BAB I
PENDAHULUAN

1.1 Latar Belakang

Optimisasi adalah proses pencarian solusi ideal dari suatu masalah dengan memberikan beberapa solusi yang memungkinkan (Dehghani et al., 2022). Pada optimisasi, permasalahan yang ada terbagi menjadi tiga bagian, yaitu variabel keputusan, batasan masalah, dan fungsi objektif (Dehghani et al., 2022). Tujuan dari proses optimisasi adalah untuk menghitung variabel-variabel keputusan berdasarkan batasan-batasan masalah yang ada untuk mencapai nilai fungsi objektif yang optimal. Ketika melakukan optimisasi, terdapat dua teknik yang biasa dilakukan yaitu pendekatan deterministik dan stokastik (Dehghani et al., 2022).

Deterministik adalah model pendekatan yang menghasilkan solusi yang tunggal pada setiap permasalahan yang diberikan (Pinsky and Karlin, 2010). Pendekatan deterministik disebut juga sebagai pendekatan matematis. Pendekatan deterministik sangat efektif digunakan untuk menyelesaikan permasalahan sederhana dan tidak rumit seperti persamaan linear atau permasalahan yang berdimensi sedikit. Tetapi metode deterministik tidak efektif untuk menyelesaikan permasalahan yang rumit seperti persamaan tak linear atau persamaan yang berdimensi banyak (Dehghani et al., 2022). Oleh karena itu, untuk menyelesaikan permasalahan yang lebih rumit digunakan pendekatan stokastik.

Tidak seperti deterministik, stokastik merupakan model pendekatan yang melibatkan variabel bernilai acak untuk menghasilkan serangkaian solusi yang ditimbang berdasarkan kemungkinan atau probabilitasnya (Pinsky and Karlin, 2010). Pendekatan stokastik disebut juga sebagai pendekatan metaheuristik. Pengembangan algoritma metaheuristik biasanya terinspirasi dari fenomena alam, perilaku satwa liar, pertumbuhan tanaman, aktivitas manusia, atau proses evolusi alami lainnya. Contoh dari metode metaheuristik adalah algoritma *particle swarn organization (PSO)*, *honey badger algorithm (HBA)*, dan algoritma *driving training based optimization (DTBO)*.

PSO adalah teknologi komputasi evolusioner adaptif, algoritma cerdas yang berbasis pencarian populasi (Kennedy and Eberhart, 1995). PSO dikemukakan pertama kali oleh James Kennedy dan Russell C. Eberhart pada tahun 1995 dan di modifikasi lebih lanjut oleh Yuhui Shi pada tahun 1998 (Shi and Eberhart, 1998). algoritma ini terinspirasi oleh pergerakan sekelompok burung yang sedang mencari makan. Kelebihan dari PSO adalah konvergensi yang cepat dan tidak banyak melakukan pengaturan parameter sehingga mudah untuk di implementasi (Wang et al., 2021). Adapun kelemahan dari PSO adalah algoritma ini mudah terjebak dalam pencarian lokal ketika melakukan perhitungan suatu permasalahan yang berdimensi tinggi dan memiliki tingkat konvergensi yang rendah (Li et al., 2014).

HBA atau algoritma ratel adalah metode pendekatan metaheuristik berbasis *swarm-based optimization* yang terinspirasi dari perilaku ratel atau *honey badger* (*Mellivora capensis*) dalam mencari makan di alam liar (Hashima et al., 2022). Konsep ini dikemukakan oleh Fatma A. Hashima, Essam

H. Houssein, Kashif Hussain, Mai S. Mabrouk, dan Walid Al-Atabany pada tahun 2021. Kelebihan dari HBA adalah algoritma ini efektif memecahkan masalah dengan ruang pencarian yang kompleks serta memiliki kecepatan konvergensi dan keseimbangan eksplorasi-eksplotasi yang lebih unggul dibanding 10 metode metaheuristik terdahulu (Hashima et al., 2022). HBA juga sangat efektif digunakan pada permasalahan matematika yang memiliki ruang pencarian yang kompleks (Hashima et al., 2022). Namun, kelemahan dari HBA adalah algoritma ini sering terjebak dalam perhitungan optimisasi nilai lokal ketika melakukan penyelesaian masalah optimisasi yang kompleks (Yasear and Ghanimi, 2022).

DTBO adalah metode pendekatan meutaheuristik yang berbasis populasi yang didasarkan pada pencarian acak di ruang pemecahan masalah (Dehghani et al., 2022). Konsep ini dikemukakan oleh Mohammad Dehghani, Eva Trojovská dan Pavel Trojovský pada papernya yang terbit pada Juni 2022. Konsep DTBO ini terinspirasi dari aktivitas siswa pengemudi dan instruktur dalam kelas kursus pengemudi. Kelebihan dari DTBO adalah memiliki kemampuan yang tinggi dalam melakukan eksplorasi, eksploitasi, dan menyeimbangkan keduanya untuk meningkatkan akurasi perhitungan selama melakukan optimisasi untuk meraih solusi optimal (Dehghani et al., 2022). Adapun kelemahan dari DTBO adalah algoritma ini membutuhkan waktu perhitungan yang lebih lama dibanding dengan algoritma sejenis seperti PSO, terutama bila permasalahan tersebut memiliki dimensi yang banyak. Hal ini disebabkan karena algoritma DTBO terdapat 3 fase perhitungan di setiap iterasinya sehingga membutuhkan waktu perhitungan yang lama.

Pada penelitian ini, penulis akan melakukan perbandingan dari PSO, DTBO, dan HBA dalam melakukan optimisasi 10 fungsi tolak ukur (*benchmark functions*) berdasarkan hasil solusi optimum yang dihasilkan. Alasan penggunaan HBA dan DTBO dalam penelitian ini adalah dikarenakan kedua fungsi ini relatif baru dan belum ada tulisan akademik yang membandingkan kedua algoritma tersebut. Adapun alasan pemakaian PSO pada penelitian ini adalah karena algoritma tersebut menjadi Hasil perbandingan tersebut nantinya akan menentukan algoritma terbaik dalam mencari solusi optimal dalam menyelesaikan permasalahan matematika tersebut.

1.2 Rumusan Masalah

Dari algoritma PSO, HBA, dan DTBO, algoritma manakah yang menghasilkan nilai solusi optimal yang paling mendekati nilai solusi minimal dari 10 *benchmark functions* yang disediakan?

1.3 Batasan Masalah

Pada penelitian ini penulis membandingkan tiga algoritma metaheuristik yaitu PSO, HBA, dan DTBO dalam melakukan optimisasi suatu permasalahan matematika. Adapun permasalahan matematika yang dipakai adalah 10 *benchmark functions* yang diambil dari paper yang berjudul "*Evolutionary Programming Made Faster*" (Yao et al., 1999).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk menentukan dari ketiga algoritma tersebut yang menghasilkan nilai solusi optimum yang paling mendekati nilai solusi sebenarnya dalam menyelesaikan 10 *benchmark functions* yang disediakan.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan informasi kepada pembaca berupa algoritma terbaik dari ketiga algoritma yang dibandingkan dalam menyelesaikan 10 *benchmark functions*. Penulis berharap agar algoritma tersebut dapat digunakan untuk melakukan penyelesaian permasalahan matematika yang lain.

BAB II

TINJAUAN PUSTAKA

2.1 *Particle Swarn Optimization* (PSO)

PSO merupakan algoritma pencarian berbasis populasi yang di inisiasi oleh kumpulan solusi acak (Shi, 2004). Solusi-solusi acak tersebut biasa disebut dengan partikel. Partikel pada PSO di inisiasi dalam matriks (2.1) dan (2.2).

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & x_{N3} & \dots & x_{Nn} \end{bmatrix}_{N \times n} \tag{2.1}$$

$$X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}] \tag{2.2}$$

di mana X merupakan matriks populasi partikel dan X_i merupakan kumpulan solusi partikel ke-i.

Setiap partikel dalam PSO memiliki ”kecepatan” (*velocity*) tersendiri yang disesuaikan secara dinamis tergantung dengan perilaku mereka di iterasi sebelumnya. ”Kecepatan” pada PSO ini bukan kecepatan dalam arti sesungguhnya, namun untuk mengistilahkan perpindahan posisi suatu partikel. Inisiasi awal dari ”kecepatan” setiap partikel digambarkan pada matriks (2.3).

$$V_i = [v_{i1}, v_{i2}, v_{i3}, \cdots, v_{in}] \tag{2.3}$$

di mana V_i merupakan kumpulan perpindahan posisi partikel ke-i.

