbandingan tersebut dirumuskan pada rumus (12). $Rand()$ pada rumus (13) adalah nilai acak antara [0, 1].

$$S\left(\vec{A}\right) = \left(\sqrt{f \, . e^{-x/l} . - e^{-x}}\right)^2 \quad (14)$$

Pada rumus (14), e adalah fungsi ekspresi. Sedangkan notasi f dan l adalah parameter kontrol untuk mendapatkan hasil eksplorasi dan eksploitasi yang baik. f dan l masing-masing memiliki yang berada pada rentang [2, 3] dan [1.5, 2] secara berurutan. Nilai tersebut telah terbukti untuk mendapatkan nilai yang lebih baik dalam menemukan solusi.

Fase terakhir dalam algoritma EPO adalah melakukan relokasi dari emperor penguin yang dilakukan berdasarkan hasil kalkulasi sebelumnya dan disesuaikan dengan hasil terbaik dari solusi optimal. Emperor penguin yang terbaik akan bertanggung jawab untuk melakukan perubahan posisi dari emperor penguin lainnya. Rumus yang digunakan untuk melakukan perubahan adalah sebagai berikut:

$$\overrightarrow{P_{ep\_new}} = \vec{P}(x) - \vec{A} \, . \vec{D}_{ep} \quad (15)$$

$$\overrightarrow{P_{ep}}(x+1) = \begin{Bmatrix} \overrightarrow{P_{ep\_new}}, \ if \ \overrightarrow{P_{ep}}(x) < \overrightarrow{P_{ep\_new}} \\ \overrightarrow{P_{ep}}(x), \ if \ \overrightarrow{P_{ep}}(x) \geq \overrightarrow{P_{ep\_new}} \end{Bmatrix} \quad (16)$$

Di mana rumus (16) merepresentasikan posisi terbaru dari emperor penguin yang telah melakukan perubahan. Menghitung Pep baru dapat dilakukan dengan menggunakan rumus (15). Pada rumus (15), nilai P(x) didapatkan dari penguin terbaik pada iterasi sebelumnya, sedangkan A

bersumber dari rumus (11), dan Dep dari rumus (10). Posisi terbaru dari emperor penguin ditentukan dari hasil fitness emperor penguin dengan posisi lama dibandingkan dengan yan baru. Posisi emperor penguin akan terubah hingga iterasi mencapai maksimal iterasi.

V. ALGORITMA FIRE HAWK OPTIMIZER (FHO)

Fire Hawk merupakan algoritma metaheuristik yang dibuat dengan inspirasi perilaku berburu dari beberapa burung seperti whistling kites, black kites, dan brown falcons. Burung – burung tersebut dinamai Fire Hawk dikarenakan terdapat aksi spesifik yang dilakukan dalam melakukan proses pencarian mangsa yang dilakukan di habitatnya, yang disebut dengan menyalakan api [12]. Algoritma Fire Hawks dengan sengaja mencoba untuk menyalakan api dengan membawa ranting yang terbakar pada paruhnya, yang dilaporkan sebagai fenomena penghancuran di alam.

GAMBAR 4. Fire Hawk Disekitar Api

Sebagai mekanisme untuk melakukan kontrol dan mendapatkan mangsanya, burung dapat mengambil ranting untuk membakar sebagian kecil dari lahan untuk menakuti mangsanya seperti tikus, ular, dan hewan lainnya. Hewan yang kabur akan dengan tergesa-gesa berlarian yang membuat hewan-hewan tersebut sangat mudah ditangkap oleh hawks. Fire Hawk Optimizer akan melakukan inisialisasi dengan cara menentukan jumlah kandidat solusi yang akan menentukan posisi vektor dari mangsa dan fire hawks. Inisialisasi acak akan dilakukan saat menentukan posisi awal dari vektor-vektor tersebut.

$$X = \begin{bmatrix} X_1 \\ X_i \\ X_N \end{bmatrix} = \begin{bmatrix} X_1^1, X_1^2, X_1^j \dots X_1^d \\ X_i^1, X_i^2, X_i^j \dots X_i^d \\ X_N^1, X_N^2, X_N^j \dots X_N^d \end{bmatrix}, \begin{Bmatrix} i = 1, 2, \dots, N. \\ j = 1, 2, \dots, d. \end{Bmatrix} \quad (17)$$

Rumus yang digunakan untuk mencari X pada (17) adalah sebagai berikut:

$$X_i^j(0) = X_{i,min}^j + rand.\left(X_{i,max}^j - X_{i,min}^j\right), \begin{Bmatrix} i = 1, 2, \dots, N. \\ j = 1, 2, \dots, d. \end{Bmatrix} \quad (18)$$

Pada rumus (18), X_i merupakan solusi ke-i dalam lingkup pencarian; variable d merupakan representasi dari dimensi dalam masalah yang sedang dicari; N adalah total dari solusi yang merupakan kandidat dalam lingkup pencarian; X_i^j merupakan variabel penentu ke-j dari solusi ke-i; $X_{i,max}^j$ dan $X_{i,min}^j$ merupakan batasan maksimal dan minimal dari variabel penentu ke-j dan kandidat solusi ke-I; $rand$ adalah nilai acak yang memiliki rentang antara 0 sampai 1.

Untuk mendapatkan lokasi dari Fire Hawk yang terdapat pada lingkup pencarian, Representasi dari Fire Hawk akan digunakan jika kandidat solusi yang memiliki fungsi objektif yang lebih baik, sedangkan representasi dari mangsa adalah semua selain Fire Hawk tersebut.

Setelah melakukan inisialisasi tersebut, maka perhitungan untuk menentukan jumlah dari jarak antara Fire Hawk dan mangsa-mangsanya. Hasil dari perhitungan tersebut adalah menemukan mangsa terdekat dari setiap burung, yang akan membantu untuk menemukan jangkauan efektif dari burung-burung tersebut. Rumus yang digunakan untuk mencari jarak adalah sebagai berikut:

$$D_k^l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \begin{Bmatrix} l = 1,2, \dots, n. \\ k = 1,2, \dots, m. \end{Bmatrix} \quad (19)$$

Pada rumus (19), D_k^l merupakan total jarak antara Fire Hawk ke-l dan mangsa ke-k; m adalah jumlah dari semua mangsa yang ada dalam lingkup pencarian; n adalah jumlah dari semua fire hawk yang terdapat pada lingkup pencarian; dan (x_1, y_2) dan (x_2, y_2) adalah representasi dari koordinat Fire Hawks dan juga mangsa yang terdapat pada lingkup pencarian.

Setelah menemukan jarak dari Fire Hawks dan juga mangsanya, proses selanjutnya adalah untuk melakukan pengumpulan ranting yang terbakar, dengan cara mengambil api dari titik yang menjadi pusat dari api agar dapat melakukan pembakaran pada tempat yang telah ditentukan. Pada proses ini burung akan melakukan pengambilan ranting terbakar lalu menjatuhkannya ditempat yang telah ditentukan yang akan membuat mangsa kabur secara tergesa-gesa. Rumus yang digunakan untuk melakukan update posisi pada proses ini ialah sebagai berikut:

$$FH_l^{new} = FH_l + (r_1 * GB - r_2 * FH_{Near}), l = 1,2,..,n \quad (20)$$

Pada rumus (20), FH_l^{new} adalah representasi dari posisi baru dari Fire Hawk ke-l; GB merupakan solusi terbaik atau Global Best dari lingkup pencarian; FH_{Near} adalah salah satu Fire Hawk yang terdapat pada lingkup pencarian; dan r_1 dan $r_2$adalah nulai acak yang berada pada nilai antara (0, 1) untuk menentukan pergerakan Fire Hawks kearah pusat api dan kearah Fire Hawk lainnya.

