Implementasi Algoritma Evolusi FHO, MVPA, dan HHO pada TSP di Tempat Pariwisata Pulau Bali

Christian B. Sabdana[1], Bryan Christopher[1], Jason G. Sutanto[1], Lawrence P. Sianto[1], Lukky Hariyanto[1], dan Nickolas Hartono[1]
[1]Departemen Informatika, Fakultas Sains dan Teknologi, Institut Sains dan Teknologi Terpadu Surabaya, Surabaya, Indonesia

Corresponding author: Christian B. Sabdana (e-mail: christian.b20@mhs.istts.ac.id).

ABSTRACT Vacation activities are essential for both individuals and families. The creation of travel route is a rather difficult task and must be thought of as a whole. In computer science terms, finding the optimal route in a *graph* is known as Traveling Salesman Problem. In case of creating good and well-defined route, we need specialized algorithm that can perform better in evaluating the route and provide an overall good result. In this research, we examine 4 algorithms, namely HHO (Harris Hawk Optimization), FHO (Fire Hawk Optimization), MVPA (Most Valuable Player Algorithm) and Modified MVPA along with trying the best hyperparameters for TSP problems in 55 tourist location across Bali Island. After conducting several experiments, we found that within the same hyperparameter, HHO perform well enough and have consistently much better fitness but with much longer runtime. Meanwhile FHO with the worst fitness has a runtime that is much faster than HHO and MVPA. Our modified MVPA algorithm was able to provide better results, although not as good as HHO. Qualitatively, the HHO algorithm provided better travel results with relatively consistent distances each day. This helps tourists make the most of their time in enjoying tourist locations compared to travel time.

KEYWORDS Evolutionary Computation, Optimization Algorithm, Travelling Salesman Problem

ABSTRAK Kegiatan berlibur merupakan kegiatan yang diperlukan baik perseorangan maupun bersama keluarga. Pembuatan rute perjalanan yang optimal dari banyak wisata liburan terkadang menjadi permasalahan rumit dan perlu dipikirkan rute optimalnya secara keseluruhan. Dalam ilmu komputer, permasalahan mencari rute optimal pada sebuah jaringan ini dikenal dengan *Traveling Salesman Problem*. Untuk mendapatkan rute yang baik, diperlukan algoritma khusus yang mampu mengevaluasi rute perjalanan dan memberikan hasil perjalanan yang cukup optimal. Di dalam penelitian ini, 4 algoritma *Evolutionary Computation* yaitu HHO (Harris Hawk Optimization), FHO (Fire Hawk Optimization), MVPA (Most Valuable Player Algorithm) dan modifikasi dari algoritma MVPA dibandingkan untuk menyelesaikan permasalahan TSP pada 55 lokasi wisata di Pulau Bali. Setelah dilakukan beberapa percobaan, HHO merupakan algoritma dengan nilai *fitness* terbaik dan konsisten tetapi dengan waktu eksekusi yang lebih lama. Sementara algoritma FHO memiliki waktu eksekusi yang lebih cepat tetapi nilai *fitness* yang lebih buruk dibandingkan dengan HHO dan MVPA. Algoritma MVPA yang telah dimodifikasi dapat memberikan hasil yang lebih baik meskipun masih belum bisa sebaik HHO. Secara kualitatif, algoritma HHO memberikan hasil perjalanan yang lebih baik dengan jarak tempuh tidak terlalu bervariasi setiap harinya. Hal ini membantu pelaku wisata agar dapat memanfaatkan waktu lebih banyak dalam menikmati lokasi wisata dibandingkan waktu perjalanan yang terbuang.

KATA KUNCI Algoritma Optimisasi, Evolutionary Computation, Travelling Salesman Problem

I. PENDAHULUAN

Traveling Salesman Problem (TSP) merupakan permasalahan yang sering dihadapi secara umum dan sering digunakan sebagai bahan uji coba terhadap algoritma optimisasi. Permasalahan ini berada pada domain teori graf dan termasuk ke dalam salah satu dari beberapa problem *NP-hard*[1]. Karena hal tersebut, tidak ada algoritma dengan kompleksitas waktu polinomial yang dapat menemukan solusi eksak permasalahan ini dengan efektif, dan kebenaran solusinya tidak dapat dibuktikan dengan mudah. Di sisi lain, algoritma optimisasi dikembangkan dengan tujuan mendapatkan solusi aproksimasi dalam waktu yang jauh lebih singkat dibandingkan teknik *brute-force*[2].

Algoritma optimisasi merupakan algoritma yang berfokus pada pencarian solusi yang mengoptimalkan sebuah fungsi objektif[3]. Proses optimisasi akan dilakukan secara berulang-ulang (iteratif) dan di setiap iterasinya, akan dilakukan evaluasi terhadap solusi-solusi yang telah ditemukan. Iterasi ini dilakukan terus menerus hingga didapatkan solusi yang cukup optimal. Pada algoritma optimisasi, terdapat *hyperparameter* yang nantinya akan mengatur bagaimana algoritma tersebut meminimalkan fungsi objektif yang diberikan. *Hyperparameter* yang berbeda membuka kemungkinan untuk mendapatkan solusi yang berbeda dan memberikan nilai objektif yang lebih baik.

Algoritma optimisasi terbagi menjadi banyak bagian seperti *stochastic optimization*, *constraint satisfaction*, *meta-heuristic*, dll. *Evolutionary computing* termasuk kedalam kelompok algoritma *meta-heuristic* dengan mekanisme yang terinspirasi oleh teori evolusi Darwin dan perilaku makhluk hidup di alam[4]. Algoritma *evolutionary computing* cukup efektif dalam melakukan pencarian solusi dikarenakan biaya komputasinya yang berkembang secara linear dengan ukuran permasalahan, dan metode pencarian menggunakan populasi yang mempercepat pencarian dan membuka peluang untuk diparalelisasi[4]. Pada *evolutionary computing*, fungsi objektif disebut juga dengan *fitness function* dan pada setiap proses evaluasi, setiap kandidat solusi akan dibandingkan dengan mengoptimalkan nilai *fitness*.

Pada penelitian ini, digunakan beberapa algoritma *Evolutionary Computing* dalam menyelesaikan *Traveling Salesman Problem* yang telah divariasikan mendekati dengan permasalahan yang dihadapi ketika perjalanan wisata sebenarnya. Algoritma yang digunakan yaitu *Harris Hawk Optimization* (HHO) [2], *Fire Hawk Optimization* (FHO) [5], dan *Most Valuable Player Algorithm* (MVPA) [6], serta algoritma modifikasi dari MVPA. Pengujian dilakukan dengan melakukan perbandingan kuantitatif deskriptif dari skalabilitas algoritma ditinjau dari perkembangan jumlah populasi, rata-rata perkembangan nilai *fitness* di setiap *epoch*, serta perbandingan kecepatan eksekusi algoritma di setiap *epoch*.

Representasi solusi yang digunakan merupakan sebuah barisan dari indeks lokasi yang dikunjungi. Representasi ini digunakan dikarenakan permasalahan *Travelling Salesman Problem* merupakan *sequencing-type problem*[7]. Selain itu, dikarenakan operasi fungsi pada algoritma yang diuji berada pada domain dan *range* bilangan real, digunakan *random-keys representation*[8] untuk mendapatkan permutasi lokasi yang dikunjungi.

II. TINJAUAN PUSTAKA

A. TRAVELING SALESMAN PROBLEM

Permasalahan ini dapat dideskripsikan secara sederhana sebagai berikut[9]: Diberikan n *vertex* dan $n(n-1)/2$ nilai yang menyatakan jarak antar pasangan *vertex*, tentukan rute yang mengunjungi semua *vertex* tersebut dengan jarak tempuh minimal. Permasalahan ini memiliki hubungan erat dengan *hamiltonian cycle*[10], yaitu sebuah *graph* memiliki rute TSP jika dan hanya jika *graph* tersebut memiliki minimal 1 *hamiltonian cycle*. *Hamiltonian cycle* didefinisikan sebagai sebuah *cycle* yang mengunjungi setiap *vertex*, sehingga dapat disimpulkan bahwa TSP merupakan permasalahan mencari *hamiltonian cycle* dengan total *weight* minimum.

TSP memiliki banyak variasi permasalahan, seperti: *Asymmetric* dan *Symmetric* TSP[11] yang masing-masing merupakan rute minimum pada graf berarah dan graf tak-berarah, *Euclidean* TSP[9] ketika *vertex* berupa titik pada bidang planar dan nilai *edge* antar dua *vertex* berupa jarak *euclidean* antara kedua *vertex* tersebut, dan beberapa variasi lainnya.

Pada penelitian ini, variasi permasalahan yang diteliti adalah *Asymmetric* TSP dengan nilai *edge* antar *vertex*/lokasi merupakan jarak lintasan/jalur antar lokasi. Terdapat tambahan kompleksitas problem yaitu total hari berkunjung dan maksimal jarak yang dapat ditempuh per harinya. Teknis pengambilan jarak antar lokasi dijelaskan pada bagian selanjutnya.