Partikel-partikel tersebut nantinya akan cenderung ”terbang” atau berpindah menuju area pencarian yang lebih baik di setiap iterasinya. Perubahan ”kecepatan” pada suatu partikel akan menentukan performa PSO dalam melakukan pencarian solusi permasalahan. Perubahan ”kecepatan” tersebut digambarkan pada persamaan (2.4) dan (2.5).

$$v_{in} = w \cdot v_i + c_1 r_1 (x_{pbest_{in}} - x_{in}) + c_2 r_2 (x_{gbest_{in}} - x_{in}) \tag{2.4}$$

$$x_i = x_i + v_i \tag{2.5}$$

di mana v_{in} merupakan perpindahan posisi partikel ke-n di iterasi ke-i, x_{in} merupakan partikel ke-n di iterasi ke-i, w atau ”berat inersia” partikel merupakan nilai acak dari interval [0.5,1], r_1 dan r_2 merupakan nilai acak dengan interval [0,1], c_1 dan c_2 merupakan nilai konstan (0.1), dan $x_{pbest_{in}}$ dan $x_{gbest_{in}}$ masing-masing merupakan nilai dan indeks posisi terbaik di iterasi ke-i.

”Berat inersia” atau w dalam persamaan ini merupakan salah satu bentuk modifikasi dari persamaan PSO yang berfungsi untuk mengeliminasi kebutuhan untuk menetapkan nilai V_{max} di setiap iterasi sehingga dapat menyederhanakan perhitungan (Shi, 2004).

2.2 *Honey Badger Algorithm* (HBA)

Seperti yang telah dibahas pada Bab I, HBA merupakan algoritma yang terinspirasi dari perilaku ratel dalam mencari makan. HBA merupakan salah satu metode metaheuristik yang berbasis populasi. Sama seperti algoritma berbasis populasi pada umumnya, dalam HBA juga melakukan inisiasi populasi berupa matriks kandidat solusi. Adapun matriks kandidat solusi digambarkan pada persamaan (2.6) dan (2.7)

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & x_{M3} & \dots & x_{Mm} \end{bmatrix}_{M \times m} \tag{2.6}$$

$$X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{im}] \tag{2.7}$$

di mana X merupakan populasi kandidat solusi x dan X_i merupakan kumpulan solusi kandidat untuk ratel ke-i.

Langkah-langkah yang dilakukan algoritma selama melakukan pencarian solusi adalah sebagai berikut:

Langkah pertama: inisialisasi jumlah populasi solusi kandidat. Pada langkah ini, algoritma akan melakukan inisialisasi pada jumlah populasi ratel (M) dengan menggunakan persamaan (2.8)

$$x_i = lb + r_1 \cdot (ub - lb) \tag{2.8}$$

di mana x_i merupakan kumpulan kandidat solusi dengan populasi M untuk ratel ke-i, lb dan ub adalah batas bawah dan batas atas untuk melakukan pencarian domain, serta r_1 adalah nilai acak dengan interval [0,1].

Langkah kedua: mendefinisikan intensitas (I). Intensitas merupakan kumpulan populasi yang menyatakan intensitas bau dari mangsa. Nilai intensitas dipengaruhi oleh kekuatan mangsa (K) dan jaraknya terhadap posisi ratel (J_i). Semakin jauh jarak dan semakin kecil kekuatan mangsa, maka nilai intensitas nya semakin kecil dan begitu pun sebaliknya. Adapun definisi intensitas digambarkan dalam persamaan (2.9)

$$\begin{aligned} I_i &= r_2 \cdot \frac{K}{4\pi \cdot J_i^2} \\ K &= (x_i - x_{i+1})^2 \\ J_i &= x_{mangsa} - x_i \end{aligned} \tag{2.9}$$

di mana r_2 merupakan nilai acak dengan interval [0,1] dan I_i merupakan nilai intensitas mangsa ke-i

Langkah ketiga: menentukan atau memperbaharui faktor densitas. Nilai faktor densitas akan memengaruhi nilai x_{baru} ketika melakukan pembaharuan posisi agen pada langkah kelima. Faktor densitas di definisikan dengan persamaan (2.10)

$$\alpha = Q \cdot \exp(\frac{-t}{T}) \tag{2.10}$$

di mana α merupakan faktor densitas, Q adalah konstanta dengan nilai ≤ 1 (nilai default = 2), t adalah nomor iterasi, dan T adalah nilai ketetapan maksimal iterasi.

Langkah keempat: mendefinisikan flag. Flag adalah sebuah konstanta yang memiliki nilai 1 atau -1. Nilai flag ini nantinya digunakan untuk memilih fase yang ada pada langkah kelima dalam memperbaharui posisi agen pencarian sehingga dapat melakukan pemindaian ruang pencarian secara optimal. Penentuan nilai flag digambarkan dengan persamaan (2.11).

$$F = \begin{cases} 1, r_3 \leq 0.5 \\ -1, r_3 \geq 0.5 \end{cases} \tag{2.11}$$

di mana F adalah nilai flag dan r_3 adalah nilai acak dengan interval [0,1].

Langkah kelima: memperbaharui posisi dari agen pencarian. Dalam melakukan pembaharuan posisi agen, algoritma HBA melakukan dua fase perhitungan yang masing-masing disebut sebagai ”fase menggali” dan ”fase mencari madu”. Tidak seperti DTBO yang menggunakan semua fase perhitungan dalam setiap iterasi nya, pada HBA hanya salah satu fase perhitungan yang digunakan dalam setiap iterasi. Penentuan tersebut akan dipengaruhi oleh nilai F. Jika F adalah 1, maka fase yang digunakan pada iterasi tersebut adalah fase menggali. Sebaliknya, jika nilai F adalah -1, maka fase yang digunakan pada iterasi tersebut adalah fase mencari madu.

1. Fase 1: menggali

   Pada fase ini, pembaharuan posisi agen menggunakan persamaan (2.12).

   $$x_{baru} = x_{mangsa} + F \cdot \beta \cdot I \cdot x_{mangsa} + F \cdot r_4 \cdot \alpha \cdot J_i \cdot |\cos(2\pi r_5) \cdot [1 - \cos(2\pi r_6)]| \tag{2.12}$$

   di mana x_{baru} adalah posisi terbaru dari agen pencarian, x_{mangsa} merupakan posisi terbaik agen pencarian saat ini, β merupakan konstanta dengan nilai ≤ 1 (nilai default = 6), α adalah nilai faktor densitas, J_i adalah jarak antara mangsa dengan ratel, dan r_4, r_5, dan r_6 masing-masing merupakan nilai acak yang berbeda dengan interval [0,1].

2. Fase 2: mencari madu

   Pada fase ini, pembaharuan posisi agen menggunakan persamaan (2.12).

   $$x_{baru} = x_{mangsa} + F \cdot r_7 \cdot \alpha \cdot J_i \tag{2.13}$$

   di mana x_{baru} adalah posisi terbaru dari agen pencarian, x_{mangsa} merupakan posisi terbaik

agen pencarian saat ini, dan r_7 adalah nilai acak dengan interval [0,1].

2.3 *Driving Training Based Optimization* (DTBO)

DTBO adalah algoritma yang berbasis populasi. Algoritma berbasis populasi adalah algoritma yang mempertahankan seluruh solusi kandidat yang di mana setiap solusi kandidat tersebut berhubungan erat dengan setiap variabel unik dalam fungsi permasalahan (Das et al., 2010). Adapun populasi dalam DTBO yang digambarkan dengan matriks berikut (Dehghani et al., 2022).

$$D = \begin{bmatrix} D_1 \\ \vdots \\ D_i \\ \vdots \\ D_M \end{bmatrix}_{M\times m} = \begin{bmatrix} d_{11} & \dots & d_{1j} & \dots & d_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{i1} & \dots & d_{ij} & \dots & d_{im} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{M1} & \dots & d_{Mj} & \dots & d_{Mm} \end{bmatrix}_{M\times m} \tag{2.14}$$

Matriks (2.14) disebut sebagai matriks populasi di mana D merupakan kumpulan populasi entri DTBO, D_i merupakan kandidat solusi ke-i, d_{ij} merupakan nilai variabel ke-j yang ditentukan oleh solusi kandidat ke-i, M adalah banyaknya agen pencarian, dan m adalah banyaknya dimensi. Untuk posisi awal entri, pada awal implementasi dilakukan inisiasi secara acak dengan menggunakan persamaan berikut.

$$d_{ij} = lb + r \cdot (ub - lb) \tag{2.15}$$

di mana r adalah nilai acak dari interval [0,1], serta lb dan ub berturut-turut merupakan batas bawah dan batas atas dalam melakukan pencarian domain.

Setiap kandidat solusi memberikan nilai pada variabel masalah yang nantinya akan digunakan dan dievaluasi dalam fungsi objektif. Oleh karena itu, setiap nilai akan dihitung pada fungsi objektif yang sesuai dengan setiap kandidat solusi nya yang digambarkan dengan vektor persamaan (2.16)

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_M \end{bmatrix}_{M\times 1} = \begin{bmatrix} F(D_1) \\ \vdots \\ F(D_i) \\ \vdots \\ F(D_M) \end{bmatrix}_{M\times 1} \tag{2.16}$$

di mana F merupakan vektor fungsi objektif dan F_i merupakan nilai fungsi objektif yang diberikan oleh kandidat solusi ke-i. Nilai yang akan didapatkan dari perhitungan vektor fungsi objektif ini adalah nilai entri terbaik untuk fungsi objektif. Nilai tersebut dikenal sebagai D_{best}. Nilai D_{best} akan diperbaharui di setiap iterasi.