Proses selanjutnya dari algortima FHO adalah pergerakan mangsa yang berada pada wilayah dari setiap Fire Hawk, hal ini merupakan hal yang penting dari mangsa untuk melakukan perubahan posisi. Saat Fire Hawk melakukan pembakaran dengan menjatuhkan ranting disekitar wilayah mangsa, mangsa tersebut dapat memilih untuk bersembunyi, kabur,

atau berlari kearah Fire Hawk karena panik. Aksi tersebut dapat dirumuskan dalam bentuk sebagai berikut:

$$PR_q^{new} = PR_q + (r_3 * FH_l - r_4 * SP_1), \begin{Bmatrix} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{Bmatrix} \quad (21)$$

Pada rumus (21), PR_q^{new} merupakan posisi terbaru dari mangsa ke-q yang dikelilingi oleh Fire Hawk (FH_l); SP_1 merupakan tempat yang aman saat dikelilingi oleh Fire Hawk ke-l; r_3 dan r_4 merupakan nilai acak yang berada diantara (0, 1) untuk menentukan pergerakan dari mangsa tersebut.

Mangsa memiliki kemungkinan untuk melakukan perpindahan menuju ke wilayah Fire Hawk lainnya. Berikut merupakan rumus yang menggambarkan perubahan tempat mangsa:

$$PR_q^{new} = PR_q + (r_5 * FH_{Alter} - r_6 * SP), \begin{Bmatrix} l = 1,2,\dots,n. \\ q = 1,2,\dots,r. \end{Bmatrix} \quad (22)$$

Pada rumus (22), PR_q^{new} merupakan posisi terbaru dari mangsa ke-q; FH_{Alter} merupakan Fire Hawk lainnya yang terdapat pada lingkup pencarian ini; dan r_5 dan r_6 merupakan nilai acak yang berada pada nilai antara (0, 1) untuk menentukan pergerakan dari mangsa tersebut.

Safe Place adalah tempat yang aman yang dapat dikunjungi oleh mangsa dan dalam algoritma SP_1 dan SP dirumuskan sebagai berikut:

$$SP_1 = \frac{\sum_{q=1}^{r} PR_q}{r}, \begin{Bmatrix} q = 1,2,\dots,r. \\ l = 1,2,\dots,n. \end{Bmatrix} \quad (23)$$

$$SP = \frac{\sum_{k=1}^{m} PR_k}{m}, k = 1,2,\dots,m. \quad (24)$$

Pada rumus (23, 24), PR_q adalah mangsa ke-q yang sedang dikelilingi oleh Fire Hawk ke-l; PR_k adalah mangsa ke-k yang berada pada lingkup pencarian.

VI. DATA

Data yang digunakan dalam penelitian adalah lokasi tempat wisata yang berada pada Surabaya, serta beberapa hotel bintang empat dan lima yang ditambahkan sebagai tempat penginapan. Data tersebut didapatkan dengan menggunakan Open Road Service, dimana website tersebut menyediakan API yang dapat digunakan untuk membantu proses pathfinding atau penemuan jalan dari satu titik ke titik yang lain, lalu untuk data gambar map yang digunakan didapatkan dengan menggunakan Open Street Map. Data lain yang digunakan pada penelitian lain adalah data hotel bintang empat dan lima. Data tersebut didapatkan dengan library bantuan library bantuan bernama Beautiful Soup, yang merupakan sebuah library yang dapat digunakan dengan bahasa pemrograman python untuk melakukan proses parsing data HTML atau XML. Lokasi-lokasi pariwisata merupakan data yang terpenting dalam penelitian, data tersebut didapatkan dengan menggunakan website yang telah disediakan oleh pemerintah yaitu website tourism Surabaya.

Data lokasi tempat wisata yang telah didapatkan tersebut akan melalui proses pembersihan data, dikarenakan beberapa tempat wisata yang tidak relevan untuk penelitian. Data preprocessing akan melakukan proses seperti membersihkan data-data yang kurang relevan seperti "bangunan kosong", "apotek", "CIMB NIAGA". Karena tidak sesuai dengan tema pada penelitian yaitu tempat wisata maka dari-data yang dikumpulkan dibersihkan terlebih dahulu sebelum digunakan dalam program. Data yang dikumpulkan sebelum pembersihan data atau preprocessing adalah 447 data lokasi tempat wisata, lalu setelah dilakukan proses pembersihan data, jumlah data berkurang menjadi 123 data lokasi wisata.

VII. IMPLEMENTASI ALGORITMA

Algoritma-algoritma diatas akan diimplementasikan pada program yang telah dibuat khusus untuk memecahkan masalah dengan bantuan hasil representasi dari data yang telah disediakan, algoritma-algortima akan mengambil 3 hasil terbaik dengan nilai yang terendah. Implementasi dari program-program tersebut akan dijelaskan lebih lanjut pada subbab-subbab selanjutnya.

GAMBAR 5. Representasi Solusi

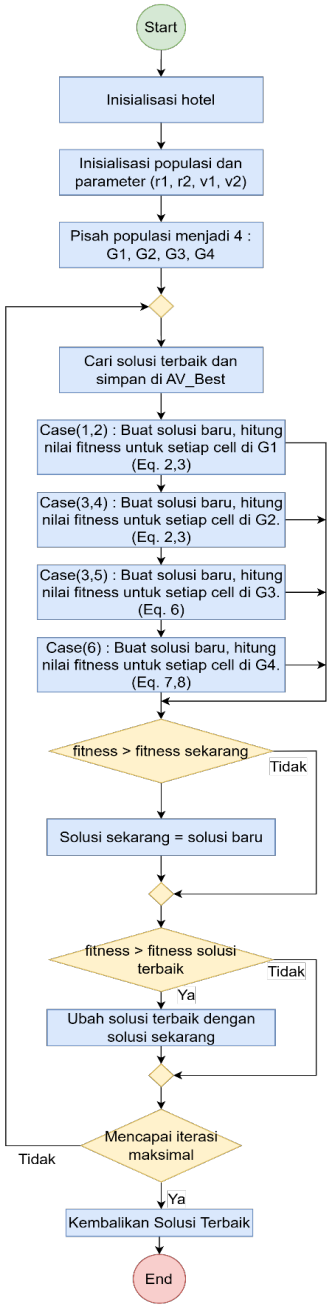
A. REPRESENTASI SOLUSI

Pada implementasi tiga algoritma yang dijelaskan pada bab ini akan menggunakan representasi solusi yang sama. Representasi digambarkan pada Gambar 5. Representasi berupa sebuah array satu dimensi dimana setiap elemennya merupakan suatu ID dari lokasi wisata. Jumlah dari elemen array merupakan jumlah semua tempat wisata yang ada yaitu 123 tempat wisata yang terletak pada surabaya. Index pada array akan merepresentasikan urutan tempat wisata yang harus dikunjungi. Ketika fitness dihitung, array akan dibagi berdasarkan jumlah hari dan kunjungan tempat lokasi wisata per hari.

Pada awalnya pembagi dari array akan ditentukan berdasarkan jumlah kunjungan per hari. Namun ketika jumlah pengalian dari hari dan kunjungan per hari lebih dari jumlah tempat wisata maka pembagi akan didapatkan dari pembagian jumlah tempat wisata dengan kunjungan per hari. Namun jumlah hari pada input program tidak dapat melebihi dari jumlah tempat wisata pada data dikarenakan hanya akan

mengurutkan tempat wisata kota yang terdekat dari tempat tinggal dimana hal tersebut kurang sesuai dengan tujuan penelitian.

Representasi yang berupa array satu dimensi dapat divisualisasikan sebagai array dua dimensi yang merupakan alat bantu untuk memudahkan ketika menghitung nilai fitnessnya. Dimensi pertama merupakan jumlah hari yang berasal dari input parameter. Dimensi kedua merupakan lokasi wisata yang hendak dikunjungi. Pada implementasinya akan memperhitungkan hotel sebagai perantara setiap harinya dengan anggapan bahwa wisatawan akan selalu berangkat dari hotel dan pulang ke hotel setiap harinya. Cara kerja dari fitness function akan dijelaskan lebih lanjut pada bagian E.