B. HARRIS HAWK OPTIMIZATION

Harris Hawk Optimization (HHO) adalah sebuah teknik optimasi *population-based* dan *gradient-free* yang terinspirasi dari perilaku berburu Elang Harris. Dalam melakukan penyerangan, sekelompok elang Harris akan bersama-sama menyergap mangsa dari berbagai arah. Dikarenakan kemampuan mangsa untuk bergerak dan berpindah tempat, maka dilakukan beberapa variasi saat melakukan penyerangan[2].

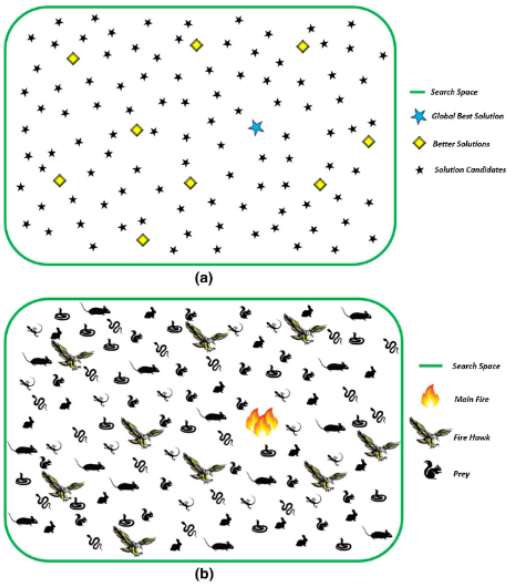
Dalam HHO, elang Harris direpresentasikan sebagai kandidat solusi dan kandidat solusi yang memiliki nilai *fitness* terbaik akan dianggap sebagai mangsa (*rabbit*). Proses pada HHO dibagi menjadi 2 fase, ekplorasi dan eksploitasi. Fase eksplorasi bertujuan untuk mengobservasi dan mengisolasi manga, bergantung terhadap energi mangsa (E). Pada beberapa kasus tertentu, energi yang dimiliki oleh mangsa dapat digunakan untuk berpindah tempat sehingga memaksa elang Harris untuk tetap melakukan eksplorasi dan mencari kandidat mangsa baru, dengan nilai *fitness* yang lebih baik. Ketika energi tidak cukup besar, maka elang Harris akan mengisolasi mangsa dan berpindah menuju tahap eksploitasi. Pada tahapan ini, elang Harris akan berfokus untuk menyerang mangsa secara lebih agresif.

Penyerangan pada fase eksploitasi dibagi menjadi *soft besiege* dan *hard besiege*. Kedua penyerangan ini memiliki variasi dengan ditambahnya perilaku *rapid dives*. Ketika melakukan eksploitasi, terdapat kemungkinan mangsa berhasil kabur. Saat mangsa mampu untuk kabur, maka elang Harris akan memadukan penyerangan dengan *rapid dives*.

Perhitungan energi mangsa pada setiap iterasi didesain sesuai dengan indeks iterasi sedemikian rupa sehingga pada awal iterasi, elang Harris akan lebih banyak melakukan eksplorasi dibandingkan eksploitasi kandidat solusi. Hal ini memberikan kesempatan bagi algoritma untuk menemukan kandidat solusi yang lebih baik sebelum berpindah menuju tahap eksploitasi.

C. FIRE HAWK OPTIMIZER

Fire Hawk Optimizer (FHO) merupakan *novel metaheuristic algorithm* yang terinspirasi dari perilaku mencari makanan dari *whistling kites, black kites,* dan *brown falcons*[5]. Ketiga elang tersebut menggunakan ranting kayu yang terbakar untuk membuat kebakaran kecil yang mengakibatkan mangsa terpaksa keluar dan kabur dari daerah tersebut sehingga mempermudah elang untuk menangkap mangsa. Perilaku tersebut yang membuat ketiga elang tersebut dinamakan *fire hawks*.



GAMBAR 1. Ilustrasi Fire Hawk Optimizer[5]**. Gambar (a) merupakan tahapan inisialisasi. Gambar (b) merupakan pembagian *Fire hawk* dengan mangsa di teritorinya.**

Dalam FHO, sejumlah kandidat solusi akan diasumsikan sebagai posisi vektor dari elang dan mangsa. Posisi awal dari kandidat solusi akan diinisialisasikan dengan proses acak yang dibatasi oleh daerah pencarian yang merepresentasikan batasan maksimum dan minimum, seperti yang ditampilkan pada Gambar 1a. Dari beberapa vektor yang diinisialisasikan, akan ada beberapa solusi yang memiliki nilai *fitness* lebih baik. Solusi-solusi inilah yang dijadikan sebagai *Fire hawks* sedangkan yang lainnya akan menjadi *preys* (mangsa), pada Gambar 1b. Selanjutnya, dilakukan perhitungan jarak *euclidean* untuk setiap pasangan elang dengan mangsa. Hal ini digunakan untuk menentukan teritori setiap elang dengan mencari nilai jarak rata-rata dari mangsa di sekitarnya.

Setiap elang selanjutnya akan bergerak mengambil dan memindahkan ranting terbakar ke teritorinya. Selain itu mangsa juga akan ikut bergerak di dalam daerah teritori setelah elang-elang tersebut menaruh apinya. Terdapat *safe place* di dalam teritori *Fire hawk* yang merupakan tempat aman bagi mangsa untuk berlindung dari *Fire hawk*. *Safe place* dapat dihitung dengan mengambil nilai rata-rata dari mangsa yang ada di daerah tertentu. Mangsa dapat memilih untuk berjalan sesuai perilaku binatang di dalam teritori *Fire hawk*.

Pada akhir iterasi, akan dilakukan perhitungan nilai *fitness* untuk setiap elang dan mangsa, serta akan ditentukan kandidat solusi *global best* (GB) atau solusi terbaik dari semua populasi. Algoritma ini akan terus berjalan hingga mencapai angka iterasi maksimum, dan mengembalikan *global best* sebagai aproksimasi solusi optimal.

D. MOST VALUABLE PLAYER ALGORITHM

Most Valuable Player Algorithm adalah sebuah *metaheuristic algorithm* yang terinspirasi dari sistem kompetisi olahraga. Pada sebuah kompetisi olahraga, sebuah populasi (*population*) dari pemain (P) akan bersaing secara tim (T) untuk memenangkan kompetisi serta mereka juga akan bersaing secara individual untuk memperebutkan gelar MVP (*Most Valuable Player*)[6].

Pada algoritma ini terdapat banyak *players*, banyak tim, dan banyak *fixture* atau pertandingan. Pada setiap *fixture*, akan terdiri dari 4 fase utama, yakni *Competition*, *Application of Greediness*, *Application of Elitism*, dan *Remove Duplicate*.

1) FASE *COMPETITION*

Fase ini akan dilakukan untuk setiap *team* dan terdiri dari 2 fase utama, yakni fase *Individual Competition* dan *Team Competition*. Pada fase *Individual Competition*, setiap *player* dari sebuah tim akan bersaing untuk menjadi *player* terbaik di timnya (*Franchise Player*). Sedangkan pada *Team Competition*, setiap *player* akan bersaing untuk menjadi yang terbaik pada sebuah *competition*.

Setelah melakukan *Individual Competition*, setiap tim (T_i) akan melakukan *Team Competition*. Pada fase ini akan dipilih satu tim lain (T_j) untuk dilawankan dengan tim T_i. Mekanisme pemilihan tim yang menang menggunakan probabilitas proporsional berdasarkan nilai *fitness* kedua tim.

2) FASE *APPLICATION OF GREEDINESS*

Fase ini dilakukan setelah semua tim melewati fase *Individual Competition*. Pada fase ini, semua player akan dibandingkan dengan dirinya sendiri sebelum melewati fase *Team Competition*. Jika player baru lebih baik, maka akan digunakan skill baru dari player tersebut, namun jika player lama lebih baik, maka tetap digunakan skill lama dari player tersebut.

3) FASE *APPLICATION OF ELITISM*

Elitism yang dimaksud pada fase ini adalah mengganti beberapa *player* yang terburuk dan digantikan dengan *player* yang terbaik. Pada MVPA, satu per tiga *player* terbaik akan menggantikan satu per tiga *player* terburuk.

4) FASE *REMOVE DUPLICATE*

Pada fase ini, *player* yang kembar dan bersebelahan, salah satunya akan digantikan dengan *player* terbaik pada urutan setelah player yang menggantikan pada fase *elitism*. Prosedur ini sama dengan yang dijelaskan pada penelitian yang dilakukan oleh Elsayed[12].

Setelah semua *fixture* telah dilakukan atau *stopping condition* telah tercapai, maka solusi terbaik bisa didapatkan dengan mengambil *player* yang memiliki nilai *fitness* terbaik.

III. METODE YANG DIAJUKAN

Metode penyelesaian permasalahan menggunakan algoritma optimisasi HHO, FHO, MVPA, dan algoritma MVPA yang telah dimodifikasi. Permasalahan yang dipilih adalah permasalahan TSP yang telah disesuaikan untuk mencari rute optimal dalam mengunjungi tempat wisata di Pulau Bali.