Dalam DTBO, solusi kandidat akan diperbaharui dalam tiga fase berbeda, yaitu fase pertama: berlatih bersama instruktur pengemudi, fase kedua: siswa mencontoh arahan instruktur, dan fase ketiga: latihan mandiri.

2.3.1 Fase pertama: berlatih bersama instruktur pengemudi

Fase ini berbasis kepada pemilihan instruktur pengemudi oleh para siswa pengemudi. Dalam matriks populasi D, sejumlah entri terbaik dianggap sebagai instruktur pengemudi sedangkan sisanya dianggap sebagai siswa pengemudi. Pemilihan dan pembelajaran instruktur pengemudi akan mengarahkan entri lain untuk bergerak ke area lain untuk mencari ruang. Pergerakan ini akan meningkatkan kemampuan eksplorasi pada DTBO dalam pencarian dan menemukan area yang optimal. Matriks populasi I digambarkan dalam persamaan (2.17)

$$I = \begin{bmatrix} I_1 \\ \vdots \\ I_i \\ \vdots \\ I_{M_I} \end{bmatrix}_{M_I \times m} = \begin{bmatrix} I_{11} & \dots & I_{1j} & \dots & I_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ I_{i1} & \dots & I_{ij} & \dots & I_{im} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ I_{M_I 1} & \dots & I_{M_I j} & \dots & I_{M_I m} \end{bmatrix}_{M_I \times m} \tag{2.17}$$

di mana I merupakan matriks instruktur pengemudi. Entri dari matriks I didapat dari matriks populasi D yang barisnya telah diurutkan berdasarkan nilai $F(D_i)$ terkecil. Banyaknya baris matriks D yang digunakan sebagai entri matriks I adalah sebesar M_I baris. Adapun besarnya nilai M_I ditentukan oleh persamaan (2.18)

$$M_I = \lfloor 0,1 \cdot M \cdot (1 - n/N) \rfloor \tag{2.18}$$

dengan n merupakan banyaknya iterasi dan N merupakan nilai iterasi maksimal yang ditetapkan.

Model matematika yang digunakan pada fase pertama menggunakan persamaan (2.19) dalam menghitung posisi baru setiap entri dengan merujuk persamaan (2.20) ketika mengganti posisi yang lama.

$$d_{i,j}^{(1)} = \begin{cases} d_{i,j} + r \cdot (I_{k_{i,j}} - L \cdot d_{i,j}), F_{I_{k_i}} < F_i; \\ d_{i,j} + r \cdot (d_{i,j} - I_{k_{i,j}}), F_{I_{k_i}} \geq F_i \end{cases} \tag{2.19}$$

$$D_i = \begin{cases} D_i^{(1)}, F_i^{(1)} < F_i; \\ D_i, F_i^{(1)} \geq F_i \end{cases} \tag{2.20}$$

di mana $D_i^{(1)}$ merupakan nilai perhitungan baru berdasarkan kandidat solusi ke-i, L merupakan nilai acak yang dipilih dari himpunan $\{1, 2\}$, dan $I_{k_{i,j}}$ yang di mana $k_{i,j}$ merupakan nilai acak dari himpunan $\{1, 2, \cdots, M_I\}$, merupakan instruktur pengemudi yang dipilih secara acak untuk melatih siswa ke-i yang berada di dimensi ke-j.

2.3.2 Fase kedua: siswa pengemudi meniru instruktur

Sesuai dengan namanya, fase kedua dari D ini terinspirasi dari aktivitas siswa pengemudi yang meniru instruktur di mana siswa mencoba membuat model berdasarkan gerakan dan kelihaian in-

struktur selama berlatih. Dalam fase ini nantinya akan memindahkan posisi entri D ke tempat yang berbeda dengan tujuan untuk mencari ruang dan meningkatkan kemampuan eksplorasi. Fase ini di gambaran dengan persamaan (2.21). Jika persamaan tersebut meningkatkan nilai fungsi objektif, maka nilai posisi lama akan diganti berdasarkan persamaan (2.22).

$$d_{i,j}^{(2)} = P \cdot d_{i,j} + (1 - P) \cdot I_{k_(i,j)} \tag{2.21}$$

$$D_i = \begin{cases} D_i^{(2)}, F_i^{(2)} < F_i; \\ D_i, F_i^{(2)} \geq F_i \end{cases} \tag{2.22}$$

di mana $Di^{(2)}$ merupakan nilai perhitungan baru berdasarkan kandidat solusi ke-i dan P merupakan indeks pola yang digambarkan dengan persamaan (2.23).

$$P = 0,01 + 0,9 \left(1 - \frac{n}{N}\right) \tag{2.23}$$

2.3.3 Fase ketiga: melakukan praktik mandiri

Fase ini merupakan fase terakhir dari DTBO. Fase ini terinspirasi dari siswa yang melakukan praktik secara mandiri untuk meningkatkan kemampuan dalam mengemudi. Dalam fase ini, setiap entri D akan mencoba menemukan nilai posisi yang lebih baik berdasarkan pencarian lokal di sekitar posisi saat ini. Fase ini digambarkan dalam persamaan (2.24). Jika persamaan tersebut meningkatkan nilai fungsi objektif, maka nilai posisi lama akan diganti berdasarkan persamaan (2.25).

$$d_{i,j}^{(3)} = d_{i,j} + (1 - 2r) \cdot C \cdot \left(1 - \frac{n}{N}\right) \cdot d_{i,j} \tag{2.24}$$

$$D_i = \begin{cases} D_i^{(3)}, F_i^{(3)} < F_i; \\ D_i, F_i^{(3)} \geq F_i \end{cases} \tag{2.25}$$

di mana $d_{i,j}^{(3)}$ merupakan nilai perhitungan baru berdasarkan kandidat solusi ke-i dan dimensi ke-j, dan C merupakan nilai konstan (0,05).

Setelah melakukan ketiga fase tersebut, maka perhitungan DTBO dalam iterasi tersebut telah selesai. Perhitungan pada iterasi selanjutnya akan dilakukan dengan menggunakan matriks populasi D yang telah diperbaharui. Proses perhitungan ini akan terus mengulang perhitungan di persamaan (2.17) sampai persamaan (2.25) hingga telah mencapai nilai iterasi maksimal yang telah ditentukan sebelumnya. Hasil akhir dari algoritma ini merupakan nilai D_i dari perhitungan di iterasi terakhir (D_{best}).

2.4 *Benchmark Functions*

Benchmark functions atau disebut juga sebagai fungsi tolak ukur adalah kumpulan dari 23 fungsi permasalahan matematika. Benchmark functions pertama kali dikemukakan oleh Xin Yao, Yong Liu, dan Guangming Lin pada tahun 1999 (Yao et al., 1999). Tujuan dari *benchmark functions* adalah untuk menguji dan mengukur suatu algoritma metaheuristik dalam melakukan pencarian solusi minimum. Secara garis besar, terdapat tiga jenis fungsi yang terdapat pada *benchmark functions*, yaitu fungsi unimodal, fungsi multimodal berdimensi banyak (*high-dimensional multimudal*), dan fungsi multimodal berdimensi sedikit (*fixed-dimensional multimodal*).

Fungsi unimodal adalah fungsi yang hanya mempunyai satu titik maksimum (puncak) dan satu titik minimum (lembah) (Muhayat and Kuswardi, 2008). Kebalikan dari unimodal, fungsi multimodal adalah sebuah fungsi yang memiliki beberapa titik maksimum dan titik minimum.

Dari 23 fungsi yang ada, penulis hanya menggunakan 10 fungsi saja. Masing-masing fungsi yang dipakai akan mewakili ketiga jenis fungsi yang ada pada *benchmark functions*. Setiap fungsi memiliki jumlah dimensi (n), nilai batas atas dan batas bawah (S), serta nilai solusi minimum (f_{min}) tersendiri. Adapun 10 fungsi yang digunakan tertera pada tabel (2.1).

Tabel 2.1. *Benchmark functions*

Fungsi uji	n	S	f_{min}	**Jenis**
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0	*Unimodal*
$f_2(x) = max_i\{\|x_i\|, 1 \leq i \leq n\}$	30	[-100,100]	0	*Unimodal*
$f_3(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0	*Unimodal*
$f_4(x) = \sum_{i=1}^{n} ix_i^4 + random[0, 1)$	30	[-1,28;1,28]	0	*Unimodal*
$f_5(x) = \sum_{i=1}^{n} -x_i \sin\sqrt{\|x_i\|}$	30	[-500,500]	-12569,5	*High-dimensional multimodal*
$f_6(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos 2\pi x_i + 10]$	30	[-5,12;5,12]	0	*High-dimensional multimodal*
$f_7(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)$	30	[-32,32]	0	*High-dimensional multimodal*
$f_8(x) = 4x_1^2 - 2,1x_1^4 + \frac{1}{3}x_1^6 - 4x_2^2 + 4x_2^4$	2	[-5,5]	-1,0316285	*Fixed-dimensional multimodal*
$f_9(x) = [(1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	3	*Fixed-dimensional multimodal*
$f_{10}(x) = -\sum_{i=1}^{n} c_i \exp[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2]$	4	[0,1]	-3,86	*Fixed-dimensional multimodal*

Khusus untuk fungsi f_{10}, terdapat variabel a, c, dan p dalam perhitungannya. Fungsi f_{10} disebut juga sebagai fungsi Hartman. Variabel a, c, dan p merupakan variabel konstanta berbentuk array yang memiliki nilai unik di setiap dimensi nya. Adapun nilai a, c, dan p tertera pada tabel (2.2).