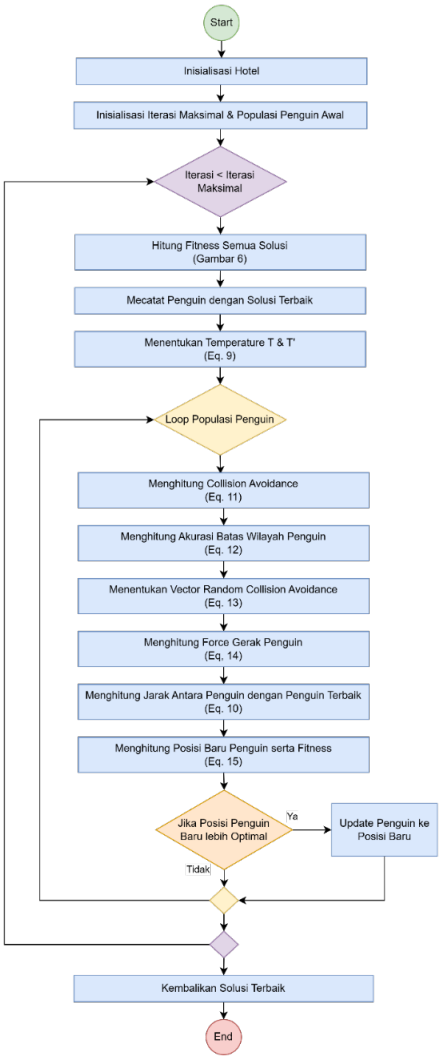
GAMBAR 6. Diagram Alur Cuttlefish Algorithm

B. CUTTLEFISH ALGORITHM

Algoritma cuttlefish diinisialisasi dengan mengatur semua solusi yang dimiliki untuk dibagi ke dalam kelompok-kelompok yang telah ditentukan, yaitu kelompok 1, kelompok 2, kelompok 3, dan kelompok 4. Setelah menentukan grup-grup tersebut lalu parameter-parameter yang diperlukan seperti r1, r2, v1 dan v2. Kelompok 1 akan menjadi representasi dari proses sotong untuk melakukan proses refleksi dari cahaya yang masuk, kelompok 2 adalah proses visibility dimana pada kelompok ini menentukan apakah cahaya akan diserap kembali atau dipantulkan. Kelompok 3 adalah proses mendapatkan mendapatkan reflection dari cahaya yang terdapat pada area sekitar best solution, dan kelompok 4. Loop akan dilakukan selama stopping condition belum terpenuhi, hal yang pertama terjadi saat melakukan proses looping adalah mencari average best dari semua solusi yang ada, lalu proses selanjutnya adalah melakukan proses-proses tertentu sesuai dengan kelompok yang telah dibagi pada proses awal. Program akan berhenti setelah stopping condition tercapai dan akan mengambil 3 nilai terendah untuk menjadi hasil terbaik sebagai output algoritma cuttlefish. Visualisasi dalam bentuk diagram terdapat pada Gambar 6.

C. EMPEROR PENGUIN OPTIMIZER

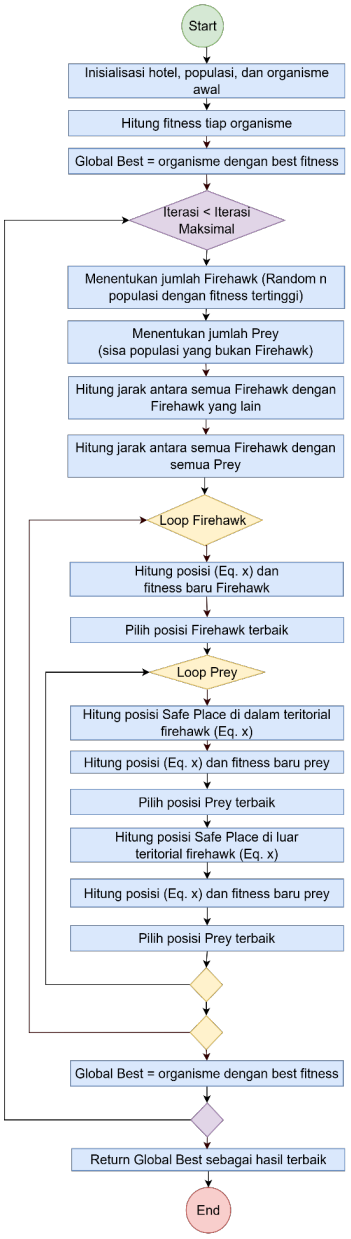
EPO diawali dengan mengatur beberapa parameter yang dibutuhkan, seperti maksimal iterasi, maksimal penguin, konstanta f, M dan l serta batas bawah dan batas atas. Pada EPO, seekor penguin merepresentasikan sebuah solusi yang merupakan sebuah array satu dimensi yang berisi sejumlah lokasi wisata yang akan dikunjungi. Algoritma akan berjalan selama iterasinya belum menyentuh batas maksimal iterasi yang ditentukan. Setiap iterasinya diawali dengan menghitung fitness untuk setiap penguin yang ada dan mencari penguin dengan fitness terbaik. Setelah itu loop untuk setiap penguin yang ada dan hitung fitness baru untuk penguin tersebut. Apabila fitness yang baru lebih baik daripada yang lama, posisi penguin akan tergantikan dengan yang baru sesuai dengan fitness yang terbaik. Posisi penguin juga perlu disesuaikan mengikuti batas bawah dan batas atas yang merupakan parameter di awal program dimana jika melebihi akan digantikan dengan nilai pada parameter tersebut. Setelah algoritma sudah berjalan sebanyak jumlah maksimal iterasi, penguin dengan nilai fitness tinggi akan digunakan sebagai solusi. Visualisasi proses EPO dalam bentuk diagram terdapat pada Gambar 7.



GAMBAR 7. Diagram Alur Emperor Penguin Optimizer

D. FIREHAWK OPTIMIZER

Algoritma FHO diimplementasi dengan menggunakan sejumlah parameter di dalam alur programnya yaitu seed, maksimal iterasi, besar populasi, daftar solusi, daftar lokasi wisata, daftar hotel, solusi pertama, firehawk terbaik, dan configs. Seed merupakan seed untuk random. Maksimal iterasi merupakan jumlah maksimal iterasi. Besar populasi merupakan jumlah kandidat solusi. Daftar solusi digunakan untuk menampung 3 solusi terbaik yang akan dihasilkan. Daftar lokasi wisata digunakan untuk menyimpan daftar tempat parawisata. Daftar hotel digunakan untuk menyimpan daftar hotel yang ada yaitu berjumlah 40 data. Solusi pertama digunakan untuk menampung inisialisasi dari solusi awal. Firehawk terbaik digunakan untuk menampung firehawk terbaik. Visualisasi dalam bentuk diagram terdapat pada Gambar 8.



GAMBAR 8. Diagram Alur Fire Hawk Optimizer

Pertama-tama akan dilakukan inisialisasi posisi dari setiap kandidat solusi dengan fitness awal adalah 0. Kemudian akan dilakukan pengurutan kandidat solusi berdasarkan fitness dengan nilai fitness tertinggi di paling depan. Kandidat solusi berupa sebuah array satu dimensi yang berisi sejumlah lokasi wisata yang akan dikunjungi. Setelah itu akan dilakukan kalkulasi fitness terhadap setiap kandidat solusi. Global best awal akan ditentukan. Bagian-bagian selanjutnya ini akan diiterasikan sebanyak maksimal iterasi. Jumlah firehawks akan diacak dalam rentang 1 hingga besar populasi-1 dan sisanya adalah jumlah preys. Kemudian dari daftar solusi akan ditentukan mana yang merupakan firehawks dan sisanya

merupakan preys. Setelah itu akan dilakukan kalkulasi jarak antara firehawks dan preys. Selanjutnya akan dilakukan kalkulasi jarak antar firehawks. Wilayah yang terdampak firehawks akan ditentukan. Kemudian akan dilakukan pembaruan posisi dan fitness setiap firehawk. Setelah itu untuk setiap prey yang terdampak di wilayah firehawk, dilakukan kalkulasi safe place. Posisi dan fitness setiap prey akan diperbarui. Selanjutnya untuk setiap prey yang diluar wilayah firehawk, dilakukan kalkulasi safe place. Posisi dan fitness setiap prey akan diperbarui. Setelah itu apabila fitness dari firehawk terbaik pada iterasi tersebut lebih baik dari pada global best saat ini, maka global best akan diperbarui. Tiga solusi terbaik kemudian akan didapatkan.