A. DATA

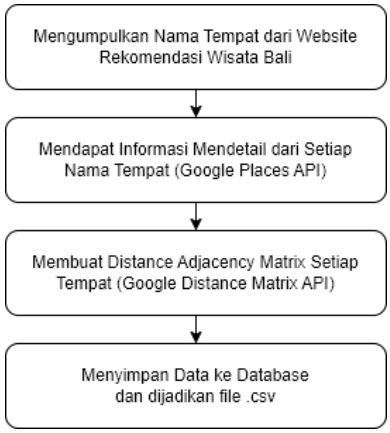
Data yang digunakan dalam penelitian ini adalah data lokasi tempat pariwisata yang berada pada Pulau Bali saja. Pengumpulan data serta detailnya akan menggunakan bantuan Google Places API[13] dan Google Distance Matrix API[14].

Untuk daftar lokasi yang dipilih, dilakukan pengumpulan data dari media online sebanyak 55 lokasi pariwisata populer di Pulau Bali. Lokasi-lokasi tersebut kemudian dimasukkan ke dalam Google Places API untuk mendapatkan informasi seperti nama lengkap, nama jalan, *latitude* dan *longitude*, dan *url* google map. Google Places API akan mengembalikan beberapa tempat yang kemungkinan mirip dengan *query* nama yang telah diinputkan. Dari beberapa hasil, akan diambil hasil yang paling relevan. Setelah itu data lokasi disimpan ke dalam basis data MySQL. Visualisasi proses ini ditampilkan pada Gambar 2.

Dikarenakan pada permasalahan ini setiap *vertex* terhubung satu sama lain, maka diperlukan *distance matrix* dari setiap titik ke titik lainnya dengan ukuran 55×55. Untuk mendapatkan jarak dari satu titik ke titik lainnya, digunakan Google Distance Matrix API. Jarak yang didapatkan dari Google Place API bukanlah hasil perhitungan *euclidean distance*, melainkan dari hasil perhitungan jalan yang sesungguhnya dapat ditempuh oleh kendaraan bermotor yang tentunya bukan merupakan garis lurus. Jarak antar pasangan lokasi-pun juga disimpan ke dalam basis data pada tabel yang berbeda.

B. PERMASALAHAN DAN REPRESENTASI

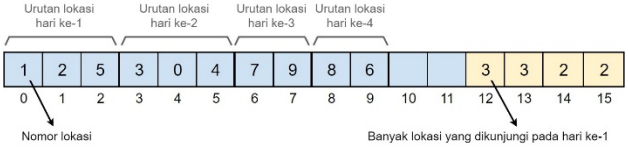
Permasalahan akan dipusatkan pada pencarian rute optimal pada 55 tempat wisata di Pulau Bali. Variabel yang menjadi *input* parameter antara lain jumlah hari berlibur (λ) di Bali dan jumlah lokasi maksimal yang dapat dikunjungi tiap harinya (ϕ). Jumlah lokasi maksimal (M) yang dapat dikunjungi bisa didapat dari hasil perkalian λ dan ϕ. Pada permasalahan ini, terdapat sedikit perbedaan dengan TSP, yakni tidak perlu kembali ke lokasi pertama setelah selesai mengunjungi lokasi lain.



GAMBAR 2. Alur Pencarian Data Lokasi Wisata di pulau Bali

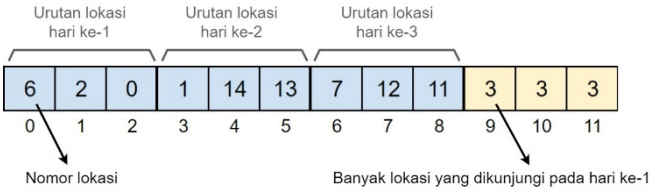
Akan terdapat 2 skenario yang dapat terjadi pada permasalahan ini. Yang pertama, bila jumlah lokasi (n) kurang dari atau sama dengan M, maka setiap lokasi pasti dikunjungi dan akan terdapat hari di mana lokasi yang dikunjungi tidak maksimal atau bahkan tidak mengunjungi lokasi sama sekali. Yang kedua, bila n lebih dari M, maka setiap hari jumlah lokasi yang dikunjungi pasti sejumlah ϕ dan ada kemungkinan terdapat satu atau lebih lokasi wisata yang tidak dikunjungi sama sekali.

Permasalahan ini akan diselesaikan dengan representasi perjalanan kota selama λ. Representasi kandidat solusi berupa array 1 dimensi sepanjang $M+\lambda$. Contohnya, apabila n adalah 55, λ adalah 5 hari dan ϕ adalah 11, maka representasinya adalah sebuah array 1 dimensi sepanjang 60 elemen. M elemen pertama (berjumlah 55) akan berisi nomor lokasi dan melambangkan urutan kunjungan lokasi. Sedangkan, λ elemen terakhir (berjumlah 5) melambangkan banyak lokasi yang akan dikunjungi untuk tiap harinya. Informasi ini digunakan untuk memastikan jumlah lokasi yang dikunjungi perhari tidak melebihi parameter ϕ. Pada skenario pertama, akan terdapat kolom array yang tidak digunakan pada akhir dari M-elemen pertama.



GAMBAR 3. Representasi pada Skenario Pertama

Pada contoh Gambar 3, parameter yang diberikan adalah 4 untuk λ, 3 untuk ϕ dan 10 tempat wisata (n). Melalui perhitungan yang telah dijelaskan sebelumnya, nilai M adalah 12. Karena n kurang dari M, contoh ini termasuk kedalam skenario pertama. M-elemen pertama (indeks ke-0 hingga ke-11) mewakilkan urutan tempat yang dikunjungi, sedangkan λ-elemen sisanya (indeks ke-12 hingga 15) mewakilkan jumlah tempat yang dikunjungi per harinya. Sebagai contoh, elemen indeks ke-12 memiliki nilai 3 yang artinya pada hari pertama, banyak lokasi yang dikunjungi adalah 3 lokasi dengan urutan indeks lokasi 1, 2, dan 5.



GAMBAR 4. Representasi pada Skenario Kedua

Pada contoh Gambar 4, parameter yang diberikan adalah 3 untuk λ, 3 untuk ϕ, dan dengan 15 lokasi wisata (n). Berdasarkan parameter yang diberikan, dapat ditentukan bahwa nilai M adalah 9. Karena nilai n lebih dari atau sama dengan M, contoh ini termasuk dalam skenario kedua. M-elemen pertama (indeks ke-0 hingga 8) mewakilkan urutan tempat yang dikunjungi, sedangkan λ-elemen sisanya (indeks ke-9 hingga 11) mewakilkan jumlah tempat yang dikunjungi per harinya. Sebagai contoh, elemen indeks ke-9 memiliki nilai 3 yang artinya pada hari pertama, banyak lokasi yang dikunjungi adalah 3 lokasi dengan urutan indeks lokasi 6, 2, dan 0.

C. FITNESS FUNCTION

Terdapat 3 fungsi objektif yang akan digunakan sebagai *fitness function*.

1) GLOBAL DISTANCE FUNCTION

Fungsi ini akan digunakan untuk meminimalkan total jarak yang ditempuh selama perjalanan mengunjungi lokasi wisata. Fungsi ini merupakan *inverse* dari total jarak pada kandidat solusi[15], yang selanjutnya akan di-*scaling* untuk mengatur jangkaukan keluaran fungsi. Secara matematis, fungsi ini didefinisikan pada formula (1).

$$F_1 = \frac{c_1}{\sum_{i=2}^{n} d_{s_{i-1}}^{s_i}} \quad \forall\, s_i \in S \tag{1}$$

n = banyaknya lokasi
S = himpunan terurut total dari kota-kota pada perjalanan
d_a^b = jarak antara kota dengan indeks a menuju b
c_1 = konstanta

Himpunan S pada formula (1) yang diberikan merupakan M-elemen pertama dari representasi solusi. Terdapat konstanta c_1 yang digunakan sebagai *scaling factor* bagi fungsi objektif agar tidak memberikan nilai *fitness* yang terlalu berjauhan jika data yang digunakan berbeda. Nilai konstanta c_1 dapat dihitung pada formula (2).

$$c_1 = n \sum_{i=1}^{n} d_{\max}(i) \tag{2}$$

n = banyaknya lokasi
$d_{\max}(i)$ = nilai maksimum ke-i dari matriks jarak d

Perhitungan nilai konstanta pada formula (2) tidak melibatkan nilai dari kandidat solusi, hal ini mengimplikasikan bahwa perhitungan konstanta dapat dilakukan pada tahapan inisialisasi. Dengan demikian, bobot komputasi pada setiap *epoch*-nya dapat dikurangi.