Tabel 2.2. Nilai a, c, dan p berdasarkan dimensi nya

i	$a_{ij}, j = 1, 2, 3$	c_i	$p_{ij}, j = 1, 2, 3$
1	3, 10, 30	1	0,3689; 0,117; 0,2673
2	0,1; 10; 35	1,2	0,4699; 0,4387; 0,747
3	3, 10, 30	3	0,1091; 0,8732; 0,5547
4	0,1; 10; 35	3,2	0,03815; 0,5743; 0,8828

Di mana i dan j merupakan dimensi dari variabel array a dan p.

BAB III
METODE PENELITIAN

3.1 Bentuk Penelitian

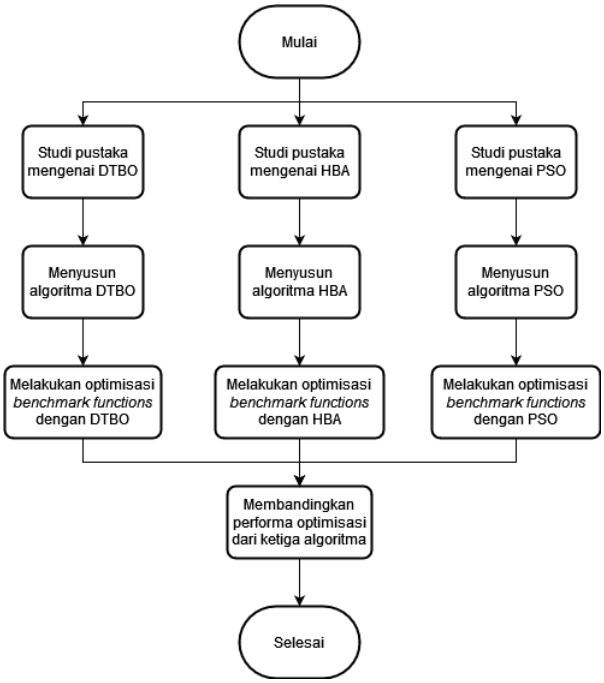
Bentuk penelitian yang dilakukan oleh penulis dalam penelitian ini adalah penelitian yang bersifat kuantitatif. Hal ini dikarenakan dalam penelitian ini penulis lebih fokus dalam melakukan perhitungan, yaitu melakukan optimisasi dari suatu persamaan dengan algoritma PSO, HBA, dan DTBO lalu membandingkan performa ketiga persamaan tersebut dalam melakukan optimisasi.

3.2 Metode Penyelesaian Masalah

Metode yang digunakan dalam penelitian ini adalah dengan membuat algoritma PSO, HBO dan DTBO. Setelah itu, penulis membandingkan performa ketiga metode metaheuristik tersebut dalam melakukan optimisasi dengan menggunakan fungsi tolak ukur dari paper "*Evolutionary Programming Made Faster*" (Yao et al., 1999).

3.2.1 Langkah Penyelesaian

Langkah penyelesaian yang dilakukan pada penelitian ini digambarkan dalam diagram alir pada gambar 3.1. Adapun penjelasan dari langkah penyelesaian tersebut adalah sebagai berikut.



Gambar 3.1. Diagram alir langkah penyelesaian

1. Studi pustaka mengenai algoritma yang akan digunakan

Pada tahap ini, penulis melakukan pemahaman dan memperdalam logika dan langkah kerja dari algoritma PSO, HBA, dan DTBO. Hal yang dilakukan penulis pada tahap ini adalah banyak membaca dari berbagai sumber literatur yang terkait dengan algoritma tersebut.

2. Menyusun ketiga algoritma berdasarkan studi pustaka

   Pada tahap ini, penulis akan menyusun ketiga algoritma tersebut berdasarkan pemahaman yang didapat dari studi pustaka sebelumnya. Penyusunan algoritma tersebut dilakukan dengan menggunakan python.

3. Melakukan optimisasi *benchmark functions* dengan algoritma yang sudah dibuat

   Pada tahap ini, penulis akan melakukan optimisasi dengan algoritma PSO, HBA, dan DTBO yang sudah dibuat. Adapun fungsi permasalahan yang digunakan adalah *benchmark functions* yang berjumlah 10 fungsi. Banyaknya iterasi yang digunakan di setiap fungsinya adalah sebanyak 1000 kali dan akan dilakukan pengulangan optimisasi sebanyak 30 kali di setiap fungsinya. Adapun *benchmark functions* yang akan digunakan terlihat pada tabel (2.1).

4. Membandingkan performa optimisasi dari ketiga algoritma

   Pada tahap ini penulis akan membandingkan performa ketiga algoritma tersebut setelah melakukan optimisasi. Perbandingan performa ketiga algoritma gabungan tersebut akan ditampilkan dalam grafik maupun tabel agar memudahkan pembaca dalam memahami perbandingan tersebut. Adapun performa yang akan dibandingkan adalah mean, median, standar deviasi, nilai terbaik, nilai terburuk, standar deviasi (std), standar error, dan Mean Absolute Error (MAE) dari 30 percobaan yang dilakukan. Di sini kita bisa mengetahui algoritma mana yang paling baik dalam melakukan optimisasi.

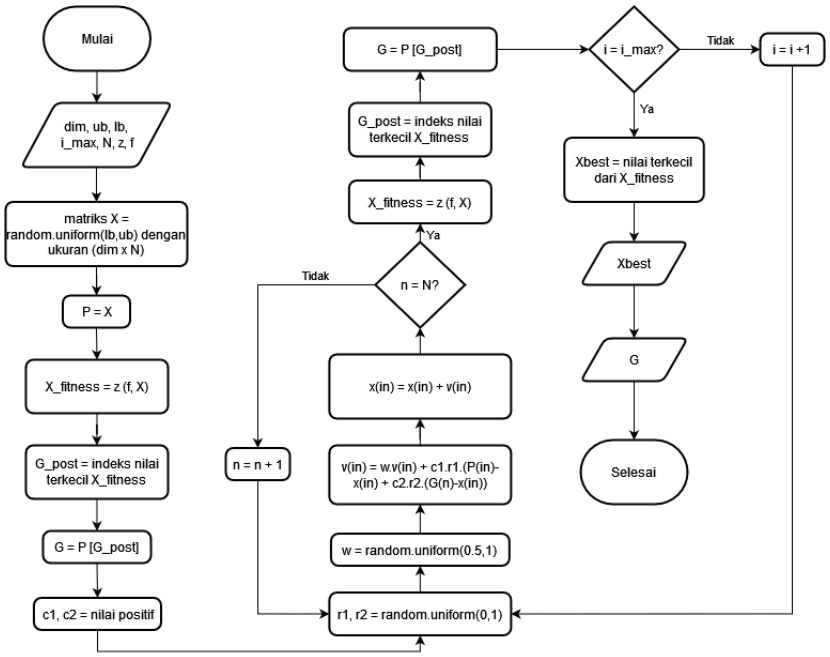
BAB IV
HASIL DAN PEMBAHASAN

4.1 Merancang algoritma metaheuristik

Dalam penelitian ini, penulis akan membangun tiga algoritma metaheuristik, yaitu PSO, HBA, dan DTBO berdasarkan studi pustaka yang telah dilakukan pada Bab II. Ketiga metode metaheuristik tersebut digunakan untuk melakukan optimisasi dalam menentukan nilai minimum dari suatu persamaan.

4.1.1 Merancang algoritma PSO

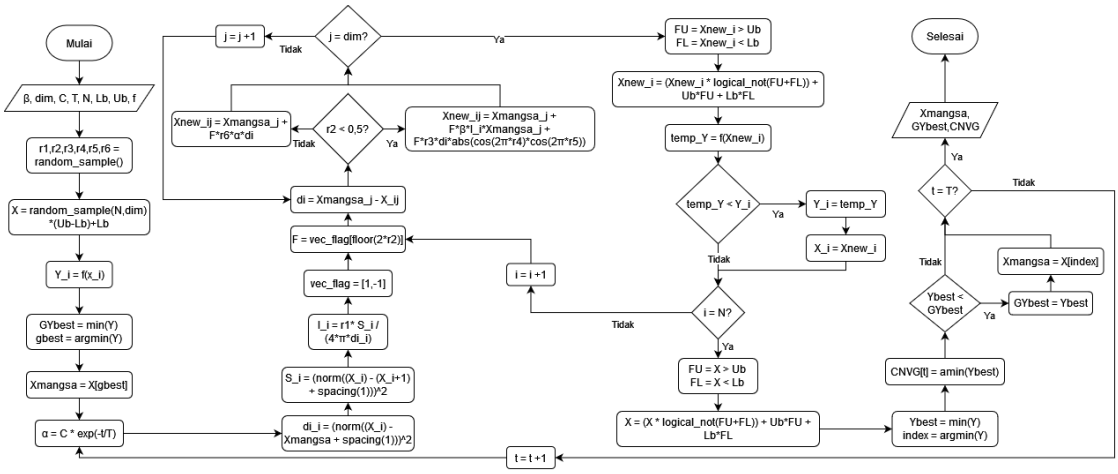
Gambar (4.1) merupakan diagram alir yang dibuat berdasarkan cara kerja algoritma PSO. algoritma PSO yang digunakan dalam penelitian ini dibuat berdasarkan algoritma yang dirancang oleh Ken Moriwaki (Moriwaki, 2022). Ken Moriwaki merancang algoritma PSO ini berdasarkan algoritma modifikasi PSO dengan menggunakan beban inersia.