E. FITNESS FUNCTION

Pada bagian ini akan dijelaskan mengenai fitness function yang akan digunakan pada semua algoritma. Fitness function ini digunakan untuk menghitung seberapa dekat antara solusi dan spektrum neutron referensi[16]. Fitness function dihitung dengan menjumlahkan jarak dari hotel yang ditempati menuju ke tempat wisata tujuan. Di awal hari akan menghitung jarak dari hotel ke tempat wisata tujuan dan di akhir hari akan menghitung jarak dari tempat wisata terakhir di hari tersebut ke hotel. Lalu dari total jarak mulai dari hotel pada hari pertama hingga kembali ke hotel pada hari terakhir akan dikali dengan -1 agar semakin besar fitness score semakin baik. Tujuan dari fitness function yang digunakan adalah mendapatkan jarak yang tempuh yang paling rendah untuk mengunjungi tempat pariwisata. Pseudocode untuk rumus fitness function terdapat pada Gambar 9.

```
procedure fitness function ()
  inisialisasi total = 0
  for l=1:n (semua tempat wisata tujuan)
    if l adalah tempat wisata pertama di suatu hari
      total = total + jarak(hotel ke l)
    if l adalah tempat wisata terakhir di suatu hari
      total = total + jarak(l-1 ke l)
      total = total + jarak(l ke hotel)
    else
      total = total + jarak(l-1 ke l)
  end
  total = total * -1
  return total
end procedure
```

GAMBAR 9. Pseudocode Fitness Function

VIII. UJI COBA

Pada bagian ini dijelaskan hasil dari perbandingan implementasi beberapa algoritma untuk menyelesaikan masalah tersebut. Terdapat beberapa variasi uji kasus yang dicoba untuk penelitian dimana terdapat parameter input jumlah hari dan kunjungan per hari, dimana pada kasus pertama menggunakan 30 hari yang setiap harinya mengunjungi 4 tempat, kasus kedua menggunakan 14 hari yang setiap harinya mengunjungi 5 tempat, kasus ketiga menggunakan 7 hari yang setiap harinya mengunjungi 5 tempat, dan kasus keempat menggunakan 3 hari yang setiap harinya mengunjungi 6 tempat. Uji coba pada penelitian ini menggunakan salah satu hotel yang didapatkan yaitu hotel Sheraton yang terletak pada Jl. Embong Malang No. 25-31. Pemilihan hotel tersebut dikarenakan 2 alasan yaitu pertama hotel yang dipilih merupakan hotel bintang lima karena umumnya akan menjadi pilihan bagi banyak turis, alasan kedua, hotel yang dipilih adalah hotel yang berada di daerah tengah kota untuk menjangkau lokasi pariwisata yang lebih beragam. Penelitian ini juga melakukan uji coba terhadap parameter khusus untuk masing-masing algoritma yang ditentukan dan digunakan pada awal algoritma dijalankan.

Pengujian parameter digunakan agar dapat menghasilkan fitness score terbaik dengan kombinasi paramater yang digunakan. Fitness score semakin baik bila nilai yang tertulis semakin besar. Terdapat 4 variasi uji kasus dilakukan pada penelitian seperti yang telah dijelaskan sebelumnya dimana setiap uji kasus memiliki jumlah hari dan jumlah kunjungan per hari yang berbeda. Jumlah kunjungan per hari menunjukkan berapa jumlah tempat yang dikunjungi setiap harinya sedangkan jumlah hari menunjukkan berapa lama perjalanan pariwisata tersebut dilakukan. Pada akhirnya jumlah tempat yang dikunjungi adalah jumlah hari dikali jumlah kunjungan per hari, seperti pada kasus pertama 30 hari dengan 4 kunjungan perhari memiliki arti bahwa jumlah tempat yang akan dikunjungi berjumlah 120. Keempat uji kasus dijelaskan pada Tabel I.

TABEL I.
VARIASI UJI KASUS

No	Jumlah Hari	Jumlah kunjungan per hari
1	30	4
2	14	5
3	7	5
4	3	6

Tentunya terdapat beberapa pengaturan parameter untuk algoritma-algoritma yang dibandingkan pada penelitian. Faktor parameter pada algoritma EPO telah dicoba berdasarkan dengan kombinasi dari nilai konstan f, M, dan l. Algoritma CFA juga memiliki beberapa parameter yang dapat mempengaruhi hasil fitness yaitu r1, r2, v1, dan v2. Sedangkan algoritma FHO tidak memiliki pengaturan parameter yang

dapat dilakukan uji coba. Rentang nilai parameter untuk setiap algoritma dijelaskan pada Tabel II.

TABEL II.
RENTANG NILAI PARAMETER

No	Algoritma	Parameter
1	Emperor Penguin Optimizer (EPO)	*f: [2-3]; M: [1-3]; l: [1.5-2]*
2	Firehawk Optimizer (FHO)	-
3	Cuttlefish Algorithm (CFA)	*r1: [-2-2]; v1: [-2-2]; r2: [-2-2]; v2: [-2-2]*

Pada setiap uji kasus, akan dilakukan menggunakan 250, 500, 1000, dan 1500 iterasi. Namun khusus uji coba pada algoritma FHO, pada percobaan iterasi 1000 dan 1500 hanya disimpan hasil yang memiliki nilai fitness terbaik, sehingga hasil yang memiliki nilai fitness kurang baik tidak tersimpan.

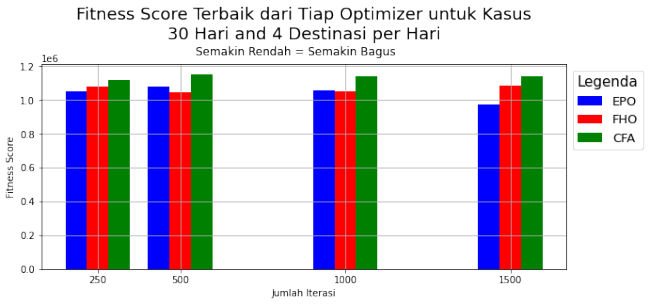
A. UJI KASUS #1

Uji kasus ini akan membahas untuk hari berjumlah 30 dan terdapat maksimal 4 kunjungan per hari. Nilai fitness terbaik yang didapatkan adalah -970963.64 yang diperoleh menggunakan algoritma EPO. Tabel III adalah tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus 30 hari dan maksimal 4 kunjungan per hari.

TABEL III.
FITNESS TERBAIK UJI KASUS #1

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-970963.64	1500	100	*f*: 2.6; *M*: 1; *l*: 2.1
	2	-1053594.97	250	100	*f*: 3; *M*: 1; *l*: 1.9
	3	-1055009.62	1000	100	*f*: 3; *M*: 2; *l*: 1.5
FHO	1	-1043266.04	500	50	-
	2	-1053691.75	1000	50	
	3	-1078990.4	500	50	
CFA	1	-1142067.97	1000	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -2
	2	-1140326.61	1500	100	*r1*: -2; *r2*: -1; *v1*: 1; *v2*: 1
	3	-1145331.48	1500	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: 0

Data pada Tabel III dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 10. Gambar 10 memberikan visualisasi data mengenai perbandingan nilai fitness setiap algoritma berdasarkan jumlah iterasi yang dilakukan. Optimisasi dengan menggunakan EPO memiliki fitness paling baik pada 250 iterasi dan 1500 iterasi, sedangkan optimisasi dengan menggunakan FHO lebih baik pada 500 dan 100 iterasi. Pada uji kasus #1, EPO memiliki performa yang lebih baik sebesar 17,44% dari CFA dan 7,45% lebih baik dari FHO.