2) LOCATION LIMIT FUNCTION

Fungsi objektif ini digunakan untuk memberikan *punishment* ketika lokasi yang dikunjungi pada suatu hari melebihi batasan ϕ. *Punishment* (p) untuk hari ke-j didefinisikan sebagai formula (3).

$$p(j) = \begin{cases} n_j - \phi, & n_j \geq \phi \\ 0, & n_j < \phi \end{cases} \tag{3}$$

n_j = banyaknya lokasi yang dikunjungi pada hari ke-j

Sistem *punishment*[16] ini dapat dilihat sebagai sebuah *rectifier*. *Penalty* akan diberikan pada hari dengan jumlah lokasi yang dikunjungi melebihi batasan ϕ, tetapi tidak ada *reward* ketika kandidat solusi dapat memenuhi batasan. Nilai akhir yang dikembalikan oleh fungsi objektif ini berupa jumlahan dari seluruh *punishment* masing-masing hari, seperti yang ditunjukkan pada formula (4).

$$F_2 = -1 \sum_{i=1}^{\lambda} p(i) \tag{4}$$

3) DISTANCE VARIANCE PER DAY

Fingsi objektif ini digunakan untuk meminimalisir perbedaan total jarak perhari. Pada perhitungan fungsi objektif ini, digunakan nilai standard deviasi pada data jarak yang ditempuh setiap harinya. Standard deviasi digunakan karena kemampuannya dalam menilai persebaran data[17], sehingga dapat diketahui seberapa jauh perbedaan total jarak yang ditempuh perhari. Fungsi ini didefinisikan sebagai formula (5).

$$F_3 = \frac{c_2}{\sigma(D)+1} \tag{5}$$

$D = \{d_{s_{i-1}}^{s_i} | s_i \in S,\ i \geq n\}$
σ = standard deviasi
c_2 = konstanta
d_a^b dan S referensi menuju formula 1

Terdapat pula sebuah konstanta c_2 yang berperan sebagai *scaling factor* dan dapat dihitung dengan formula (6).

$$c_2 = \frac{d_{\max} - d_{\min}}{2} \tag{6}$$

Nilai d_{max}, dan d_{min} pada formula (6) masing-masing merupakan nilai maksimum dan nilai minimum pada matriks jarak *d*.

Setelah ketiga fungsi objektif pada formula (1), (4), dan (5) digunakan untuk memberikan evaluasi, dilakukan agregasi pada ketiga nilai tersebut untuk memberikan nilai *fitness* tunggal bagi kandidat solusi. Agregasi yang dilakukan menggunakan *weighted sum*[7] dengan formula (7).

$$F = F_1 + F_2 + 0.2F_3 \quad (7)$$

Kegunaan *scaling factor* pada formula (1) dan (5) untuk menyeimbangkan peran masing-masing fungsi pada perhitungan *fitness* di formula (7). Fungsi F_1 pada formula (1) memiliki nilai minimum mendekati *n* yang terjadi apabila total jarak pada kandidat solusi sama dengan total *n* jarak maksimum pada *distance matrix*. Sedangkan, fungsi F_2 pada formula (4) dan F_3 pada formula (5) masing-masing memiliki nilai maksimum 0 dan c_2 (lihat formula (6)). Fungsi F_2 yang bersifat sebagai *punishment*, memiliki bobot yang seimbang dengan fungsi F_1 sebagai objektif utama. Kondisi ini mengharuskan algoritma untuk mencari kandidat solusi yang dapat menyesuaikan dengan batasan, dikarenakan batasan ϕ dianggap sebagai *hard constraint* (termasuk kedalam salah satu input parameter). Sedangkan, fungsi F_3 pada formula (5) yang mengatur persebaran jarak, diberikan bobot yang lebih kecil dan konstanta *scaling factor* yang dinamis menyesuaikan dengan data, menjaga agar fungsi objektif ini tidak terlalu mendominasi fungsi lainnya.

D. MODIFIED MVPA

Selain melakukan uji coba menggunakan tiga algoritma metaheuristik yang telah disebutkan sebelumnya, dilakukan juga uji coba dengan menggunakan hasil modifikasi dari *Most Valuable Player Algorithm*. Modifikasi yang diberikan berada pada bagian *Individual Competition* dan pada fase *Application of Elitism* dan *Application of Greediness*. Perubahan pada Individual Competition ini bertujuan untuk memberikan porsi bagi setiap player melakukan improvisasi mengikuti *Franchise Player* dan *MVP*. Formula modifikasi pada tahapan *Individual Competition* didefinisikan sebagai formula (8).

$$T_i = T_i + (1-\alpha) \times r_1 \times (FP_i - T_i) + \alpha \times r_2 \times (MVP - T_i) \quad (8)$$
$$\alpha = hyperparameter$$

Modifikasi didasarkan pada mutasi *pattern-search-type* seperti formula perubahan kecepatan partikel pada PSO (Particle Swarm Optimization)[7][18]. *Franchise Player* dan *MVP* digunakan sebagai faktor pembanding (seleksi/improvisasi) dengan pemain terkait. Yang menjadi perbedaan dengan formula kecepatan pada PSO adalah, penggunaan α sebagai *control variable*, yang mengatur kecenderungan improvisasi *player*. Nilai α yang mendekati 0 akan membuat *player* lebih cenderung melakukan improvisasi mengikuti *Franchise Player*. Sebaliknya, nilai α yang mendekati 1 akan membuat *player* cenderung melakukan improvisasi mengikuti *MVP*. Bilangan acak r_1 dan r_2 dipertahankan sebagai unsur ketidakpastian.

Sedangkan untuk perubahan pada fase *Application of Elitism* dan *Application of Greediness* digunakan konsep yang mirip dengan penentuan pemenang pada *Team Competition*, yakni dengan melihat probabilitas berdasarkan *fitness* untuk menentukan apakah pada *player* tersebut perlu dilakukan seleksi menggantikan kandidat solusi saat ini. Dengan demikian, akan ada kemungkinan *player* tidak terseleksi pada fase *Greediness* atau *Elitism*. Probabilitas yang digunakan pada *Greediness* dan *Elitism* didapatkan dengan menggunakan formula (9).

$$Prob_{P_j}^{P_i} = \frac{F(P_i)}{F(P_i)+F(P_j)} \quad (9)$$

$Prob_{P_b}^{P_a}$ = probabilitas *player a* menggantikan *player b*
$F(P_a) = fitness\ score$ dari kandidat solusi *player a*

Pada fase *Greediness*, P_i merupakan kandidat solusi *player* sesudah fase *competition*, dan P_j merupakan kandidat solusi *player* yang sama setelah fase *competition*. Sedangkan pada fase *Elitism*, P_i mewakilkan player yang dipilih secara terurut menurun dari satu per tiga player terbaik, dan P_j mewakilkan player yang dipilih secara terurut menaik dari satu per tiga player terburuk.

Perhitungan probabilitas pada formula (9) didasari oleh motivasi untuk memberikan peluang proporsional bagi sebuah solusi yang dianggap lebih buruk untuk bisa *survive*. Pemberian peluang proporsional digunakan untuk menghindari *strong elitism* yang dapat membuat algoritma menuju fase konvergensi terlalu dini[7]. Hal ini membantu populasi untuk tidak terjebak pada *local maxima* dan bisa lebih tersebar untuk mencari solusi dengan nilai *fitness* yang lebih baik.

IV. UJI COBA

Uji coba yang dilakukan berupa pengujian beberapa parameter yang dapat mempengaruhi dan menghasilkan kandidat solusi terbaik. Kota yang akan dikunjungi pertama bersifat acak, dan berdasarkan representasi yang digunakan, algoritma optimisasi akan menentukan kota pertama yang dikunjungi agar dapat memberikan hasil *fitness* yang lebih baik. Uji coba ini akan dilakukan pada layanan *cloud service* dengan *environment* yang sama dan memanfaatkan library *meta-heuristic evolution algorithm MealPy*[19]. Dari ketiga algoritma yang diuji di paper ini, implementasi yang tersedia pada library terkait hanya algoritma FHO dan HHO. Algoritma MVPA dan modified-MVPA diimplementasikan menggunakan bantuan *optimizer class* sebagai *custom optimizer* pada library *MealPy*.

Uji coba yang dilakukan meliputi pengujian skenario seperti yang telah dijelaskan sebelumnya, pengujian *population size* yang berbeda, dan khusus untuk *Modified* MVPA juga dilakukan pengujian nilai *alpha* berbeda. Secara

keseluruhan 7 algoritma yang diuji coba yaitu FHO, HHO, *Original* MVPA, *Modified* MVPA (*alpha* 0.2), *Modified* MVPA (*alpha* 0.4), *Modified* MVPA (*alpha* 0.6), dan *Modified* MVPA (*alpha* 0.8).

Di setiap uji coba, setiap algoritma pada masing-masing konfigurasi ukuran populasi dan jumlah iterasi akan diuji sebanyak 4 kali dan dari ke-empat percobaan tersebut, akan diambil nilai *fitness* terbaik (*Best F*), rata-rata nilai *fitness* (*Mean*), standard deviasi (*Std*), dan rata-rata waktu eksekusi setiap *epoch* (*Avg Time*). Khusus pada algoritma MVPA (*original* maupun modifikasi), terdapat *hyperparameter* tambahan berupa jumlah tim pada populasi (N Tim) dan jumlah *player* yang ada pada satu tim. Jumlah *player* pada satu tim didapat dengan membagi ukuran populasi uji coba dengan jumlah tim.