Gambar 4.1. Diagram alir algoritma PSO

4.1.2 Merancang algoritma HBA

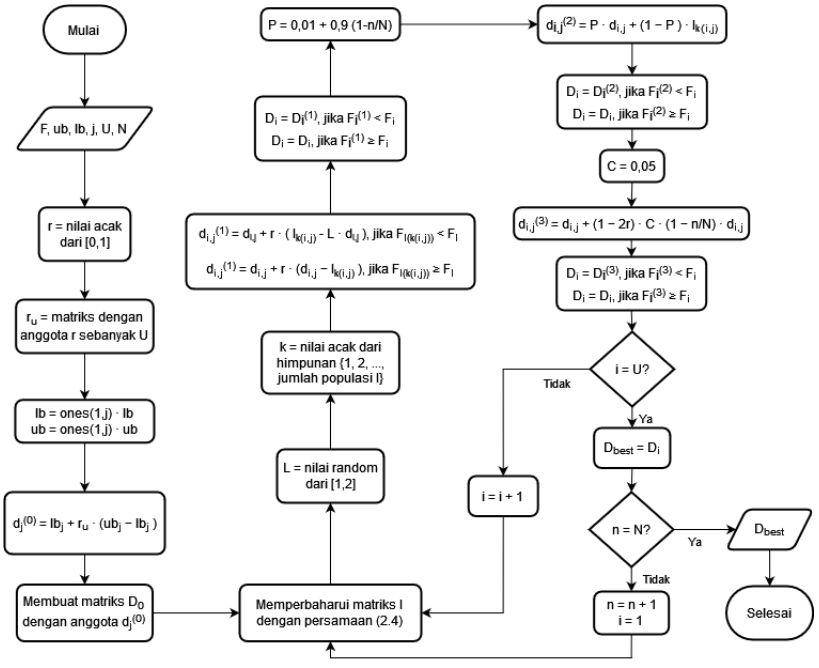
Gambar (4.2) merupakan diagram alir yang dibuat berdasarkan cara kerja algoritma HBA. algoritma HBA yang digunakan dalam penelitian ini dibuat berdasarkan algoritma yang dirancang oleh Essam H Houssein (Hashima et al., 2022).



Gambar 4.2. Diagram alir algoritma HBA

4.1.3 Merancang algoritma DTBO

Gambar (4.3) merupakan diagram alir yang dibuat berdasarkan cara kerja algoritma DTBO. algoritma DTBO yang digunakan dalam penelitian ini dibuat berdasarkan algoritma yang dirancang oleh Mohammad Dehghani (Dehghani et al., 2022).



Gambar 4.3. Diagram alir algoritma DTBO

4.2 Hasil dan perbandingan optimisasi algoritma terhadap *benchmark functions*

Setelah merancang algoritma, penulis melakukan optimisasi *benchmark functions* dengan menggunakan ketiga algoritma yang telah dibuat. Seperti yang telah dibahas di Bab II dan Bab III, penulis menggunakan sepuluh *benchmark functions* yang tertera pada tabel (2.1). Fungsi $f_1 - f_4$ merupakan fungsi unimodal, fungsi $f_5 - f_7$ merupakan fungsi multimodal berdimensi banyak , dan fungsi $f_8 - f_{10}$ merupakan fungsi multimodal berdimensi sedikit. Setiap fungsi tersebut akan dilakukan

optimisasi sebanyak 30 kali percobaan di setiap algoritma yang telah dibuat. Hasil optimisasi dari ketiga algoritma tersebut terlampir pada lampiran A. Penulis juga akan membandingkan performa ketiga algoritma yang telah dibuat dalam melakukan optimisasi *benchmark functions*. Kriteria yang akan dibandingkan pada penelitian ini adalah nilai mean, median, nilai terbaik, nilai terburuk, standar deviasi (std), standar error, dan Mean Absolute Error (MAE). Sebuah algoritma dikatakan memiliki performa yang baik pada optimisasi suatu fungsi bila memenuhi kriteria sebagai berikut:

1. Nilai mean, median, dan nilai terbaik mendekati nilai solusi minimum.
2. Memiliki nilai selisih antara nilai terbaik dan nilai terburuk yang minimal.
3. Nilai yang dihasilkan setiap percobaan selalu konsisten (nilai std dan standar error yang minimal).
4. Memiliki nilai MAE yang minimal.

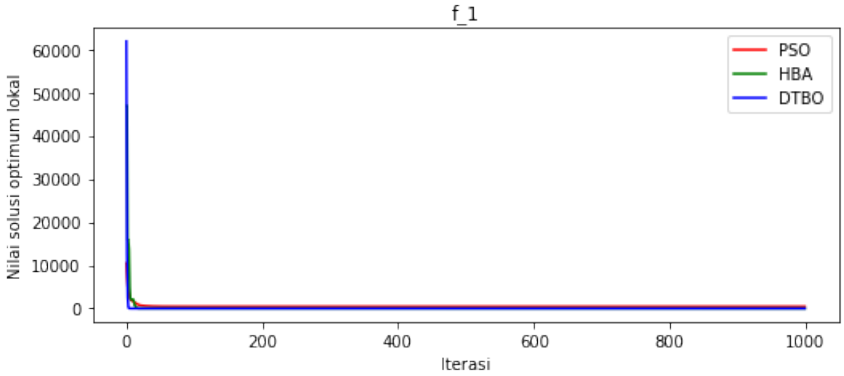
4.2.1 Perbandingan optimisasi pada fungsi f_1

Tabel 4.1. Tabel perbandingan performa ketiga algoritma pada fungsi f_1

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F1	Mean	4,402E+02	5,278E-285	0,000E+00
	Nilai Terbaik	1,955E+02	8,685E-299	0,000E+00
	Nilai Terburuk	8,889E+02	1,559E-283	0,000E+00
	Std	2,051E+02	0,000E+00	0,000E+00
	Median	3,913E+02	3,388E-290	0,000E+00
	Standar Error	3,744E+01	0,000E+00	0,000E+00
	MAE	4,402E+02	5,278E-285	0,000E+00
	Rank	3	2	1

Berdasarkan tabel (4.1), penulis mendapatkan *insight* bahwa pada fungsi f_1, PSO memiliki nilai solusi optimum terbaik (1,955E+02) yang jauh dari nilai solusi minimum (f_{min}) dibanding pada algoritma DTBO (0,000E+00) dan HBA (8,685E-299). Ini juga dibuktikan dengan nilai MAE PSO yang lebih besar (4,402E+02) dibanding pada HBA (5,278E-285) dan DTBO (0,000E+00). Selain itu, pada algoritma PSO menghasilkan nilai solusi optimum yang tidak konsisten. Hal ini dibuktikan dengan nilai standar deviasi (2,051E+02) dan standar error (3,744E+01) yang cukup tinggi pada PSO.

Adapun pada HBA dan DTBO terdapat sedikit perbedaan dalam performa nya. Nilai standar deviasi dan standar error yang pada kedua algoritma tersebut memiliki nilai yang sama (0,000E+00), meski pun pada algoritma HBA terdapat ketidak konsisten pada nilai yang dihasilkan. Menurut penulis, hal ini disebabkan karena pada HBA menghasilkan nilai yang sangat kecil pada setiap percobaan. Sehingga pada pencarian standar deviasi dan standar error, nilai yang dihasilkan sangat kecil sehingga dianggap sebagai nilai nol. Ini juga terlihat pada nilai MAE nya yang sangat kecil (5,278E-285). Pada fungsi ini, DTBO unggul dalam melakukan pencarian solusi optimum dengan hasil yang mencapai nilai minimum secara konsisten pada setiap percobaan.