GAMBAR 10. Hasil Uji Coba Kasus #1

B. UJI KASUS #2

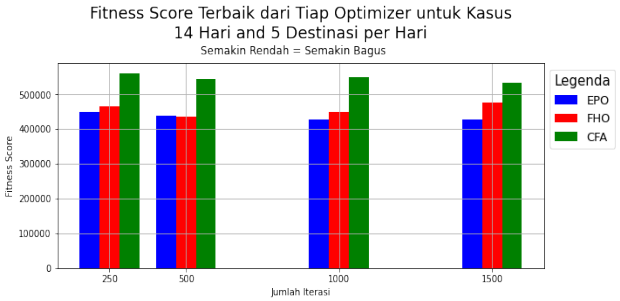
Uji kasus kedua akan membahas kasus di mana jumlah hari sebanyak 14 dan terdapat maksimal 5 kunjungan dalam satu hari. Nilai fitness terbaik yang didapatkan pada uji kasus kedua adalah -426475.63, nilai fitness tersebut diperoleh menggunakan algoritma EPO. Tabel IV merupakan tabel yang menunjukkan nilai fitness terbaik yang diperoleh untuk setiap algoritma pada uji kasus kedua.

TABEL IV.
FITNESS TERBAIK UJI KASUS #2

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-426475.63	1000	100	*f*: 2.1; *M*: 2; *l*: 2.1
	2	-427561.85	1500	100	*f*: 2.4; *M*: 1; *l*: 1.7
	3	-427859.84	1500	100	*f*: 3; *M*: 1; *l*: 1.7
FHO	1	-435565.46	500	50	-
	2	-449880.12	1000	50	
	3	-458713.13	500	50	
CFA	1	-531663.69	1000	100	*r1*: -2; *r2*: -1; *v1*: -1; *v2*: 1
	2	-539349.36	1000	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: -2
	3	-539349.36	500	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: 0

Data pada Tabel IV dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 11. Pada uji coba kasus #2, ditemukan bahwa optimisasi menggunakan EPO unggul pada 250, 1000, dan 1500 iterasi dibandingkan dengan algoritma optimalisasi lainnya. Optimisasi menggunakan FHO unggul pada percobaan dengan 500 iterasi dengan selisih yang kecil. Pada uji kasus #2, EPO menghasilkan performa 24,66% lebih

baik dari CFA dan 2,13% lebih baik daripada FHO. Namun hasil uji coba kedua memperlihatkan bahwa FHO memiliki nilai fitness yang lebih baik jika dibandingkan dengan uji kasus yang pertama.



GAMBAR 11. Hasil Uji Coba Kasus #2

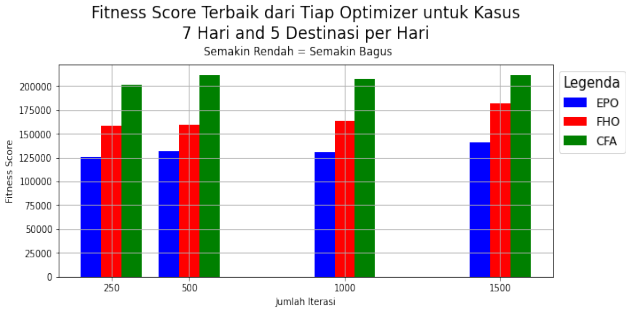
C. UJI KASUS #3

Pada uji kasus ketiga akan membahas di mana jumlah hari sebanyak 7 dan terdapat maksimal 5 kunjungan yang dilakukan dalam satu hari. Nilai fitness terbaik yang didapatkan adalah -125116.57, nilai fitness yang diperoleh tersebut dihasilkan dengan algoritma optimalisasi dengan menggunakan algoritma EPO. Tabel V merupakan tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus ketiga.

TABEL V.
FITNESS TERBAIK UJI KASUS #3

Algoritma	No	Fitness	Iterasi	Pop ulasi	Parameter
EPO	1	-125116.57	250	100	*f*: 2.8; *M*: 1; *l*: 1.5
	2	-130636.13	1000	100	*f*: 2.1; *M*: 2; *l*: 1.8
	3	-131709.49	500	100	*f*: 2.9; *M*: 1; *l*: 1.8
FHO	1	-158777.89	250	50	-
	2	-159058.52	500	50	
	3	-163292.29	1000	50	
CFA	1	-200979.13	250	100	*r1*: -2; *r2*: -2; *v1*: 2; *v2*: -1
	2	-207344.54	1000	100	*r1*: -2; *r2*: -2; *v1*: -1; *v2*: 2
	3	-208092.88	1000	100	*r1*: -2; *r2*: -2; *v1*: -1; *v2*: 1

Data pada Tabel V dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 12. Pada grafik tersebut, optimisasi menggunakan EPO unggul pada seluruh percobaan iterasi (60% lebih baik dari CFA dan 26,9% lebih baik dari FHO). Grafik pada Gambar 12 juga menunjukkan bahwa hasil nilai fitness dari algoritma CFA memiliki hasil yang buruk



GAMBAR 12. Hasil Uji Coba Kasus #3

yaitu nilai fitness functionyang melebihi 500000 pada kasus uji coba ketiga.

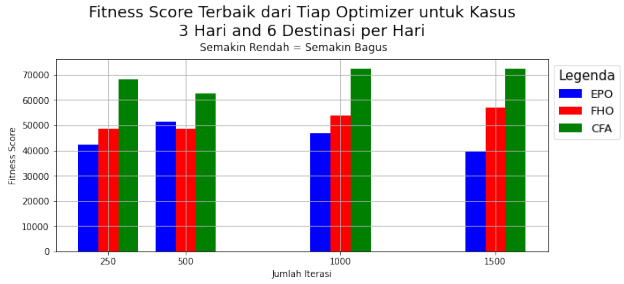
D. UJI KASUS #4

Pada uji kasus keempat akan membahas untuk hari berjumlah 3 dan terdapat maksimal 6 kunjungan dalam satu hari. Nilai fitness terbaik yang didapatkan adalah -39590.81, nilai fitness tersebut diperoleh dengan menggunakan algoritma EPO. Tabel VI merupakan tabel yang menunjukkan nilai fitness terbaik untuk setiap algoritma pada uji kasus keempat.

TABEL VI.
FITNESS TERBAIK UJI KASUS #4

Algoritma	No	Fitness	Iterasi	Populasi	Parameter
EPO	1	-39590.81	1500	100	*f*: 2.2; *M*: 2; *l*: 2.1
	2	-42403.18	250	100	*f*: 2; *M*: 1; *l*: 1.7
	3	-44399.97	250	100	*f*: 2.8; *M*: 1; *l*: 2.1
FHO	1	-48501.43	250	50	-
	2	-48643.86	500	50	
	3	-50825.39	500	50	
CFA	1	-62467.32	500	100	*r1*: -2; *r2*: -1; *v1*: 2; *v2*: 2
	2	-68228.72	250	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -2
	3	-68955.56	500	100	*r1*: -2; *r2*: -2; *v1*: 1; *v2*: -1

Data pada Tabel VI dapat disimpulkan menjadi suatu grafik yang dapat dilihat pada Gambar 13. Percobaan dengan menggunakan EPO memiliki hasil yang jauh lebih baik dibandingkan dengan algoritma lainnya pada 250, 1000, dan 1500 iterasi. Algoritma FHO memiliki hasil yang lebih baik dari optimisasi lainnya pada 500 iterasi. Pada uji kasus keempat, algoritma optimasi EPO memiliki performa 57,78% lebih baik dari CFA dan 22,51% lebih baik dibandingkan dengan algoritma FHO.