Uji Coba 1

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **11 hari *Vacation Duration*** (λ) dan **6 *Limit Location per Day*** (ϕ). Uji coba 1 ini merepresentasikan skenario pertama, sehingga terdapat kemungkinan satu atau beberapa hari, jumlah lokasi yang dikunjungi tidak mencapai ϕ. Hasil kuantitatif dari uji coba ini ditunjukkan pada tabel I.

Uji Coba 2

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **11 hari *Vacation Duration*** (λ) dan **3 *Limit Location per Day*** (ϕ). Uji coba ini merepresentasikan skenario kedua, sehingga memungkinkan terdapat beberapa lokasi yang tidak dikunjungi sama sekali. Hasil kuantitatif dari uji coba ini ditunjukkan pada tabel II.

Uji Coba 3

Pada semua model algoritma, akan dibandingkan beberapa nilai populasi dan iterasi dengan parameter **30 hari *Vacation Duration*** (λ) dan **6 *Limit Location per Day*** (ϕ). Uji coba ini merepresentasikan skenario pertama dengan durasi berlibur yang panjang, sehingga memberikan opsi cukup banyak bagi algoritma. Hasil kuantitatif dari uji coba ini ditampilkan pada tabel III.

TABEL I
PERFORMA UJI COBA 1 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (11 λ DAN 6 ϕ)

N Tim	Pop	Iter	FHO				HHO				*ORIGINAL* MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	23254.8	20866.6	1380.3	0.00454	73529.5	62021.8	9480.81	0.20621	50751.9	39917.9	6800.0	0.25787
12	120	1000	21987.8	20339.8	1007.3	0.00675	72427.1	52996.2	**12951.9**	0.15898	68102.1	53064.2	9606.4	0.24606
12	120	5000	21399.9	20457.9	963.4	0.00571	111870.1	87039.4	21112.0	0.19563	49322.5	43947.8	**3132.0**	0.24655
40	600	500	23736.7	22929.9	650.6	0.02970	98037.8	82469.7	18176.5	1.11382	69879.3	46710.2	14427.8	1.22115
40	600	1000	21767.6	21262.1	**312.1**	0.03468	102490.2	77432.1	15045.6	1.31281	74497.1	51681.9	13868.5	1.24702
40	600	5000	23064.7	22175.2	795.8	0.04065	129357.0	95177.5	29538.9	1.31517	63531.5	50517.7	8139.9	1.34217
100	2000	500	24081.6	22453.9	1227.5	0.13779	**212255.2**	119114	57094.1	5.84633	55192.7	47333.2	5228.6	4.42658
100	2000	1000	**25632.2**	23810.4	1750.9	0.12889	128425.6	88635.6	26122.5	7.49057	**83055.1**	**65325.8**	15785.9	4.72104
100	2000	5000	24462.3	**22943.1**	1003.2	0.11438	190605.0	**134313**	35215.6	6.59766	76942.6	61504.6	14480.4	4.36209

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
72536.8	55240.2	10147.8	0.23207	53458.4	48162.7	**3405.3**	0.26262	44938.7	41547.3	**2670.6**	0.20980	41348.2	39250.5	**1626.1**	0.22076
65273.6	48067.3	10124.8	0.21265	74683.8	57595.2	14256.0	0.22506	64729.5	52499.7	8200.9	0.24742	44110.6	39939.5	2451.9	0.22659
64554.1	62096.9	**1942.5**	0.22554	65736.9	56417.2	8055.7	0.22127	62633.9	53504.5	6816.6	0.23949	49862.7	42260.7	5074.6	0.25021
90203.0	77489.3	9885.0	1.10030	90326.2	68316.3	14355.2	1.15466	66181.3	54488.3	7871.9	1.15329	54378.0	45575.9	5117.6	1.04789
88287.4	76866.6	7804.0	1.16666	75389.7	66269.2	7700.8	1.23143	69166.8	59864.9	6476.5	1.28540	72734.8	58917.6	8570.2	1.23640
122483.6	99455.7	18050.4	1.17583	102471.1	80370.2	13306.2	1.25028	81951.6	68653.7	13195.2	1.23039	**73929.0**	56216.7	10251.9	1.23013
140616.5	119350.3	17638.5	4.15141	99912.2	80408.9	13031.3	3.94342	84169.8	65806.7	11457.9	4.33358	69401.3	57225.8	7050.1	4.02652
142138.7	**122547.6**	22082.6	3.81548	91748.1	85118.1	4667.3	4.74443	**86830.6**	69695.5	13114.7	4.03688	62076.4	53802.5	4958.3	4.47950
157275.3	129580.4	18355.7	3.94974	**117414.2**	**91153.5**	24085.3	4.65814	86661.5	**73295.2**	9946.9	4.09069	76949.7	**61768.7**	9248.6	4.39873

TABEL II
PERFORMA UJI COBA 2 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (11 λ DAN 3 ϕ)

N Tim	Pop	Iter	FHO				HHO				*ORIGINAL* MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	20182.3	19675.9	480.5	0.00125	60730.1	49099.3	8052.5	0.04120	43229.3	37793.3	5145.0	0.05484
12	120	1000	23658.8	21276.8	1496.2	0.00119	84771.9	78159.5	8225.5	0.04166	49345.2	43570.3	3727.6	0.05381
12	120	5000	21609.6	20916.7	**423.2**	0.00121	88116.9	70816.2	11800.9	0.04234	46638.9	40929.8	4621.6	0.05373
40	600	500	24954.4	22279.0	1733.0	0.00685	90238.0	77971.8	**7917.8**	0.26466	54678.2	53962.4	**796.6**	0.25779
40	600	1000	21539.5	20960.7	784.7	0.00633	165196.8	109512	37951.1	0.27745	61485.7	54413.8	8421.1	0.27378
40	600	5000	22980.7	22187.4	542.7	0.00648	159967.9	81916.7	46863.5	0.25955	56805.2	44442.8	7703.4	0.26776
100	2000	500	24566.9	22953.8	1083.9	0.02139	**179087.2**	**131829.0**	48202.9	1.48002	**89137.1**	**60221.0**	16897.8	0.87154
100	2000	1000	**26898.0**	**24804.0**	1398.7	0.02543	154855.4	131773.7	20239.3	1.37753	68950.2	59262.8	10778.7	0.91145
100	2000	5000	23483.4	22918.9	539.0	0.02641	167967.0	109978.1	34725.8	1.31055	68613.9	58171.2	6377.1	0.94613

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
56180.8	44647.6	8134.3	0.04877	67611.7	53248.8	8817.98	0.04709	53802.3	47458.4	4236.7	0.05042	46190.5	39847.2	3773.7	0.05032
64682.5	54547.6	7717.0	0.04746	53975.8	48856.5	**5035.2**	0.05039	48223.5	45337.2	**2312.2**	0.04843	59186.2	47141.6	8131.9	0.04806
76093.8	65997.5	10017.8	0.04727	79443.9	62020.5	10498.5	0.04909	52300.3	49105.2	2752.6	0.04948	66807.6	53527.9	8448.9	0.05073
81788.8	77798.5	4895.5	0.24521	83682.3	68196.2	10336.1	0.25872	64319.1	55569.2	6260.5	0.24371	53575.2	50630.7	**2384.9**	0.23929
89923.3	86868.1	**3135.2**	0.25771	93283.2	75364.5	12865.1	0.23777	75488.6	61661.4	9585.7	0.24697	55818.6	47274.6	5217.9	0.25559
112262.0	102961.3	9337.1	0.23838	107949	93835.4	9084.7	0.23996	75661.6	61240.5	8359.5	0.24979	**76584.4**	**59641.1**	10386.9	0.24276
123783.1	108507.1	10926.6	0.80637	95494.3	86973.4	5929.7	0.83956	76761.6	72552.8	3357.1	0.82310	73362.7	52977.0	12684.7	0.80323
160794.6	**132293.8**	18371.0	0.81349	97866.3	89023.9	7827.7	0.85697	**105925.1**	**83982.8**	14458.3	0.81924	62470.7	55498.5	5064.8	0.88925
174693.8	127636.3	27575.1	0.83819	**132054.9**	**94923.1**	22858.6	0.81763	78444.1	70247.6	5560.2	0.81518	57825.3	53345.9	3471.7	0.93155

TABEL III
PERFORMA UJI COBA 2 FHO, HHO, *ORIGINAL* MVPA, DAN *MODIFIED* MVPA (30 λ DAN 6 ϕ)