Gambar 4.4. Grafik perbandingan performa ketiga algoritma pada fungsi f_1

4.2.2 Perbandingan optimisasi pada fungsi f_2

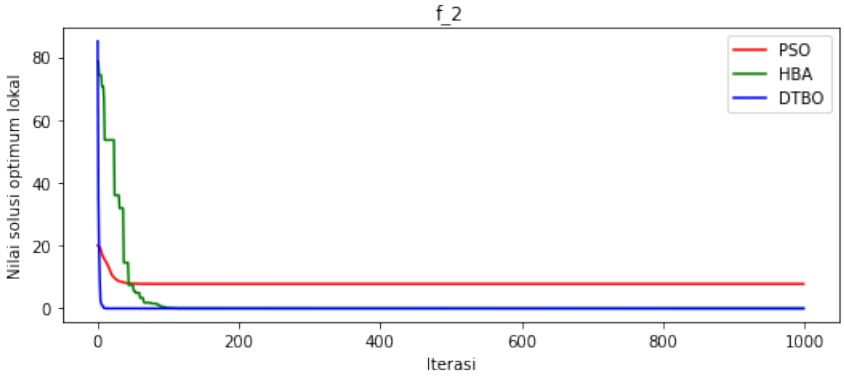
Tabel 4.2. Tabel perbandingan performa ketiga algoritma pada fungsi f_2

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F2	Mean	7,365E+00	5,834E-121	0,000E+00
	Nilai Terbaik	4,142E+00	1,669E-129	0,000E+00
	Nilai Terburuk	1,003E+01	1,664E-119	0,000E-01
	Std	1,459E+00	2,983E-120	0,000E+00
	Median	7,412E+00	1,246E-124	0,000E+00
	Standar Error	2,664E-01	5,446E-121	0,000E+00
	MAE	7,365E+00	5,834E-121	0,000E+00
	Rank	3	2	1

Untuk fungsi f_2, perbedaan performa ketiga algoritma jelas terlihat pada grafik (4.5). Perbedaan performa juga terlihat mencolok pada (4.2) di mana PSO memiliki nilai solusi optimum terbaik (4,142E+00) yang jauh dari f_{min} dibanding pada HBA (1,669E-129) dan DTBO (0,000E+00). Ini lah yang membuat nilai MAE pada PSO (7,365E+00) begitu tinggi dibanding HBA (5,834E-121) dan DTBO (0,000E+00). Selain itu nilai standar deviasi dan standar error pada PSO (1,459E+00 dan 7,412E+00) juga lebih tinggi dibanding pada HBA (2,983E-120 dan 5,446E-121) dan DTBO (0,000E+00 dan 0,000E+00) yang membuktikan nilai yang dihasilkan setiap percobaan pada PSO tidak se konsisten pada HBA dan DTBO. Walaupun HBA dan DTBO memiliki performa yang hampir sama pada fungsi ini, tetapi DTBO memiliki sedikit keunggulan di mana algoritma ini berhasil mencapai nilai f_{min} fungsi f_2. Hal ini terbukti pada nilai MAE yang mencapai nilai nol (0,000E+00).

4.2.3 Perbandingan optimisasi pada fungsi f_3

Pada fungsi ini, ketiga algoritma menghasilkan nilai solusi optimum yang masih terlalu jauh dari nilai f_{min}. Hal ini dikarenakan pada fungsi ini, nilai MAE dari ketiga algoritma ini cukup tinggi, di mana PSO sebesar (1,244E+04), HBA sebesar (2,190E+01), dan DTBO sebesar (2,773E+01). Walaupun begitu, perbedaan performa dari ketiga algoritma ini masih bisa dilihat pada tabel 4.3 dan grafik (4.6) di mana HBA merupakan algoritma dengan performa terbaik. Ini dibuktikan dengan

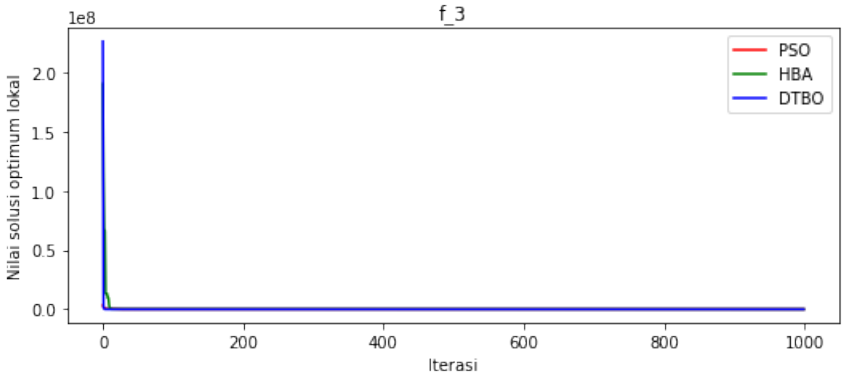


Gambar 4.5. Grafik perbandingan performa ketiga algoritma pada fungsi f_2

Tabel 4.3. Tabel perbandingan performa ketiga algoritma pada fungsi f_3

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F3	Mean	1,244E+04	2,190E+01	2,773E+01
	Nilai Terbaik	3,611E+03	2,100E+01	2,589E+01
	Nilai Terburuk	3,750E+04	2,392E+01	2,885E+01
	Std	8,475E+03	5,764E-01	9,803E-01
	Median	1,114E+04	2,179E+01	2,785E+01
	Standar Error	1,547E+03	1,052E-01	1,790E-01
	MAE	1,244E+04	2,190E+01	2,773E+01
	Rank	3	1	2

HBA mendapatkan nilai solusi optimum (2,100E+01) yang paling mendekati nilai f_{min} dibanding algoritma lain yang dibandingkan. HBA juga memiliki nilai standar deviasi (5,764E-01) dan standar error (1,052E-01) terbaik dibanding algoritma lain, di mana hal ini menandakan bahwa algoritma HBA menghasilkan nilai yang lebih konsisten.



Gambar 4.6. Grafik perbandingan performa ketiga algoritma pada fungsi f_3

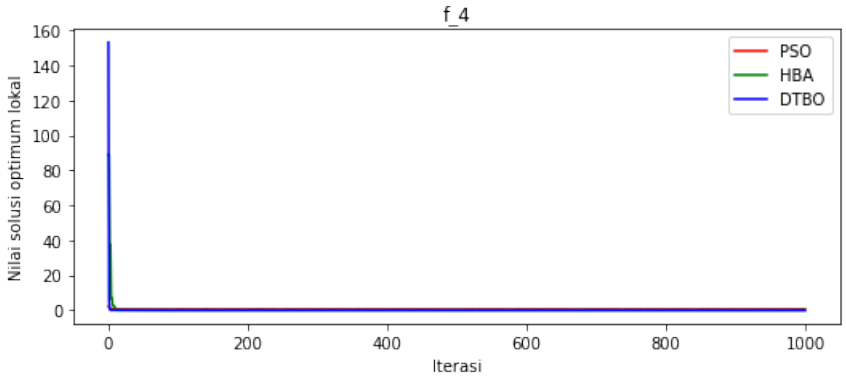
4.2.4 Perbandingan optimisasi pada fungsi f_4

Berdasarkan tabel (4.4), ketiga algoritma dalam mencari nilai solusi optimum memiliki performa yang tidak terlalu jauh perbandingannya antara satu sama lainnya, seperti yang juga terlihat pada

Tabel 4.4. Tabel perbandingan performa ketiga algoritma pada fungsi f_4

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F4	Mean	9,573E-02	1,639E-04	3,202E-05
	Nilai Terbaik	2,085E-02	2,094E-05	1,123E-07
	Nilai Terburuk	2,180E-01	7,160E-04	1,494E-04
	Std	5,027E-02	1,354E-04	3,159E-05
	Median	9,530E-02	1,429E-04	2,218E-05
	Standar Error	9,178E-03	2,472E-05	5,768E-06
	MAE	9,573E-02	1,639E-04	3,202E-05
	Rank	3	2	1

grafik (4.7). Algoritma PSO menjadi algoritma dengan performa terburuk dalam mencari solusi optimum dengan nilai MAE sebesar (9,573E-02), disusul dengan HBA sebesar (1,639E-04), dan DTBO sebesar (3,202E-05). Untuk konsistensi nilai, DTBO memiliki performa terbaik dengan mendapatkan nilai standar deviasi dan standar error terendah ,yaitu masing-masing sebesar (3,159E-05) dan (5,768E-06). Tidak ada algoritma yang mencapai nilai f_{min} pada fungsi f_4. Algoritma yang menghasilkan nilai solusi optimum terdekat dari f_{min} adalah DTBO (1,123E-07).