GAMBAR 13. Hasil Uji Coba Kasus #4

E. HASIL UJI KASUS

Pada setiap uji kasus yang dilakukan, terdapat pencatatan berapa lama program berjalan dari awal hingga akhir iterasi. Data tersebut dirata-rata dan dimasukkan ke dalam sebuah grafik pada Gambar 15. Kemudian untuk membuktikan hasil rute yang didapatkan berdasarkan fitness score terbaik, maka telah dibuat visualisasi rute pada peta. Seperti yang tergambar pada Gambar 14, dapat terlihat rute pada hari yang sama sudah terkumpul di area yang berdekatan.

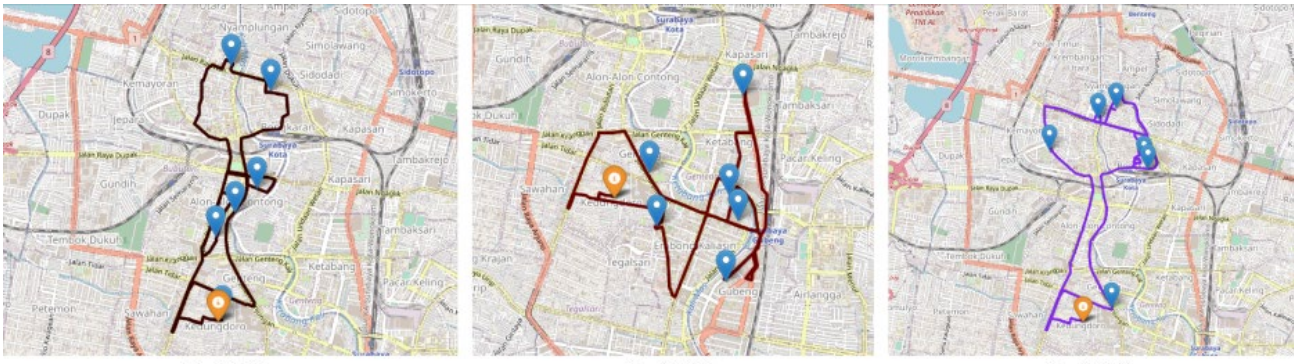
Di dalam grafik pada Gambar 15, memiliki pola yang sama pada setiap uji coba dan iterasi. Uji Coba dengan FHO memiliki waktu yang paling lama dari algoritma yang lainnya, terutama pada percobaan menggunakan iterasi dan populasi yang besar. Setelah FHO, CFA adalah algoritma dengan waktu jalan yang lebih lama daripada EPO, dan EPO adalah algoritma dengan waktu yang paling cepat daripada kedua algoritma lainnya. Selain pencatatan waktu, dilakukan pula pencatatan distribusi nilai fitness terhadap iterasi untuk setiap uji coba yang dilakukan. Grafik tersebut dapat dilihat pada Gambar 16 hingga Gambar 19.

Grafik distribusi menunjukkan bahwa setiap algoritma memiliki nilai fitness yang jarak maksimal dan minimal yang bervariasi. Persebaran fitness yang diperoleh menggunakan CFA cenderung memiliki range yang kecil dibandingkan dengan hasil fitness dengan menggunakan algoritma lain. Algoritma EPO dapat menghasilkan fitness terbaik dengan memberikan solusi yang memiliki fitness yang beragam. Sedangkan algoritma FHO memiliki distribusi fitness yang tidak tersebar merata melainkan terpusat pada fitness tertentu terutama pada iterasi 1000 dan 1500 dikarenakan percobaan hanya dilakukan untuk mencari fitness terbaik saja sehingga hasil yang kurang baik tidak tersimpan.

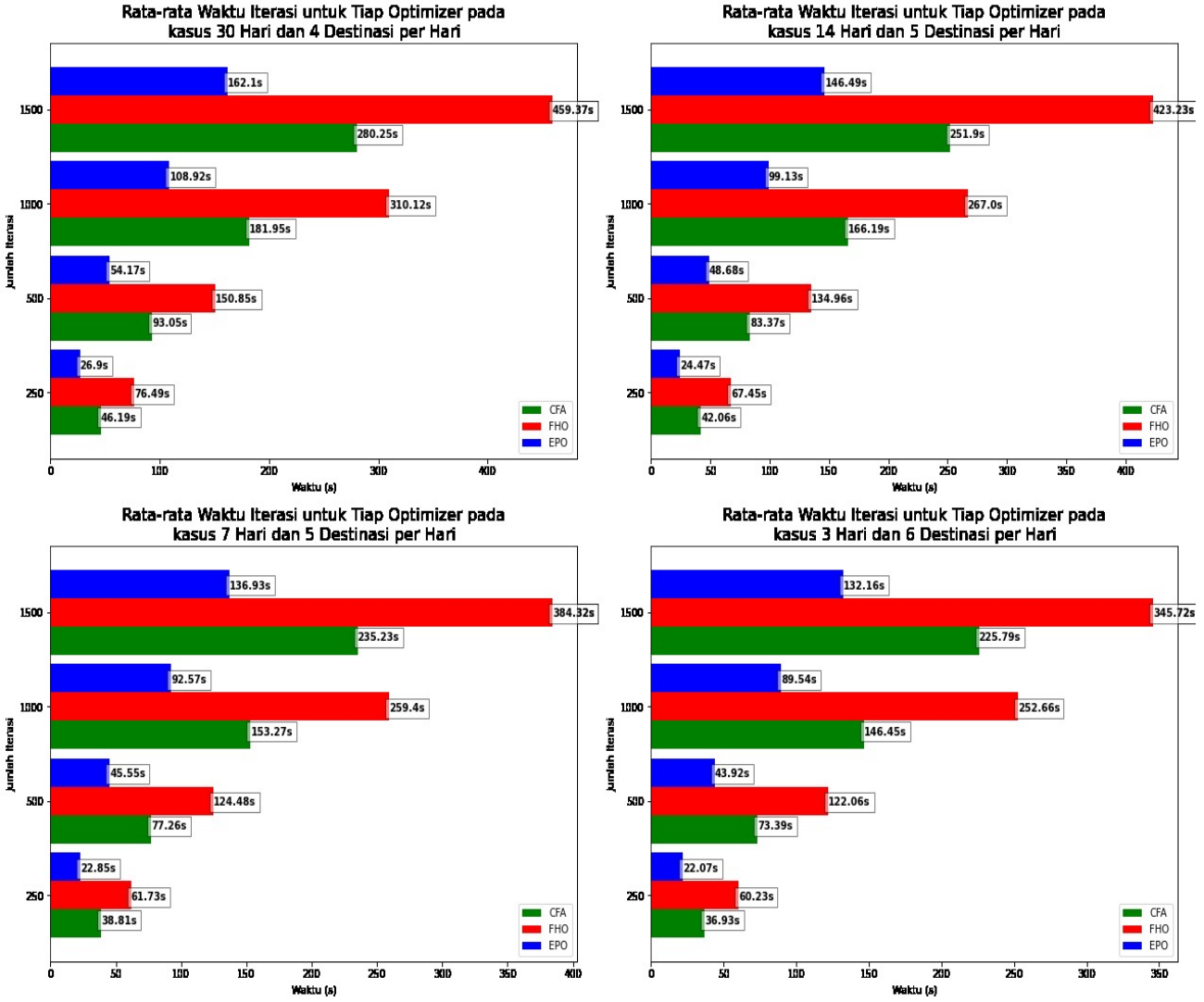
IX. KESIMPULAN

Berdasarkan hasil penelitian dan uji coba yang dilakukan, dapat terlihat EPO merupakan algoritma yang menghasil nilai fitness terbaik. Baik ditelaah dari sisi jumlah iterasi serta jumlah populasi ataupun melihat dari kasus yang dilakukan. Dimana pada setiap kasus tetap EPO menghasilkan fitness terbaik dengan nilai masing-masing, pada uji kasus pertama sebesar -970963.64, pada uji kasus kedua sebesar -426475.63, pada uji kasus ketiga sebesar -125116.57, dan uji kasus keempat sebesar -39590.81. Sedangkan dari sisi waktu komputasi, EPO tetap memegang waktu terbaik, sedangkan CFA memiliki waktu komputasi rata-rata terbaik kedua (69.59% lebih lambat dari EPO) serta FHO yang terlama dengan perbandingan 178.34% lebih lambat dari EPO. Maka dari semua hasil yang kami peroleh, baik dari sisi efisiensi komputasi dan hasil fitness score, maka algoritma EPO menjadi yang terbaik pada studi kasus ini.