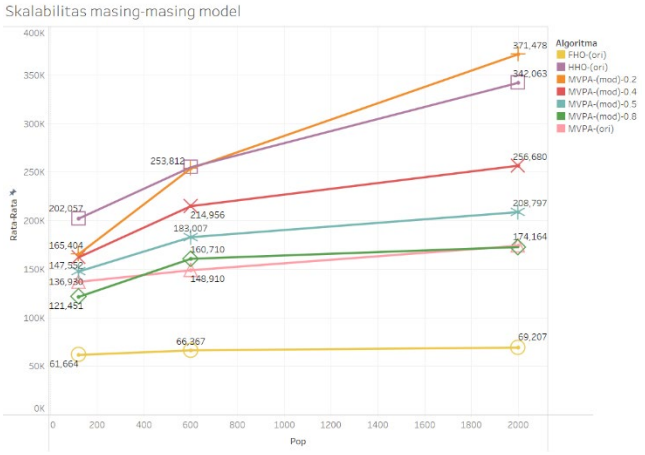
N Tim	Pop	Iter	FHO				HHO				ORIGINAL MVPA			
			Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
12	120	500	21390.1	20514.3	799.5	0.00133	72778.7	47983.2	14445.5	0.06644	48175.1	40180.1	7597.8	0.08560
12	120	1000	23764.6	20549.7	1866.7	0.00114	72903.3	57560.8	14129.2	0.06342	61021.7	45682.5	9417.6	0.07980
12	120	5000	22846.4	20467.5	1411.8	0.00116	78227.5	66657.4	7103.4	0.08026	54243.2	50688.2	4596.8	0.08229
40	600	500	24891.2	22079.1	1649.7	0.00601	79468.8	71060.2	**6821.2**	0.36980	62881.8	52579.6	9761.9	0.42174
40	600	1000	22449.3	21411.0	680.7	0.00616	107027.2	69691.2	22461.1	0.42771	52096.4	48311.6	5238.7	0.52387
40	600	5000	23469.7	21624.8	1440.1	0.00583	142207.7	103224.9	33539.0	0.39870	58048.5	53832.9	5493.1	0.39482
100	2000	500	**26153.5**	**24580.8**	1220.3	0.02326	161722.8	116387.0	36553.4	1.69491	61089.5	53046.3	6550.4	1.54096
100	2000	1000	23725.3	23143.3	**480.6**	0.02077	162456.9	111941.1	39577.7	1.80135	**70616.0**	53025.1	11241.1	1.32373
100	2000	5000	24413.0	23161.9	781.1	0.02448	**165908.0**	**128842.1**	38582.3	1.82289	59410.2	**57354.8**	**2886.7**	1.50740

MODIFIED MVPA ALPHA 0.2				*MODIFIED* MVPA ALPHA 0.4				*MODIFIED* MVPA ALPHA 0.6				*MODIFIED* MVPA ALPHA 0.8			
Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)	Best F	Mean	Std	Avg Time (s)
58617.5	51153.1	5403.08	0.07168	56780.5	50497.2	4390.6	0.07279	47894.6	45183.3	3050.6	0.07458	51874.6	42902.9	6807.6	0.07235
61491.3	54185.1	6346.73	0.07533	79204.6	59217.7	14016.3	0.08520	53450.0	44712.9	5788.0	0.07475	53814.9	43882.5	8252.9	0.07712
55958.9	50556.5	**3446.3**	0.07431	51719.7	47015.1	**2820.7**	0.07447	55468.9	52317.8	**2927.6**	0.08208	53324.4	46953.9	4996.0	0.07298
100778.1	84449.8	12940.8	0.37020	76568.6	67638.1	8930.3	0.38438	72232.4	61296.2	11414.9	0.42565	53307.8	49914.5	2916.1	0.40868
105898.6	94876.4	11296.4	0.39416	86341.1	77362.8	6502.1	0.36441	67039.3	59003.1	4758.8	0.38766	56114.4	48436.2	5184.1	0.36654
111706.4	98819.5	12070.1	0.37378	100472	79271.2	13010.7	0.38631	69841.5	61339.7	8079.4	0.43045	**88925.3**	62525.4	15344.1	0.39344
127801.2	111408.6	12151.3	1.25682	92958.4	77523.4	12096.7	1.35042	78329.1	67477.9	6483.0	1.35722	67694.1	52471.3	10504.4	1.28920
116886.8	104600.3	9857.0	1.22911	91118.9	78875.1	8438.9	1.23464	86841.4	70643.1	10367.6	1.28184	70527.3	54459.5	9867.2	1.25031
156474.0	**128452.9**	19576.1	1.29607	**112146.3**	**102599.0**	6098.6	1.41489	**100306.3**	**88753.4**	10385.2	1.26889	66344.5	**64219.1**	**1919.5**	1.36447

Berdasarkan uji coba yang telah dilakukan, berikut analisis efektifitas masing-masing algoritma jika dibandingkan dari skalabilitas, performa eksplorasi, dan hasil kualitatif:

1) PERBANDINGAN SKALABILITAS MASING-MASING ALGORITMA

Skalabilitas yang akan dibahas adalah seberapa besar pengaruh kenaikan populasi terhadap nilai *fitness* dari solusi yang diberikan untuk setiap algoritma. Berikut adalah grafik dari rata-rata *fitness* dari setiap algoritma terhadap perubahan ukuran populasi dari data hasil uji coba tabel I, tabel II, dan tabel III.



GAMBAR 5. Grafik rata-rata *fitness* tiap model untuk berbagai ukuran populasi dengan jumlah *epoch* 500

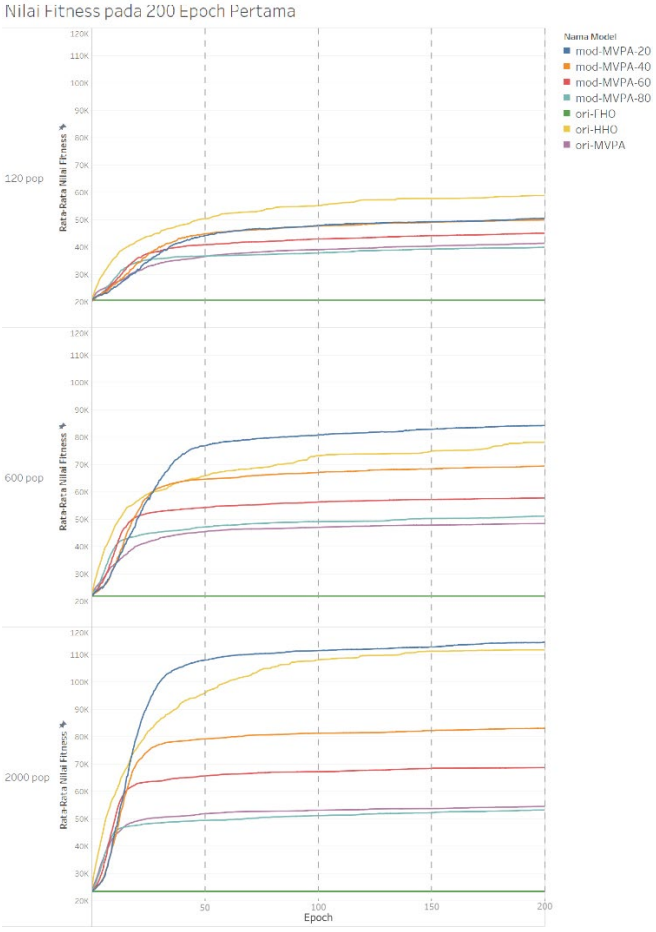
Dari grafik Gambar 5 dapat dilihat bahwa HHO dan *Modified* MVPA dengan *alpha* 0,2 memiliki skalabilitas yang cukup baik dibandingkan dengan algoritma yang lain. Selain itu, semakin kecil *alpha* pada *Modified* MVPA, semakin baik pula skalabilitasnya. Oleh karena itu, dapat disimpulkan nilai *alpha* 0,2 merupakan nilai terbaik untuk *Modified* MVPA.

Jika diperhatikan, skalabilitas yang ditunjukkan oleh *Modified* MVPA cukup baik. Namun, untuk ukuran populasi yang lebih kecil, algoritma *Modified* MVPA kesulitan dalam mendapatkan solusi yang baik. Hal ini menunjukkan bahwa *Modified* MVPA cukup bergantung pada ukuran populasi. Sedangkan algoritma HHO dapat memberikan hasil yang baik walaupun ukuran populasi yang diberikan relatif lebih kecil.

2) GRAFIK PERUBAHAN NILAI *FITNESS* BERBANDING DENGAN *EPOCH*

Gambar 6 menunjukkan perubahan rata-rata nilai *fitness* berbanding pada *epoch* untuk setiap populasi. Dari grafik yang ditunjukkan, dapat dilihat bahwa secara umum, ukuran populasi yang lebih besar akan mengurangi jumlah *epoch* yang dibutuhkan untuk mencapai titik konvergen, karena individu yang terlibat pada pencarian kandidat solusi terbaik lebih banyak. Dari keseluruhan percobaan, algoritma HHO dan *Modified* MVPA dengan alpha 0.2 adalah dua algoritma yang dapat menghasilkan solusi dengan nilai *fitness* yang cukup baik jika dibandingkan dengan FHO dan MVPA biasa.

Dari hasil uji coba, dua algoritma yang memberikan hasil lebih baik secara kuantitatif yaitu *Modified* MVPA dan HHO. Jika diperhatikan lebih lanjut, algoritma HHO memiliki kemampuan eksplorasi lebih tinggi, ditunjukkan dengan kemampuan algoritma dalam menghindari konvergensi dini serta menemukan kandidat solusi baru walaupun kandidat solusi terbaik saat ini memiliki nilai *fitness* yang sangat baik. Hal ini merupakan salah satu efek dari penggunakan fungsi *Lévy flights* yang cukup efektif pada tahapan eksplorasi[7].