Gambar 4.7. Grafik perbandingan performa ketiga algoritma pada fungsi f_4

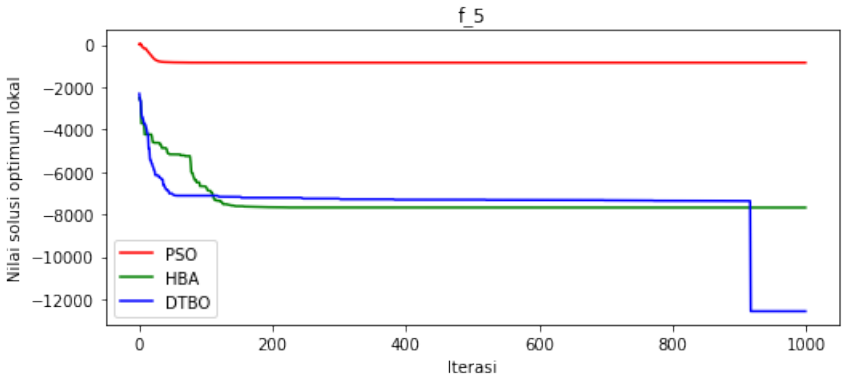
4.2.5 Perbandingan optimisasi pada fungsi f_5

Tabel 4.5. Tabel perbandingan performa ketiga algoritma pada fungsi f_5

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F5	Mean	-8,023E+02	-8,281E+03	-9,701E+03
	Nilai Terbaik	-1,176E+03	-1,009E+04	-1,257E+04
	Nilai Terburuk	-5,492E+02	-5,396E+03	-6,052E+03
	Std	1,629E+02	1,301E+03	2,006E+03
	Median	-7,755E+02	-8,756E+03	-9,016E+03
	Standar Error	2,973E+01	2,376E+02	3,663E+02
	MAE	1,177E+04	4,289E+03	2,868E+03
	Rank	3	2	1

Pada fungsi f_5, seperti yang terlihat pada tabel (4.5) ketiga algoritma memiliki masalah yang

sama dalam konsistensi nilai dalam setiap percobaan nya. Ini dapat dibuktikan dengan nilai standar deviasi dan standar error yang cukup tinggi, baik pada PSO (1,629E+02 dan 2,973E+01), HBA (1,301E+03 dan 2,376E+02), dan DTBO (2,006E+03 dan 3,663E+02). Jarak antara nilai terbaik dan nilai terburuk ketiga algoritma ini juga sangat jauh jaraknya, di mana PSO sebesar (-1,176E+03) dan (-5,492E+02), HBA sebesar (-1,009E+04) dan (-5,396E+03), dan DTBO sebesar (-1,257E+04) dan (-6,052E+03). Ketidak konsisten nilai juga membuat nilai MAE ketiga algoritma juga lebih besar dibanding pada fungsi lain, dengan PSO sebesar (1,177E+04) , HBA sebesar (4,289E+03), dan DTBO sebesar (2,868E+03). DTBO merupakan algoritma yang memiliki performa terbaik dengan nilai solusi optimum paling mendekati f_{min} dengan nilai sebesar (-1,257E+04). Mengejutkannya, PSO memiliki nilai standar deviasi (1,629E+02) dan standar error (2,973E+01) yang lebih superior dibanding algoritma lain yang dibandingkan, yang menandakan algoritma menghasilkan nilai yang lebih konsisten dibanding algoritma lain. Namun begitu, hal tersebut tidak membantu banyak dalam mencapai nilai f_{min} dari fungsi f_5. Perbandingan performa juga dapat terlihat jelas pada grafik (4.8).



Gambar 4.8. Grafik perbandingan performa ketiga algoritma pada fungsi f_5

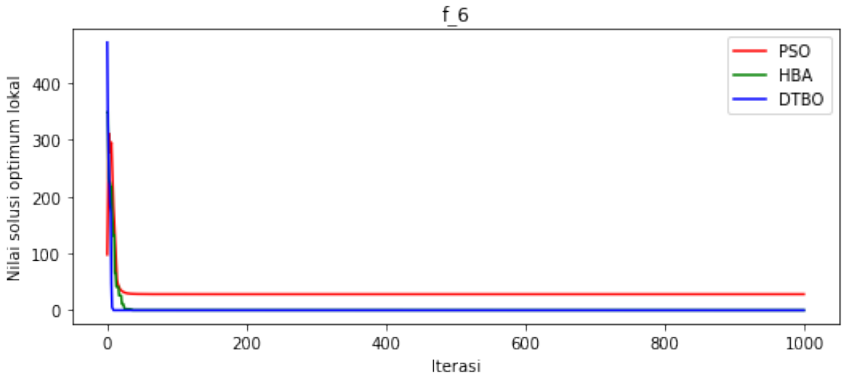
4.2.6 Perbandingan optimisasi pada fungsi f_6

Tabel 4.6. Tabel perbandingan performa ketiga algoritma pada fungsi f_6

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F6	Mean	5,195E+01	0,000E+00	0,000E+00
	Nilai Terbaik	3,019E+01	0,000E+00	0,000E+00
	Nilai Terburuk	1,473E+02	0,000E+00	0,000E+00
	Std	2,527E+01	0,000E+00	0,000E+00
	Median	4,251E+01	0,000E+00	0,000E+00
	Standar Error	4,614E+00	0,000E+00	0,000E+00
	MAE	5,195E+01	0,000E+00	0,000E+00
	Rank	3	1	1

Pada fungsi f_6, seperti yang terlihat (4.6), HBA dan DTBO memiliki performa yang kurang lebih sama dan jauh meninggalkan PSO dalam berbagai faktor, baik dalam mencari nilai solusi minimum maupun dalam konsistensi data. PSO merupakan satu-satunya algoritma yang tidak mencapai nilai f_{min} (bisa dilihat pada grafik (4.9)) dengan nilai terbaik sebesar (3,019E+01), nilai terburuk sebesar

(1,473E+02), dan MAE sebesar (5,195E+01). Adapun dalam konsistensi data, PSO memiliki performa buruk dengan nilai standar deviasi sebesar (2,527E+01) dan standar error sebesar (4,614E+00). Pada fungsi ini, baik algoritma HBA dan DTBO adalah algoritma terbaik dalam mencari nilai solusi optimum, di mana nilai standar deviasi, standar error dan MAE yang memiliki nilai yang sama yaitu (0,000E+00).



Gambar 4.9. Grafik perbandingan performa ketiga algoritma pada fungsi f_6

4.2.7 Perbandingan optimisasi pada fungsi f_7

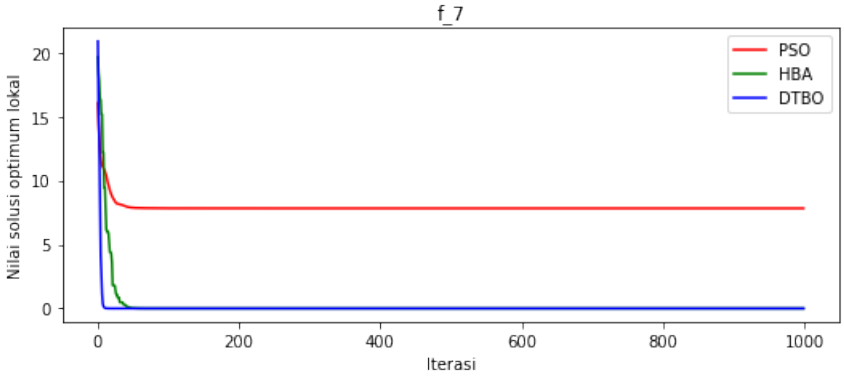
Tabel 4.7. Tabel perbandingan performa ketiga algoritma pada fungsi f_7

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F7	Mean	5,561E+00	1,328E+00	4,441E-16
	Nilai Terbaik	4,357E+00	4,441E-16	4,441E-16
	Nilai Terburuk	7,243E+00	1,993E+01	4,441E-16
	Std	6,993E-01	4,967E+00	0,000E+00
	Median	5,428E+00	4,441E-16	4,441E-16
	Standar Error	1,277E-01	9,069E-01	0,000E+00
	MAE	5,561E+00	1,328E+00	4,441E-16
	Rank	3	2	1

Seperti yang terlihat pada tabel (4.7) dan grafik (4.10), performa PSO dalam pencarian solusi optimum yang jauh lebih rendah dari HBA dan DTBO. Hal tersebut dibuktikan dengan nilai MAE PSO (5,561E+00) yang lebih tinggi dibanding pada HBA (1,328E+00) dan DTBO (4,441E-16). Adapun konsistensi data, PSO sedikit lebih unggul dibanding dengan HBA di mana standar deviasi dan standar error dari PSO sebesar (6,993E-01) dan (1,277E-01), serta HBO sebesar (4,967E+00) dan (9,069E-01). Namun konsistensi data nya masih lebih buruk dibanding DTBO (0,000E+00 dan 0,000E+00). Pada fungsi ini, DTBO memiliki performa terbaik dengan nilai terbaik sebesar (4,441E-16).