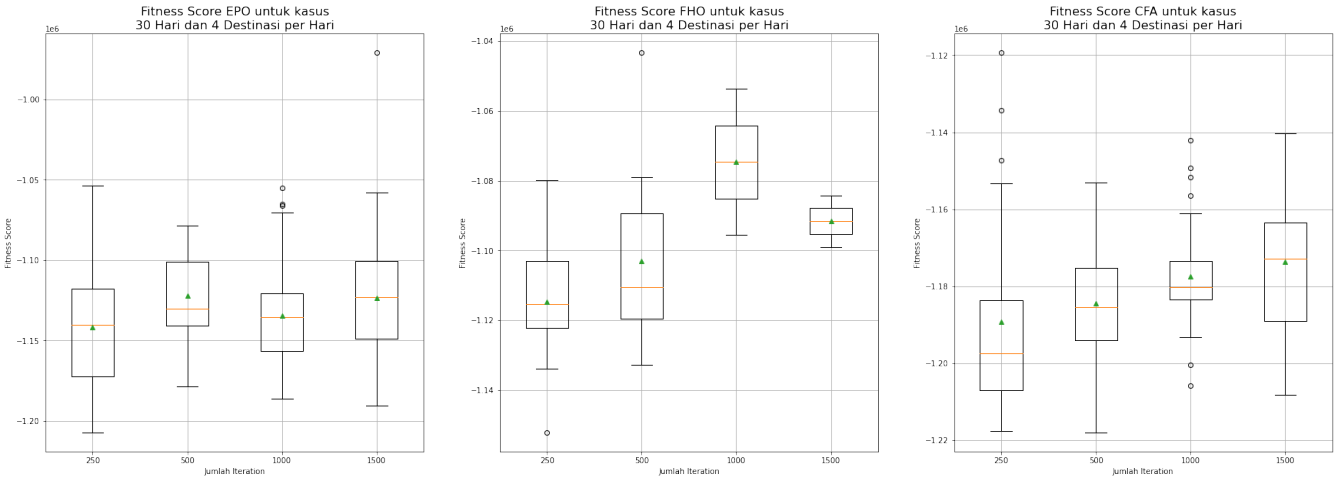
Terdapat beberapa hal yang masih dapat ditingkatkan untuk penelitian berikutnya. Jenis uji coba yang kami lakukan masih belum memiliki variasi yang memadai. Dimana bila dicoba dengan jumlah iterasi, jumlah penguin, serta parameter algoritma yang bervariasi memiliki kemungkinan untuk mendapatkan hasil fitness score yang lebih baik. Titik hotel yang digunakan pada penelitian ini hanya dari satu lokasi saja, akan lebih baik untuk menggunakan lebih banyak titik hotel untuk mendapatkan hasil yang lebih valid. Selain itu, data yang digunakan dapat ditingkatkan termasuk melakukan sortir spot pariwisata yang dimasa depan dapat berubah pula.



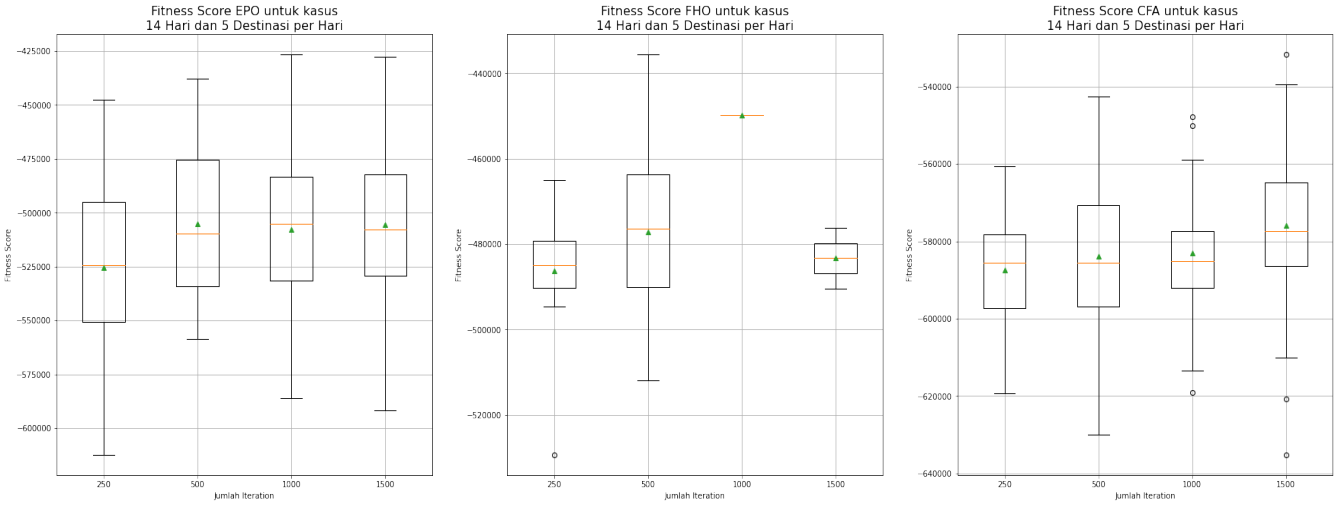
GAMBAR 14. Visualisasi Rute Hasil Uji Kasus #4



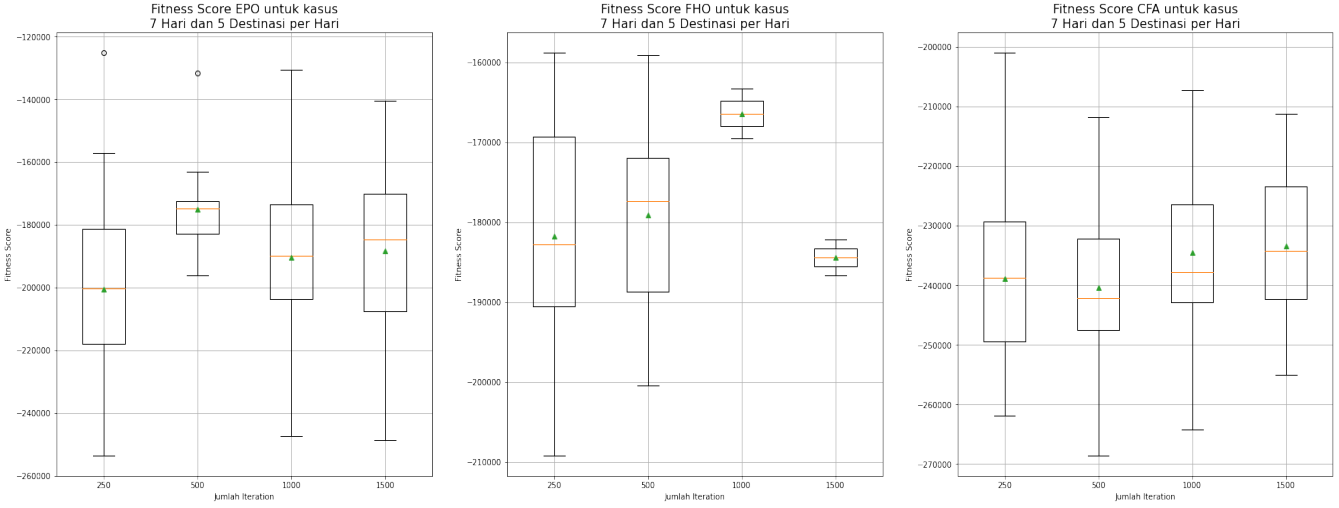
GAMBAR 15. Rata–rata Waktu untuk Setiap Algoritma