GAMBAR 6. Grafik *Average Fitness* pada 200 *Epoch* pertama untuk tiap ukuran populasi

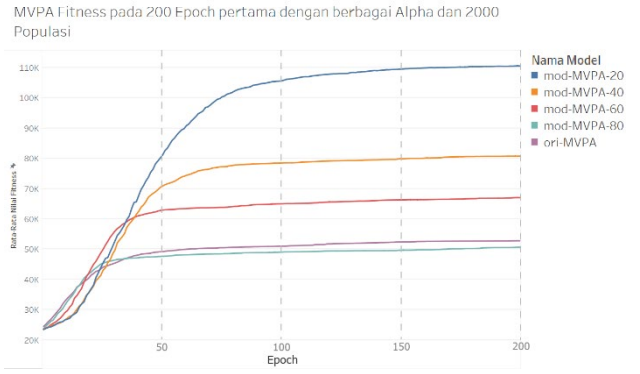
Pada Gambar 6, dapat dilihat perbedaan kecenderungan nilai *fitness* yang dihasilkan algoritma HHO dan *Modified* MVPA. Untuk jumlah populasi yang lebih sedikit, HHO memberikan hasil yang lebih superior dibandingkan *Modified* MVPA. Algoritma *Modified* MVPA juga terlihat kesulitan dalam melakukan eksplorasi lebih ketika memasuki fase konvergensi yang berakibat jika algoritma sudah cukup konvergen, penambahan jumlah *epoch* tidak memberikan pengaruh signifikan pada kualitas kandidat solusi yang diberikan. Berbeda halnya dengan algoritma HHO yang tetap mampu melakukan eksplorasi dan

menemukan kandidat solusi lebih baik meskipun telah memasuki fase konvergensi.

3) PERBANDINGAN MVPA DENGAN MODIFIED MVPA

Gambar 7 memberikan grafik perbandingan nilai *fitness* antara algoritma MVPA dan *Modified* MVPA. Pada grafik yang ditampilkan, secara umum terlihat bahwa algoritma *Modified* MVPA memberikan nilai *fitness* yang lebih baik dibandingkan dengan algoritma *original*, terutama pada pemberian *hyperparameter alpha* 0.2. Selain itu, dapat pula dilihat korelasi antara perubahan nilai *alpha* dengan performa algoritma. Nilai *alpha* yang lebih kecil cenderung memberikan hasil yang lebih baik.

Seperti yang ditunjukkan pada formula 8, nilai *alpha* berpengaruh pada kecenderungan improvisasi *player* pada algoritma *Modified* MVPA. Nilai *alpha* yang lebih kecil membuat *player* lebih cenderung melakukan improvisasi mengikuti *Franchise Player* pada tim terkait. Dominasi pengaruh *Franchise Player* yang lebih signifikan dibandingkan MVP, memberikan kesempatan *player* pada tim untuk dapat bergerak secara berkelompok pada area pencarian, namun tetap memasukkan faktor kandidat solusi terbaik (MVP) pada proses mutasi kandidat solusi.



GAMBAR 7. Grafik Perbandingan MVPA dan Mod MVPA dengan berbagai Alpha

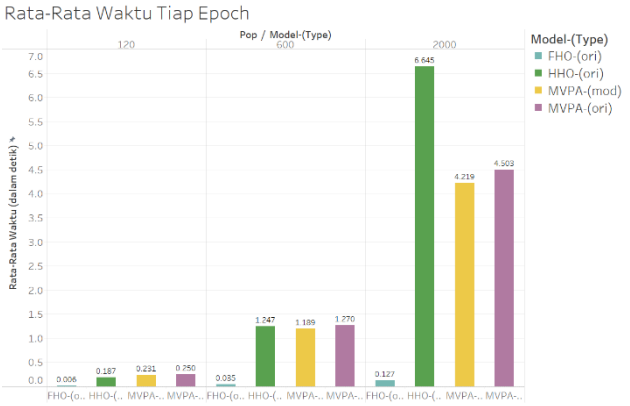
Pada Gambar 7, dapat pula terlihat fenomena *cold start* pada algoritma *Modified* MVPA. Jika diperhatikan kembali pada Gambar 6, fenomena ini dipengaruhi salah satunya oleh ukuran populasi. Fenomena ini terjadi dikarenakan fase *greediness* dan fase *elitism* yang tidak melakukan seleksi secara deterministik (formula 9 sebagai fungsi probabilitas). Solusi yang dinilai lebih buruk tetap memiliki kesempatan untuk bertahan pada kompetisi selanjutnya. Sedangkan, inisialisasi populasi dilakukan secara acak sehingga kebanyakan kandidat solusi memiliki nilai *fitness* yang relatif buruk. Hal ini mengakibatkan pada *epoch* awal, solusi yang memiliki nilai lebih baik tidak cepat untuk mendominasi seleksi dan fase *Team Competition*, serta membantu algoritma dalam memperluas fase eksplorasi sebelum akhirnya mencapai tahap konvergensi yang ditandai dengan mendominasinya kandidat solusi yang memiliki nilai *fitness* baik pada fase *Team Competition*.

Kelemahan dari algoritma *Modified* MVPA terletak pada kemampuan eksplorasinya di *epoch* menengah-akhir. Pada Gambar 7, terlihat algoritma *Modified* MVPA mengalami kesulitan dalam melakukan eksplorasi lebih ketika memasuki *epoch* ke-150. Diperlukan modifikasi yang meningkatkan daya jelajah algoritma agar mampu menemukan kandidat solusi yang lebih baik.

Meskipun hasil uji coba menunjukkan algoritma *Modified* MVPA konsisten memberikan hasil yang lebih baik dibandingkan algoritma MVPA, tetap diperlukan penelitian komperhensif terhadap modifikasi yang dilakukan. Pengujian konvergensi dan uji coba dengan berbagai permasalahan diperlukan untuk memberikan perbandingan akurat dan mengukur keberhasilan modifikasi pada algoritma MVPA.

4) PERBANDINGAN *EPOCH* TIME MASING-MASING ALGORITMA

Gambar 8 menunjukkan perbandingan rata-rata waktu yang dibutuhkan di setiap *epoch*.

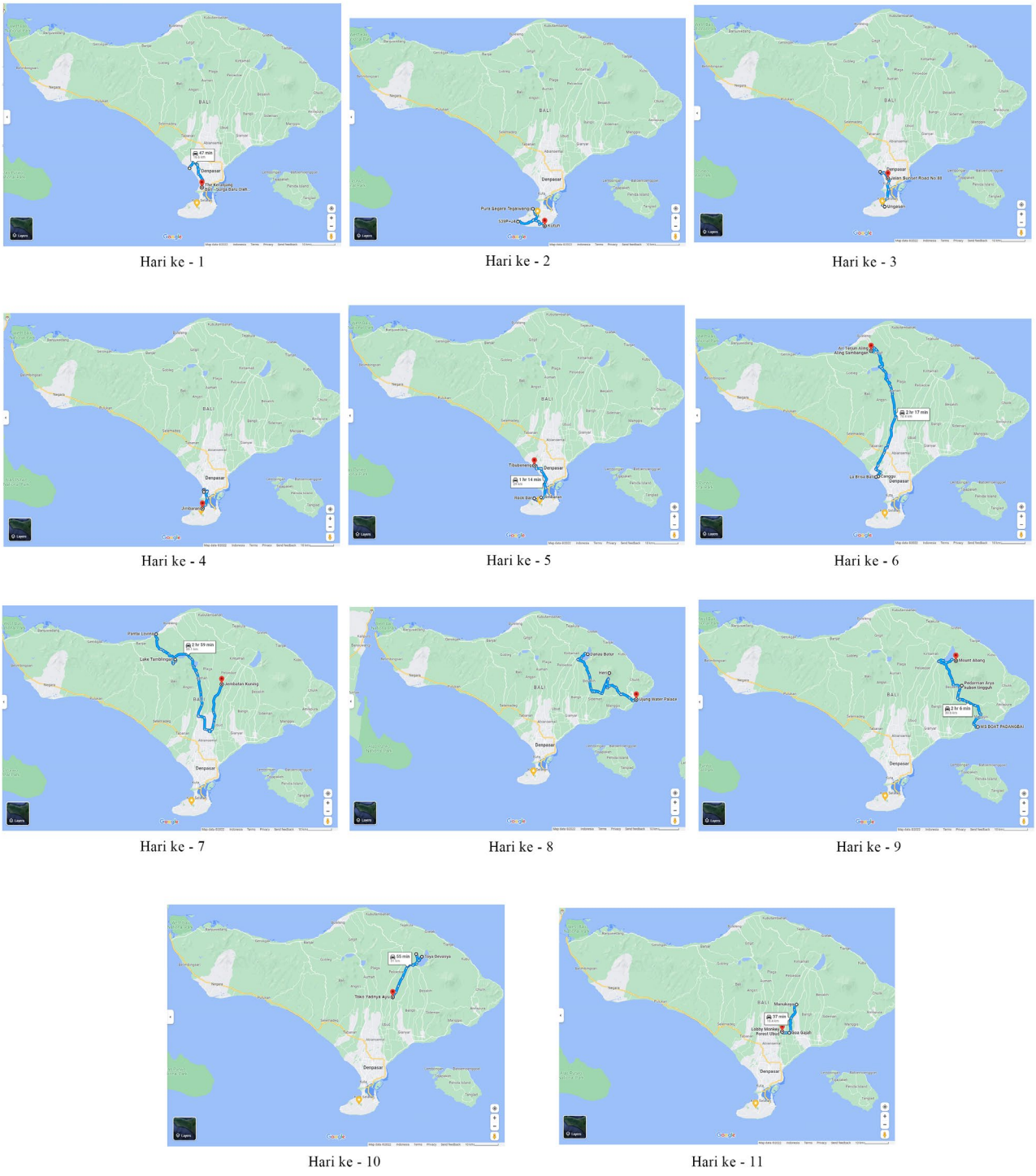


GAMBAR 8. Rata-rata *Epoch* time untuk masing-masing algoritma dan ukuran populasi