4.2.8 Perbandingan optimisasi pada fungsi f_8

Seperti yang terlihat pada tabel (4.8) dan grafik (4.11), penulis mendapat *insight* yang unik dan menarik. Pada fungsi ini, ketiga algoritma berhasil mencapai f_{min}. Uniknya, DTBO menghasilkan

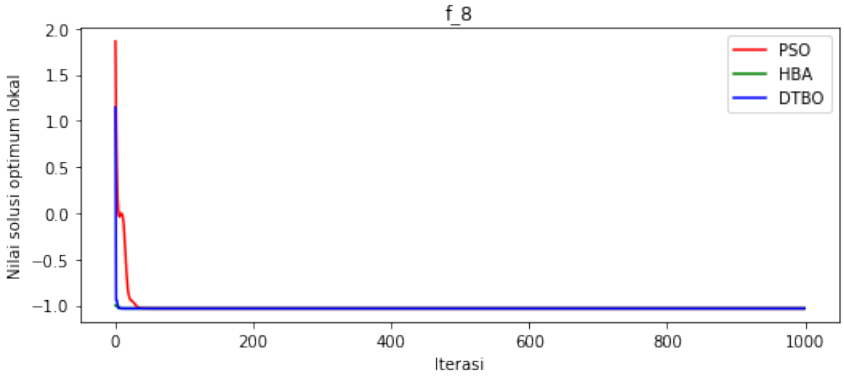


Gambar 4.10. Grafik perbandingan performa ketiga algoritma pada fungsi f_7

Tabel 4.8. Tabel perbandingan performa ketiga algoritma pada fungsi f_8

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F8	Mean	-1,032E+00	-1,032E+00	-9,901E-01
	Nilai Terbaik	-1,032E+00	-1,032E+00	-1,032E+00
	Nilai Terburuk	-1,032E+00	-1,032E+00	-2,155E-01
	Std	0,000E+00	0,000E+00	1,448E-01
	Median	-1,032E+00	-1,032E+00	-1,032E+00
	Standar Error	0,000E+00	0,000E+00	2,643E-02
	MAE	4,651E-08	4,651E-08	4,149E-02
	Rank	1	1	3

nilai solusi optimum yang tidak konsisten serta memiliki performa paling buruk. Ini bisa dibuktikan dengan nilai standar deviasi, standar error, dan MAE yang tertinggi dibanding dua algoritma lain yang dibandingkan. Hal ini terjadi karena pada DTBO tidak terdapat variabel yang mengatur keseimbangan antara fase-fase perhitungannya sehingga DTBO sering kesulitan dalam melakukan optimisasi pada fungsi berdimensi rendah (seperti yang telah disinggung pada BAB I). Lebih mengejutkan lagi, performa PSO di sini cukup baik dan bahkan menyamai performa HBA dalam melakukan pencarian nilai solusi optimum.



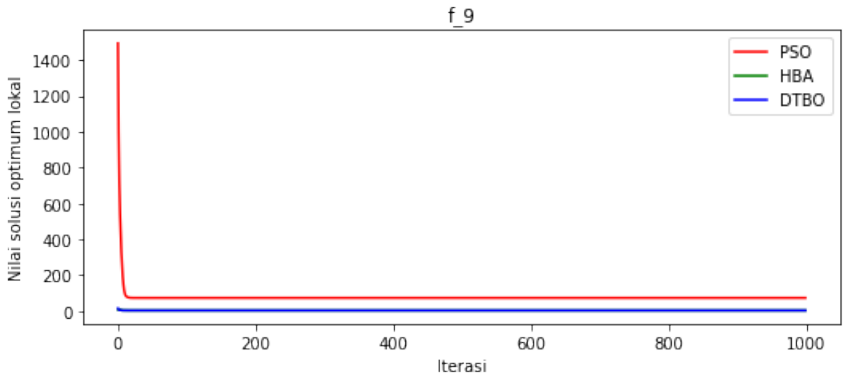
Gambar 4.11. Grafik perbandingan performa ketiga algoritma pada fungsi f_8

4.2.9 Perbandingan optimisasi pada fungsi f_9

Tabel 4.9. Tabel perbandingan performa ketiga algoritma pada fungsi f_9

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F9	Mean	7,296E+01	3,900E+00	4,919E+01
	Nilai Terbaik	7,296E+01	3,000E+00	3,010E+00
	Nilai Terburuk	7,296E+01	3,000E+01	8,400E+02
	Std	0,000E+00	4,847E+00	1,495E+02
	Median	7,296E+01	3,000E+00	3,528E+00
	Standar Error	0,000E+00	8,849E-01	2,730E+01
	MAE	6,996E+01	9,000E-01	4,619E+01
	Rank	2	1	3

Seperti yang terlihat pada tabel (4.9 dan gambar (4.11), PSO menghasilkan nilai solusi optimum yang kurang akurat dibanding HBA dan DTBO, di mana nilai MAE PSO sebesar (6,996E+01), HBA sebesar (9,000E-01), dan DTBO sebesar (4,619E+01). Namun PSO unggul dalam konsistensi data dengan standar deviasi sebesar (0,000E+00) dan standar error sebesar (0,000E+00) sehingga membantu meningkatkan performa PSO. DTBO memiliki performa terburuk dalam konsistensi data yang dibuktikan dengan standar deviasi (1,495E+02) dan standar error (2,730E+01) tertinggi dibanding algoritma lain. Penyebabnya karena f_9 merupakan fungsi berdimensi rendah sehingga DTBO agak kesulitan untuk menghasilkan nilai yang konsisten. Adapun HBA merupakan algoritma dengan performa terbaik yang paling mendekati f_{min} dalam melakukan pencarian solusi optimum dengan nilai terbaik sebesar (3,000E+00).



Gambar 4.12. Grafik perbandingan performa ketiga algoritma pada fungsi f_9

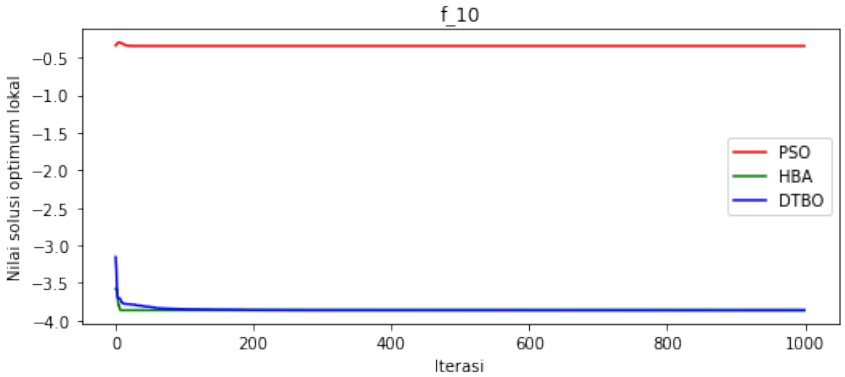
4.2.10 Perbandingan optimisasi pada fungsi f_{10}

Seperti yang terlihat pada tabel (4.10) dan grafik (4.13), terlihat sangat jelas sekali performa PSO dalam mencari nilai solusi optimum tertinggal jauh dibanding algoritma lain yang dibandingkan. Nilai solusi optimum terbaik dari PSO (-3,466E-01) sangat jauh dari f_{min} dibanding HBA (-3,863E+00) dan DTBO (-3,862E+00). Ini juga terlihat dari nilai MAE dari PSO yang lebih tinggi (3,513E+00) dibanding algoritma lain. Namun dalam segi konsistensi nilai, PSO menghasilkan nilai solusi yang

Tabel 4.10. Tabel perbandingan performa ketiga algoritma pada fungsi f_{10}

Fungsi	Performa	Algoritma		
		PSO	HBA	DTBO
F10	Mean	-3,466E-01	-3,863E+00	-3,158E+00
	Nilai Terbaik	-3,466E-01	-3,863E+00	-3,862E+00
	Nilai Terburuk	-3,466E-01	-3,855E+00	-1,001E+00
	Std	0,000E+00	1,415E-03	1,114E+00
	Median	-3,466E-01	-3,863E+00	-3,854E+00
	Standar Error	0,000E+00	2,583E-04	2,034E-01
	MAE	3,513E+00	2,859E-03	7,025E-01
	Rank	3	1	2

lebih konsisten dibanding algoritma lain, dengan nilai standar deviasi sebesar (0,000E+00) dan standar error sebesar (0,000E+00). Pada fungsi ini, algoritma HBA menghasilkan nilai solusi optimum paling mendekati nilai f_{min} dengan nilai terbaik sebesar (-3,863E+00).



Gambar 4.13. Grafik perbandingan performa ketiga algoritma pada fungsi f_{10}

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan penelitian dan analisis yang sudah dilakukan, penulis mendapatkan kesimpulan yang menarik. Perkembangan algoritma metaheuristik dari waktu kewaktu dapat menghasilkan algoritma yang lebih baik pula performa nya seperti yang terlihat di penelitian ini di mana HBA dan DTBO, yang di mana keduanya termasuk algoritma baru, mengungguli jauh performa dari PSO yang termasuk algoritma versi lama. Ketiga algoritma juga menampilkan performa yang berbeda-beda di setiap *benchmark functions* yang diuji coba. Performa DTBO lebih unggul dalam melakukan optimisasi fungsi-fungsi yang berdimensi banyak dan performa HBA lebih unggul pada fungsi-fungsi yang berdimensi sedikit. PSO juga sebenarnya memiliki performa yang lebih baik ketika melakukan optimisasi fungsi berdimensi sedikit, namun performa nya masih kalah jauh dibanding algoritma HBA dan DTBO.

5.2 Saran

Saran untuk penelitian selanjutnya yaitu untuk melakukan perbandingan optimisasi dengan menggunakan lebih banyak *benchmark functions* dan menggunakan lebih banyak algoritma yang akan dibandingkan. Penulis juga menyarankan agar juga melakukan perbandingan pada implementasi kasus matematika dalam kehidupan sehari-hari, seperti misalnya *wield beam design*, *pressure field design*, dan lain sebagainya.