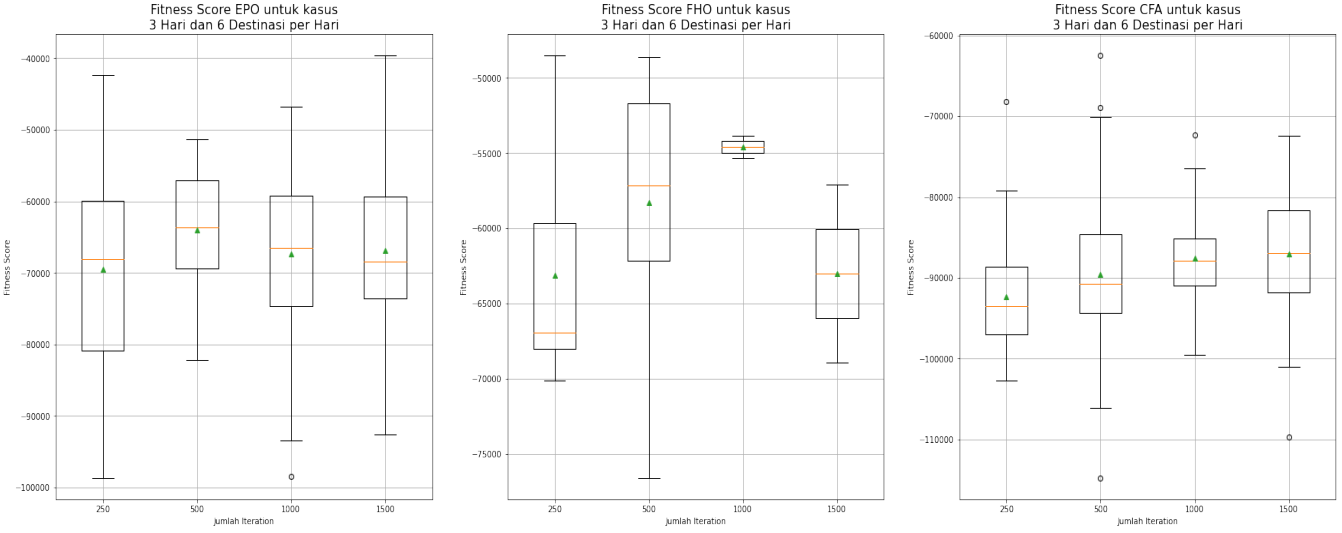
GAMBAR 16. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #1



GAMBAR 17. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #2



GAMBAR 18. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #3



GAMBAR 19. Perbandingan Distribusi Fitness Antar Algoritma Pada Uji Kasus #4

PERAN PENULIS

Christian Trisno Sen Long Chen: Konseptual, Persiapan Data, Implementasi Sistem, Visualisasi dan Uji Coba;
David Cahyadi: Pencarian dan Kurasi Data, Implementasi Sistem, Visualisasi;
Jonathan Arelio Bevan: Kurasi Data, Penyusunan Draf Asli, Penulisan Review dan Penyuntingan;
Williandy Takhta: Konseptual, Kurasi Data, Implementasi Sistem, Visualisasi dan Uji Coba;
Ariel Pratama Lesmana: Penyusunan Draf Asli, Penulisan Review dan Penyuntingan;
Christopher Davin Agus Poernomo: Implementasi Sistem, Visualisasi dan Uji Coba, Penyusunan Draf Asli;
Widean Nagari: Implementasi Sistem, Uji Coba, Penyusunan Draf Asli;

COPYRIGHT

DAFTAR PUSTAKA

[1] A. E. Eiben and G. Rudolph, "Theory of evolutionary algorithms: a bird's eye view," 1999. [Online]. Available: www.elsevier.com/locate/tcs
[2] A. N. Sloss and S. Gustafson, "2019 Evolutionary Algorithms Review," Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.08870
[3] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary Algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3. Wiley-Blackwell, pp. 178–195, 2014. doi: 10.1002/widm.1124.
[4] K. L. Hoffman and M. Padberg, "Traveling salesman problem," in *Encyclopedia of Operations Research and Management Science*, New York, NY: Springer US, 2001, pp. 849–853. doi: 10.1007/1-4020-0611-X_1068.
[5] Z. Hashim and W. R. Ismail, "Applications of Travelling Salesman Problem in Optimizing Tourist Destinations Visit in Langkawi," in *Regional Conference on Science, Technology and Social Sciences (RCSTSS 2014)*, Springer Singapore, 2016, pp. 265–273. doi: 10.1007/978-981-10-0534-3_25.
[6] O. Nurdiawan, F. A. Pratama, D. A. Kurnia, Kaslani, and N. Rahaningsih, "Optimization of Traveling Salesman Problem on Scheduling Tour Packages using Genetic Algorithms," in *Journal of Physics: Conference Series*, 2020, vol. 1477, no. 5. doi: 10.1088/1742-6596/1477/5/052037.
[7] F. Hanif Khan, N. Khan, and S. Inayatullah, "SOLVING TSP PROBLEM BY USING GENETIC ALGORITHM," *Article in International Journal of Basic & Applied Sciences*, vol. 28, p. 170, 2010, [Online]. Available: https://www.researchgate.net/publication/236009452
[8] J. Juwairiah, D. Pratama, H. C. Rustamaji, H. Sofyan, and D. B. Prasetyo, "Genetic Algorithm for Optimizing Traveling Salesman Problems with Time Windows (TSP-TW)," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 1, no. 1, pp. 1–8, Nov. 2019, doi: 10.25139/ijair.v1i1.2024.
[9] M. Tryana Sembiring and S. Chailes, "Ant Colony Optimization Implementation on Traveling Salesman Problem to Achieve the Shortest Logistic Route," in *IOP Conference Series: Materials Science and Engineering*, Dec. 2020, vol. 1003, no. 1. doi: 10.1088/1757-899X/1003/1/012045.
[10] A. Sabry Eesa, A. Mohsin Abdulazeez, Z. Orman, A. Mohsin, and A. Brifcani, "A Novel Bio-Inspired Optimization Algorithm Analysis of Health Data with Heuristic Learning Methods View project A New Dimensional Reduction Approach Based on Cuttlefish Algorithm and K-Nearest Neighbor for Gene Expression Data View project Cuttlefish Algorithm-A Novel Bio-Inspired Optimization Algorithm," *Article in International Journal of Scientific and Engineering Research*, vol. 4, no. 9, 2013, [Online]. Available: http://www.ijser.org
[11] G. Dhiman and V. Kumar, "Emperor penguin optimizer: A bio-inspired algorithm for engineering problems," *Knowl Based Syst*, vol. 159, pp. 20–50, Nov. 2018, doi: 10.1016/j.knosys.2018.06.001.
[12] M. Azizi, S. Talatahari, and A. H. Gandomi, "Fire Hawk Optimizer: a novel metaheuristic algorithm," *Artif Intell Rev*, 2022, doi: 10.1007/s10462-022-10173-w.
[13] Dr. James B. Wood, "http://www.thecephalopodpage.org/."
[14] H. Armanto, R. Kevin, and C. Pickerling, "Perencanaan Perjalanan Wisata Multi Kota dan Negara dengan Algoritma Cuttlefish," *INSYST*, vol. 1, no. 2, pp. 99–109, Dec. 2019.
[15] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst Appl*, vol. 42, no. 5, pp. 2670–2679, Apr. 2015, doi: 10.1016/j.eswa.2014.11.009.
[16] R. Li, J. Yang, X. Tuo, and R. Shi, "Research on Fitness Function of Two Evolution Algorithms Using for Neutron Spectrum Unfolding," 2021.