Dari data uji coba pada Gambar 8, ditunjukkan bahwa FHO memiliki *epoch* time yang lebih cepat dibandingkan dengan algoritma lainnya. Sedangkan, HHO memiliki *epoch* time yang lebih lama dibanding dengan algoritma lainnya. Algoritma MVPA dan *Modified* MVPA tidak memberikan perbedaan waktu yang cukup signifikan. Pada ukuran populasi 600, HHO dan MVPA memiliki *epoch* time yang relatif sama, namun pada ukuran populasi 2000, HHO memiliki *epoch* time sekitar 50% lebih tinggi dibandingkan dengan MVPA. Perbedaan waktu yang ditunjukkan oleh algoritma HHO bisa jadi dipengaruhi oleh variasi eksplorasi yang melibatkan perhitungan kompleks. Namun, berdasarkan Gambar 6, algoritma HHO mampu memberikan hasil yang baik dengan ukuran populasi yang lebih kecil. Sehingga, efektifitas algoritma HHO dapat diperoleh dengan menggunakan ukuran populasi minimal dan jumlah *epoch* yang lebih banyak.

5) VISUALISASI SOLUSI YANG DIHASILKAN

Rute yang ditampilkan pada visualisasi ini adalah hasil terbaik dari semua percobaan yang telah dilakukan. Hasil ini memiliki nilai *fitness* sekitar 212.000 dan didapatkan dengan algoritma HHO dengan ukuran populasi 2000 dan jumlah *epoch* 500 serta parameter λ=11 dan ϕ=3. Visualisasi solusi ditampilkan pada Gambar 9.



GAMBAR 9. **Visualisasi Rute Hasil Dari Algoritma HHO**

V. DISKUSI, KESIMPULAN, DAN BATASAN

Dari ketiga algoritma yang telah dicoba, HHO merupakan algoritma dengan hasil yang terbaik pada permasalahan TSP. Meskipun dengan waktu eksekusi yang lebih lama dari algoritma lain, hasil yang didapatkan berbeda cukup jauh jiak dibandingkan dengan kedua algoritma lainnya. Sedangkan FHO merupakan algoritma dengan hasil yang terburuk dari ketiga algoritma lainnya, tetapi dengan waktu eksekusi yang paling cepat. Algoritma MVPA yang telah dimodifikasi memiliki hasil yang lebih baik jika dibandingkan dengan MVPA biasa, tetapi masih belum bisa sebaik HHO. Untuk pemilihan algoritma, jika *environment* dan *resource* yang dimiliki mumpuni serta waktu bukan

menjadi penghalang, maka HHO adalah yang paling disarankan karena dapat menghasilkan solusi yang paling baik. Tetapi, ketika *environment* dan *resource* yang dimiliki terbatas serta tidak memiliki cukup banyak waktu, maka MVPA adalah algoritma yang disarankan karena dapat menghasilkan solusi yang cukup baik.

Uji coba yang dilakukan sangat terpusat pada penyelesaian permasalahan spesifik yang didefinisikan pada penelitian ini. Kurangnya variasi permasalahan menjadi salah satu kelemahan dalam melakukan perbandingan efektifitas antar algoritma. Namun, hal ini berada diluar fokus penelitian ini karena yang menjadi fokus penelitian hanyalah perbandingan performa algoritma pada permasalahan spesifik. Sedangkan, hal yang menjadi batasan penelitian pada algoritma modifikasi ialah, tidak adanya uji konvergensi dan variasi uji coba dengan berbagai permasalahan yang lebih komperhensif. Hal ini diperlukan untuk memvalidasi kebenaran dan hasil positif yang diberikan agar tidak bias pada salah satu permasalahan saja.

PERAN PENULIS

Christian Budhi Sabdana: Konseptualisasi, Metodologi dan Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review & Editing, Visualisasi;
Bryan Christopher: Konseptualisasi, Penyusunan Draf Asli, Visualisasi;
Jason Gerald Sutanto: Konseptualisasi, Metodologi dan Implementasi Sistem, Penyusunan Draf Asli, Penulisan Review & Editing, Visualisasi;
Lawrence Patrick Sianto: Kurasi Data, Sumber Daya, Administrasi Proyek, Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review & Editing;
Lukky Hariyanto: Konseptualisasi, Penyusunan Draf Asli, Visualisasi;
Nickolas Hartono: Konseptualisasi, Implementasi Sistem, Uji Coba, Penyusunan Draf Asli, Penulisan Review;

COPYRIGHT

DAFTAR PUSTAKA

[1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The traveling salesman problem: A computational study*. 2011.
[2] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, Aug. 2019, doi: 10.1016/j.future.2019.02.028.
[3] A. P. Engelbrecht, "Appendix A: Optimization Theory," *Comput Intell*, pp. 551–579, 2007, doi: 10.1002/9780470512517.app1.
[4] B. Galvan, G. D., P. J., M. Sefrioui, and G. Winter, "Parallel Evolutionary Computation for Solving Complex CFD Optimization Problems : A Review and Some Nozzle Applications," in *Parallel Computational Fluid Dynamics 2002*, Elsevier, 2003, pp. 573–604. doi: 10.1016/B978-044450680-1/50072-3.
[5] M. Azizi, S. Talatahari, and A. H. Gandomi, "Fire Hawk Optimizer: a novel metaheuristic algorithm," *Artif Intell Rev*, vol. 56, no. 1, pp. 287–363, Jan. 2023, doi: 10.1007/s10462-022-10173-w.
[6] H. R. E. H. Bouchekara, "Most Valuable Player Algorithm: a novel optimization algorithm inspired from sport," *Operational Research*, vol. 20, no. 1, pp. 139–195, Mar. 2020, doi: 10.1007/s12351-017-0320-y.
[7] *Nature-Inspired Optimization Algorithms*. Elsevier, 2014. doi: 10.1016/C2013-0-01368-0.
[8] J. C. Bean, "Genetic Algorithms and Random Keys for Sequencing and Optimization," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 154–160, May 1994, doi: 10.1287/ijoc.6.2.154.
[9] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theor Comput Sci*, vol. 4, no. 3, pp. 237–244, Jun. 1977, doi: 10.1016/0304-3975(77)90012-3.
[10] S. S. Skiena, *The Algorithm Design Manual*. 1998.
[11] E. Guanabara, K. Ltda, E. Guanabara, and K. Ltda, *The Traveling Salesman Problem and Its Variations*, vol. 12. in Combinatorial Optimization, vol. 12. Boston, MA: Springer US, 2007. doi: 10.1007/b101971.
[12] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "A new genetic algorithm for solving optimization problems," *Eng Appl Artif Intell*, vol. 27, pp. 57–69, 2014, doi: 10.1016/j.engappai.2013.09.013.
[13] "Google Maps Platform Documentation | Places API | Google Developers." https://developers.google.com/maps/documentation/places/web-service (accessed Apr. 20, 2023).
[14] "Google Maps Platform Documentation | Distance Matrix API | Google Developers." https://developers.google.com/maps/documentation/distance-matrix (accessed Apr. 20, 2023).
[15] C. Fu, L. Zhang, X. Wang, and L. Qiao, "Solving TSP problem with improved genetic algorithm," 2018, p. 40057. doi: 10.1063/1.5039131.
[16] T. V Maia, "Avoidance Learning," in *Encyclopedia of the Sciences of Learning*, Boston, MA: Springer US, 2012, pp. 403–406. doi: 10.1007/978-1-4419-1428-6_968.
[17] J. M. Bland and D. G. Altman, "Statistics notes: Measurement error," *BMJ*, vol. 312, no. 7047, p. 1654, Jun. 1996, doi: 10.1136/bmj.312.7047.1654.
[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, IEEE, pp. 1942–1948. doi: 10.1109/ICNN.1995.488968.
[19] N. Van Thieu and S. Mirjalili, "MEALPY: a Framework of The State-of-The-Art Meta-Heuristic Algorithms in Python." zenodo, Jun. 22, 2022.